

# **CLASSIFYING THE SURFACE TYPES OF CYCLING ROADS USING STREET- LEVEL IMAGES AND DEEP LEARNING: A COMPARATIVE STUDY**

Emma Bekaert

Student number: 01900257

Promotor: Prof. Dr. Haosheng Huang – Department of Geography and Geomatics

Words: 20 567

*Master dissertation submitted to obtain the degree of Master of Science in Geography and Geomatics*

Ghent University Faculty of Sciences

Academic year: 2023 - 2024

## PREFACE

Finally, my thesis is done! It was a bumpy road with a lot of challenges, but we made it to the final destination. In the process of making this, I learned a lot: I took a deep dive into deep learning, relearned how to have self-discipline, to be consistent and to embrace the inevitable shortcomings every work will have. I could not have done this without the help of some people, so I want to take the time to thank them here.

First of all, I want to thank my parents for making it possible for me to attend university, to raise me and show me how to be my own person. Without their constant support (and genes) I could not have done this. Next, I want to thank my boyfriend, Lennard Derudder. He helped create all of this by proofreading, collecting a lot of the data, offering advice and another perspective in things, listening to my worries and overall always being there for me. Of course, a key player in all of this was my promotor, Prof. Dr. Haosheng Huang, who was always available to answer my questions and offer solutions and different perspectives to my problems. I would definitely recommend him as a promotor to any student, he knows just how to push a student's performance to their best abilities. Some other people I would like to thank are everybody who contributed to the dataset: Lennard, Luca, Manou, Joren and Pieter and people that provided me with interesting (technical) input: Lieven Vanoverbeke, Kevin Baker, Kenzo Milleville, Bart de Wit and Xiaobing Wei.

Fun fact: The word 'test set', which is used a lot in this thesis, is (if you ignore the spaces) a palindrome!

## ENGLISH ABSTRACT

Promoting cycling as a mode of transportation requires providing cyclists with accurate information about the road conditions they will encounter, including the type of road surface. However, this information is often lacking, and there is a scarcity of systematic methods for providing it. This thesis addresses this gap by comparing the performance of different pre-trained Deep Learning models on a self-collected dataset of street-level images of cycling roads. The study begins by training seven pre-trained models - GoogLeNet, VGG16, ResNet, DenseNet, MobileNet, EfficientNet, and a VisionTransformer - on two differently split datasets: the interval-based dataset and the road-based dataset. All models are trained using the same hyperparameters. Next, automated hyperparameter optimization is employed to search for the optimal hyperparameters for the three best-performing models. Using these optimized hyperparameters, the models are retrained and compared to their versions without hyperparameter-tuning. The comparisons are made using performance metrics, confusion matrices, learning curves, and saliency maps. Finally, the best model configuration is evaluated on a separate dataset of images collected using different cameras, bicycles, and recording angles. Results indicate that of the compared model configurations, the DenseNet model with hyperparameter-tuning is the most suitable for this classification task. Notably, results also show that when using the interval-based data splitting method, there is some data leakage that causes a hidden overfitting problem. These findings highlight the need for a larger and more diverse dataset to enhance model performance, while showing that the resulting method provides a solid foundation for future research.

## NEDERLANDS ABSTRACT

Het promoten van fietsen als vervoersmiddel vereist dat fietsers nauwkeurige informatie krijgen over de wegen die ze zullen tegenkomen, zoals het type wegdek. Zulke informatie ontbreekt echter vaak en er zijn weinig systematische methoden om deze te verstrekken. Deze thesis probeert een antwoord te bieden op dit probleem door de prestaties van verschillende pre-trained Deep Learning modellen getraind op een handmatig verzamelde dataset van foto's van fietspaden op straatniveau te vergelijken. De studie begint met het trainen van zeven modellen - GoogLeNet, VGG16, ResNet, DenseNet, MobileNet, EfficientNet en een Vision Transformer - op twee datasets met verschillende data splitting methodes: de interval-gebaseerde dataset en de weg-gebaseerde dataset. Alle modellen worden in eerste instantie getraind met dezelfde hyperparameters, maar daarna wordt geautomatiseerde hyperparameteroptimalisatie gebruikt om de optimale hyperparameters te zoeken voor de drie best presterende modellen. Met deze geoptimaliseerde hyperparameters worden de modellen opnieuw getraind en vergeleken met hun versies zonder hyperparameterafstemming. Het vergelijken wordt uitgevoerd met behulp van prestatiestatistieken, verwarringsmatrices, leercurves en saliency maps. Ten slotte wordt de beste modelconfiguratie geëvalueerd op een aparte dataset van beelden die zijn verzameld met verschillende camera's, fietsen en opnamehoeken. De resultaten geven aan dat van de vergeleken modelconfiguraties, het DenseNet-model met hyperparameterafstemming het meest geschikt is voor deze classificatie-opgave. Opvallend genoeg laten de resultaten zien dat er bij gebruik van de intervalgebaseerde gegevensbemonsteringsmethode sprake is van enige data leakage, wat een verborgen overfitting probleem veroorzaakt. Deze bevindingen benadrukken de noodzaak van een grotere en meer diverse dataset om de modelprestaties te verbeteren, terwijl de bekomen methode een robuuste basis biedt voor toekomstig onderzoek.

# TABLE OF CONTENTS

1	Introduction.....	1
2	Literature review.....	4
2.1	Road surface classification.....	4
2.2	Image processing based on deep learning.....	6
2.2.1	Computer vision.....	6
2.2.2	Image classification methods.....	7
2.2.3	State-of-the-art models and their architectures.....	10
2.3	Transfer learning and hyperparameter tuning.....	12
2.4	Research gaps.....	15
3	Method.....	16
3.1	Overview of methodology.....	16
3.2	Defining classes of road surface types.....	17
3.3	Data collection, labelling and splits for model building.....	21
3.3.1	Data collection.....	21
3.3.2	Data split ratio and sampling.....	22
3.4	Comparison of methods.....	24
3.4.1	Models used in this thesis.....	24
3.4.2	Aim of the evaluation.....	25
3.4.3	Evaluation metrics.....	26
3.5	Experiment: comparing different image capturing methods.....	28
4	Results.....	30
4.1	Resulting dataset.....	31
4.2	Comparison of different models and datasets without hyperparameter tuning.....	33
4.2.1	Performance metrics.....	33
4.2.2	Confusion matrices.....	34
4.2.3	Saliency maps.....	37
4.2.4	Learning curves.....	40
4.3	Comparison of different models with hyperparameter tuning.....	42
4.3.1	Hyperparameter optimisation results.....	42
4.3.2	Performance metrics.....	43

4.3.3	Confusion matrices .....	43
4.3.4	Saliency maps.....	45
4.3.5	Learning curves .....	45
4.4	Total comparison .....	46
4.5	Experiment: data capturing methods.....	47
4.5.1	Influence of blur on performance .....	47
4.5.2	Influence of data collection methods on blur and performance.....	50
5	Discussion .....	54
5.1	Main findings .....	54
5.2	Limitations .....	56
5.3	Future research .....	58
6	Conclusion.....	59
7	References .....	61
8	Appendices.....	66

## LIST OF FIGURES

Figure 1 Satellite image showing a road with cobblestones in Kanegem (Belgium) .....	5
Figure 2 Street-level image taken from a bicycle showing a road with cobblestones in Kanegem (Belgium) .....	5
Figure 3 Structure of a classical CNN .....	8
Figure 4 Overview of the Vision Transformer model .....	9
Figure 5 Inception moduel with dimensionality reduction within GoogLeNet .....	10
Figure 6 Left: a standard building block, right: a 'bottleneck' building block for ResNet-50/101/152....	11
Figure 7 Overview of the different parts in the method .....	16
Figure 8 Simplified example of the interval-based splitting approach .....	23
Figure 9 Simplified example of the road-based splitting approach .....	23
Figure 10 Different ways of mounting a recording device data collection while cycling, from left to right: GoPro on chest mount, GoPro mounted with an arm, Phone on a flat bike mount and Phone on a bike mount an with arm .....	29
Figure 11 Map of data collection test course in Kanegem (Belgium) with examples of the different road surface types .....	30
Figure 12 Confusion matrices for each model without optimised hyperparameters for the interval-based dataset .....	35
Figure 13 Confusion matrices for each model without optimised hyperparameters for the road-based datase .....	36
Figure 14 Saliency maps for the seven models with the interval-based dataset .....	38
Figure 15 Saliency maps for the seven models with the road-based dataset.....	39
Figure 16 Train accuracy learning curve comparison for the seven models without optimised hyperparameters using the interval-based dataset .....	40
Figure 17 DenseNet train and test accuracy comparison with the interval-based dataset .....	40
Figure 18 Train accuracy learning curve comparison for the seven models without optimised hyperparameters using the road-based dataset .....	41
Figure 19 DenseNet train and test accuracy comparison with the road-based dataset .....	41
Figure 20 Confusion matrices for the models with hyperparameter optimisation using the road-based dataset.....	44
Figure 21 Saliency maps of models with hyperparameter optimisation with the road-based dataset ..	45
Figure 22 Hyperparameter-tuned DenseNet train and test accuracy comparison using the road-based dataset .....	46
Figure 23 F1 scores for all models for the different datasets with or without hyperparameter optimisation, sorted from higher to lower score .....	46
Figure 24 Accuracy per blur level.....	47
Figure 25 Confusion matrix for blur level 0.....	48
Figure 26 Confusion matrix for blur level 5.....	48
Figure 27 Mean Laplacian scores per class for the different blur levels .....	49
Figure 28 Mean Laplacian scores per class for the total dataset.....	49

Figure 29 Confusion matrices for the different combinations of camera, bicycle and attachment method with the best performing model and the road-based dataset ..... 53

**LIST OF TABLES**

Table 1 Openstreetmap (OSM) Classes and their abundance ..... 17

Table 2 Chosen surface types with description and examples ..... 19

Table 3 Camera, bicycle and attachment combinations with specifications ..... 29

Table 4 Dataset summary per class and subset ..... 31

Table 5 Example images in various conditions for the six road surface types..... 32

Table 6 Result for the seven models after training on the interval-based dataset ..... 33

Table 7 Result for the seven models after training on the road-based dataset ..... 34

Table 8 Results of hyperparameter optimisation with Optuna for the road-based dataset ..... 42

Table 9 Result for the three best models after training on the road-based dataset with their optimal hyperparameters ..... 43

Table 10 Example images of the ground class in different blur levels ..... 47

Table 11 Different camera, bicycle and attachment combinations with their resulting accuracy, mean Laplacian scores and an example..... 50



# 1 INTRODUCTION

Promoting cycling is crucial for both mental and physical health. Regular cycling has been shown to improve cardiovascular fitness, reduce the risk of chronic diseases such as type 2 diabetes and certain types of cancer, and enhance overall well-being (Ming Wen & Rissel, 2008; Oja et al., 2011). Additionally, cycling has been linked to improved mental health, with studies demonstrating that it can reduce symptoms of depression and anxiety and increase feelings of happiness and life satisfaction (Biddle et al., 2019; Oja et al., 2011). However, the success of efforts to encourage recreational cycling is heavily dependent on the quality and safety of the cycling infrastructure, including the surface of the roads (Winters et al., 2010). When a race cyclist goes cycling, it is very likely that they want to avoid any gravel or dirt roads to protect their tires and a mountain biker will experience a route with a lot of asphalt or concrete roads as very boring. Both will probably look on a map or routing website to decide where to go, and therefore (unknowingly) use the road surface attributes that are attached to the roads on OpenStreetMap. OpenStreetMap is a crowdsourced geographical database that combines objects with a lot of different attributes (OpenStreetMap Contributors, n.d.). These attributes, like the road surface type, are subjective, not obligatory and can differ in their specificity (paved instead of asphalt, cobblestones, ...). OSM is used in a lot of commercial cycling apps (like Strava and RouteYou), but also in policy making, as it is the biggest free geographical database in the world (Denneman, 2022). So because the OSM data is frequently exploited, it is important to keep it as complete and up to date as possible. This is done by OSM volunteers, who manually map their area. Creating methods to automate the collection of Volunteered Geographic Information (VGI) like this, could enhance the dataset and make collecting more efficient and low-threshold. Examples where OSM data is collected more automatically, are MapComplete and StreetComplete, which are apps that ask the user to add specific attributes of geographical entities in a user-friendly way (*MapComplete*, n.d.; *StreetComplete*, n.d.).

To automate this process of road surface classification and enhance it, several types of input data can be used. The oscillation of the vehicle as collected by an accelerometer indicates where there are bumps on the road, and based on the amount of bumps, a road surface class can be allocated. Another way to collect the characteristics of the surface is to create a point cloud of the road surface with Lidar technology, which is a very precise but expensive method. Using available remote sensing data would be more affordable, but a birds-eye view of the road does not offer enough details to distinguish different road classes and is often obstructed by vegetation. A more detailed source of road surface imagery are street-level images, where images are recorded at ground level while standing (or driving) on the street itself. This approach requires more effort to collect, but is detailed and widely applicable.

In the case of this thesis, images will be used as input to classify the bicycle road surface types. Image classification is one of the many computer vision tasks and can be done with a range of methods, such as Random Forest, K-Nearest Neighborhood, Support Vector Machine, Neural Networks and Vision Transformers. Most methods have already been applied to the task of road surface classification (mostly for car roads), but a comparison of different Convolutional Neural Networks and Vision Transformers,

which are currently the methods with the most potential, has not been carried out for bicycle paths. Convolutional Neural Networks (CNN) consist of different specialised layers with nodes that are connected to each other and use a combination of these layers to find the difference between classes. It is inspired by the biological structure of how thinking works in animal brains. Vision Transformers (ViT) combine local and global information and use attention mechanisms to distinguish classes.

For the most part, existing road surface classification methods are tailored to cars, where the data collection happens in a fairly stable environment, as cars have good shock absorption and drive mostly over paved roads. The need for road surface evaluation in cars grows with the development of autonomous vehicles, where the driving style should be adjusted in real time according to the status of the road. A related task is detecting potholes and cracks in the road, which uses similar computer vision techniques and is well-researched.

Compared to this, there is less focus on the classification of bicycle paths, where the data collection is more shaky and more unpaved roads are present. Both accelerometer data and image data have been used to classify bicycle road surfaces, using different methods and classes. Verstockt et al. (2013) use a Random Forest algorithm to classify cycle path images from a testing route into six classes, and reach an accuracy of 90%. Hoffmann et al. (2013) developed a K-Nearest Neighbour model to classify into three classes, resulting in an accuracy of 77.45%. More recently, MobileNet (a Convolutional Neural Network), was used by Beecken & Reinhardt (2019) to classify cycle paths into five classes, reaching an accuracy of 99.4%. However, there is no in-depth comparison of state-of-the-art classification methods, where different aspects of the image classification process are discussed.

In general, a method to collect bicycle road surface imagery and accurately classify them is needed to enhance information about bicycle infrastructure. This information is essential for assessing cycling comfort and promoting recreational cycling but is often lacking because cycling roads are less frequently travelled and can be rapidly altered by municipal interventions or natural processes. The (limited) research on cycling road classification has not utilised the whole array of different CNN models or the more recent Vision Transformers. A comparison of these models is therefore provided in this thesis. Furthermore, collecting image data on a bicycle poses specific potential issues. It is a time-intensive way of collecting data, and therefore the collected dataset might be small. It is known that the size of the dataset has a big influence on the model performance, and in this case it would be useful to analyse this influence. Also, bicycles are not steady vehicles, so collecting data on them will definitely introduce blurriness to the images. It would be useful to determine if the blurriness is a hindrance to the model performance, as most researchers discard these kinds of images, but it is inherent to the classification task at hand. Lastly, every road surface classification method uses different classes, ranging from binary classes such as 'unpaved' and 'paved' to more specific classes such as 'asphalt', 'grass', 'gravel' et cetera. Therefore it is difficult to compare all these classification methods. More discussion is needed on the distinctive characteristics of these classes.

To provide some insights and possibly narrow this existing research gap, this thesis explores the possibilities of using Deep Learning for image-based road surface classification. Therefore, the following research questions will be answered:

- Is it feasible to use Deep Learning methods to classify the surface types of cycling roads using street-level images?
  - How do different state-of-the-art image classification models perform on this task, and which model(s) are the most suitable?
  - How does hyperparameter tuning affect the model performance?
  - How do the models perform under different training/validation/test splitting methods?
  - Do the data collection methods and image quality influence the model performance?

The proposed method for classifying bicycle road surface types using CNN and ViT models is a comprehensive approach that involves data collection, preprocessing, model training, and performance evaluation. First, six road surface classes - asphalt, concrete, fine gravel, ground, paving stones and sett - are selected. Four volunteers go on cycling trips to collect video data for these classes with their smartphone or action camera. From the resulting videos, frames are extracted every five seconds, pre-processed and annotated to their respective classes. These annotated images are then divided into three sets: training (70%), testing (10%), and validation (20%). This split can be performed in two ways: by taking every 5<sup>th</sup> and 10<sup>th</sup> image and putting them into the smaller subsets (the interval-based approach) or by putting entire roads together for the subsets (the road-based approach). By using both these splitting approaches and comparing them, it can be detected if the similarity between subsets lead to an overly optimistic performance, which is a form of data leakage.

With the prepared datasets, seven pre-trained models are trained using the same hyperparameters: GoogLeNet, VGG16, ResNet, DenseNet, MobileNet, EfficientNet and Vision Transformer. These models are compared using performance metrics such as the training and testing accuracy, Cohen's Kappa and F1-score. The results are further interpreted via confusion matrices, learning curves and saliency maps, a form of explainable AI. The top three models then undergo an automated hyperparameter optimisation process, and their performances are compared against their 'default' versions.

Additionally, an experiment is conducted to assess the impact of different data collection methods on image blurriness and model performance. Two cameras (a smartphone and an action camera) are mounted on different bicycles (a mountain bike and a city bike) in different positions (directly on the handlebars, angled with an arm or mounted on the cyclist's chest) and used to film the same course. The results are compared based on their Laplacian score, which measures the number of edges in an image, and the performance of the best model configuration on these images.

This master's thesis uses the following structure. Firstly, related research on the field of road classification will be discussed to create a background for this thesis. Secondly, an overview will be provided on important terms, practices and architectures within the field of Computer Vision, Image Classification, Deep Learning and Convolutional Neural Networks in particular. After this general overview, the state-of-the-art model architectures and some used concepts and techniques are explored in greater depth in section 2.3.3. The research gaps are then explained. In section 3, the proposed methodology will be explained, starting with the chosen surface classes (section 3.2) and the way the dataset was collected, annotated and split (section 3.3). In part 3.4 of the method the techniques for comparing, visualising and optimising the models will be discussed and in the last part (section 3.5), the data collection experiment is explained. All results will be discussed in section 4, starting with the resulting dataset (section 4.1), the comparison of the models with or without hyperparameter optimisation (section 4.2 and 4.3) and ending with a total comparison of all models (section 4.4) and the results of the data collection experiment (section 4.5). The main findings, shortcomings and future research follow in the discussion section and are summarized in the conclusion (sections 5 and 6).

## 2 LITERATURE REVIEW

### 2.1 Road surface classification

The issue of road surface classification is an important one and has been a popular research topic. Most research has focused on the classification of car road surfaces, as cars have built-in parking sensors, cameras and accelerometers to monitor the surroundings and with the evolution of (semi-)automated vehicles, knowledge about the road conditions is necessary to adjust the driving style. Different kinds of input are used to carry out this task. One common source of information is the oscillation of the vehicle when driving on the street, which can be deduced from the accelerometer values of the car itself or a phone within the car. After all, the vertical motion of the car will be less intense when driving on a smooth asphalt road than over cobblestones or a speed bump. Varona et al. (2020) have used accelerometer data to identify four different kinds of road surface (concrete panels, dirt roads, asphalt roads and cobblestones) and at the same time detecting potholes and other road obstructions. They compared different Deep Learning methods, among which the best performing model was a Convolutional Neural Network (CNN), with an accuracy of 85% for the road surface classification. Convolutional Neural Networks are deep learning architectures that consist of different (hidden) layers and are commonly applied to image data (Gu et al., 2018). Next to accelerometer data, Lidar data is sometimes used to create a point cloud of the road surface and make it possible to identify potholes and road surface types. This approach is difficult to apply for large-scale studies, as devices to capture Lidar images are expensive.

Consequently, images are often used to classify the road surface. A first and commonly available source of surface imagery is remote sensing data. In this research area, there is more focus on detecting where on the earth's surface the roads are rather than finding out their surface type, as the roads are often

obstructed by vegetation and the image quality is not precise enough to detect specific road types. So, most research focuses on using street-level images to classify the road surface. Street-level imagery is captured at ground level along streets, by a camera mounted on a vehicle or held by pedestrians. Collecting images like this is more time-consuming than by satellite, but offers a detailed, ground-level view of the landscape. Below are two images of the same road in Kanegem (Belgium), but Figure 1 is captured by satellite and Figure 2 by a phone attached to a bicycle while cycling on the road. With the second image, the details of the stones and the grass by the road are visible, but from the satellite image the stones are not even detectable, indicating that using street-level images is more suitable for the purpose of road surface classification.



*Figure 1 Satellite image showing a road with cobblestones in Kanegem (Belgium)*

*Note.* Adapted from [Satellite image of a road near Neringenstraat, Kanegem], by Google, (n.d.), Google Maps, <https://www.google.be/maps/@51.009524,3.4081662,203m/data=!3m1!1e3?entry=ttu>



*Figure 2 Street-level image taken from a bicycle showing a road with cobblestones in Kanegem (Belgium)*

In 2018, Pereira et al. created a classification for road surfaces using CNN that could classify images taken with a phone from inside a vehicle. They created a model based on the Visual Geometry Group (VGG) architecture and compared the results with a Random Forest (RF) and Support Vector Machine (SVM) algorithm. The model reached an accuracy of 98.2%, which was more accurate than both RF and SVM. However, they only used two categories, paved and unpaved. Balcerek et al. (2020) made a similar classification, but with nine categories (asphalt, concrete, trylinka (a Polish pavement type), concrete paver blocks, granite paver blocks, openwork surface, gravel, sand and grass). Using an architecture based on AlexNet/CaffeNet, and a self-collected dataset of 497 images, they reached an accuracy of 84.5%.

All of the previously mentioned research focuses on road surface classification methods captured for and by cars, which means the roads are mostly paved and the recording device is fairly stable. Therefore, these methods are not directly applicable to a classification for bicycle road surface. Road

surface classification for bicycle paths requires more classes, as the impact of the road on a cyclist's comfort or safety is bigger than on that of a car driver and recreational cyclists seek out the unpaved tracks more often. Also, a bicycle is less stable than a car, possibly resulting in more shaky images. In conclusion, separate classification methods are needed for bicycle road surface classification.

There is considerably less research on this classification task. Verstockt et al. (2013) created a method to classify road points in three paved categories (asphalt, cobblestone and tiles) and three unpaved categories (gravel, grass and mud) based on accelerometer and image data collected on mountain bikes. For this, a Random Forest (RF) algorithm was used and an accuracy of 90% was achieved with the visual-only model, which outperformed the model with the accelerometer. However, their evaluation was performed in a more controlled setting, on only one route. Another study by Hoffmann et al. (2013) also uses accelerometer data to classify road surfaces into the classes smooth, bumpy and rough and at the same time detects potholes and bumps. With a K-Nearest Neighbour algorithm, their best accuracy for the classification task was 77.45%. In more recent years, CNN, namely MobileNet, was used for classifying a dataset of 67 000 augmented street-level images into the classes asphalt track, gravel track, cobblestone track, grass track and forest track. In this project, called SURF, an accuracy of 99.4% was achieved (Beecken & Reinhardt, 2019).

## 2.2 Image processing based on deep learning

### 2.2.1 Computer vision

Computer vision is a field of artificial intelligence (AI) where the goal is to teach a computer to understand and analyse visual inputs (IBM, n.d.). The main tasks in computer vision include image classification, object detection, semantic segmentation, and instance segmentation. **Image classification** involves assigning a label or category to an entire image, enabling a holistic understanding of its content. This results in each image getting one or multiple labels corresponding to what is depicted on the image, for example 'living room', 'meadow' or 'coast'. **Object detection** focuses on identifying specific objects within an image or video and localising them using bounding boxes, providing information about their presence and location. Every object is identified with one bounding box, so with object detection it is possible to count objects, e.g. people in a crowd or trees in a landscape (KeyLabs, 2024). **Semantic segmentation** divides an image or video into meaningful regions, assigning class labels to each pixel, and is particularly valuable in applications like medical image analysis and pothole or crack detection where precise object boundaries are critical (Gu et al., 2018). **Instance segmentation** is a kind of combination of object detection and semantic segmentation, where individual instances are detected along with their boundaries. So if the input picture would be a road with houses and cars, an image classification method could describe it as 'outside', 'city scenery' and 'cars' with varying degrees of certainty, object detection would for example detect all the 'people' and 'cars', with semantic segmentation, the output would split all pixels into classes like 'road', 'car', 'sky' and 'house' and with instance segmentation the cars and houses would be individually detected together with their boundaries.

The computer vision task in this thesis is image classification, as the classes of the total image are important, and not the number of instances or their boundaries.

### 2.2.2 *Image classification methods*

In general, there are two types of image classification methods: supervised classification and unsupervised classification. With supervised classification, a training set with already classified samples is made based on the knowledge of the user and then fed to the model, which learns to classify the samples in the same way and then predict classes for new data. Unsupervised classification is when there is less input of the user, the model tries to group certain images into a new class based on their similarity. The classification task in this thesis calls for supervised classification, because there are certain classes defined that need to be learned by the model.

Some common supervised image classification methods are (Sanghvi, 2020):

- Random Forest Algorithm (RF), which uses combinations of Decision Trees (DT) that have nodes where a binary decision is made, for example 'Does the image have more than X edges?', and combine these to make a prediction.
- K-Nearest Neighbour (KNN), where the distances between different instances in the feature space (e.g. the amount of edges compared to the intensity of the colour red) are compared to find more similar instances.
- Support Vector Machine (SVM), where the feature space is multidimensional, and a set of 'hyperplanes' are created to separate the classes.
- Artificial Neural Networks (ANN), which consist of different nodes in different layers (e.g. an edge detection layer) that are connected to other layers and combine the features of these layers to predict a class.
- Convolutional Neural Networks (CNN), which are advanced neural networks with many specialised layers to classify images.
- Vision Transformers (ViT), which divide an image into a sequence of patches and then use attention mechanisms, which link contextual meaning to more local meaning, to classify them (Dosovitskiy et al., 2020).

In this thesis, Convolutional Neural Networks (CNNs) and Vision Transformers (ViT) will be used as image classification methods, as they are the most advanced and popular techniques for this purpose. A CNN is a "well-known deep learning architecture inspired by the natural visual perception mechanism of the living creatures" (Gu et al., 2018). CNNs consist of layers that analyse patterns (e.g. edges or curves) to distinguish between different classes. Every layer applies filters to the image, by sliding a patch with mathematical operations across every neighbourhood of pixels to create corresponding neurons. This results in each image getting one or multiple labels corresponding to what is depicted on the image.



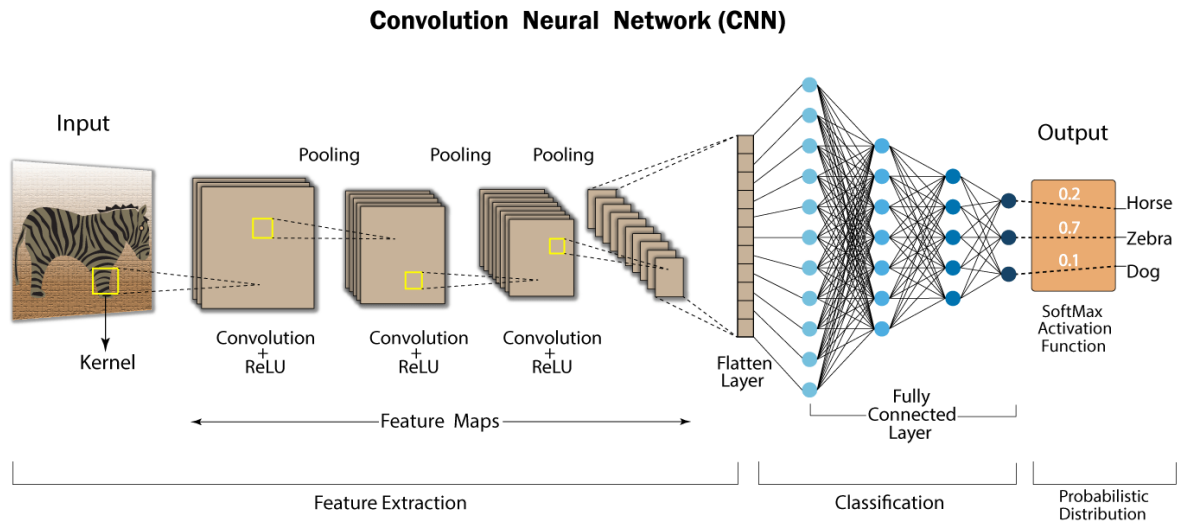


Figure 3 Structure of a classical CNN

Note. Adapted from *Basics of CNN in Deep Learning*, by K. Debasish, (2024) Analytics Vidhya, <https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>

Three types of layers in CNNs can be distinguished (see Figure 3): convolutional layers, pooling layers and fully connected layers. In a convolutional layer, an element-wise filter is applied to find spatial patterns (e.g. Edge-detection), resulting in a feature map. The second type of layer, a pooling layer, is used to decrease the size of the feature map, which is why convolutional layers and pooling layers are often alternated. To reduce the amount of features, a mathematical operation is applied to each region, like the maximum value of that region (max pooling) or the average (average pooling). Pooling layers are necessary to reduce the computing time. A fully connected layer is usually added in the end of the CNN architecture (as in Figure 3), connects all of the neurons of the previous layer with the next layer and adds the weights and biases to predict the output classes (Gurucharan, 2022). Besides these three types of layers, there are also some other important parameters, like the dropout and activation function. The dropout function randomly removes a few neurons in the neural network to prevent **overfitting**, which happens when a model fits the training data very well, but performs less well on the test data, because it has used irrelevant patterns inherent to the training data to classify. The activation function is used to add non-linearity to the network, meaning that it introduces which information is more relevant, and how it should be weighed against other information (Gurucharan, 2022). Some commonly used activation functions are Rectified Linear Unit (ReLU), sigmoid and softmax functions. The ReLU function turns all negative neurons into zero. With sigmoid function, values are transformed to be following a sigmoid curve between 0 and 1. The softmax function is a combination of several sigmoid functions, but can be used for more than two classes (Sharma et al., 2020).

The process of training a model starts with the model architecture, consisting of the previously mentioned layers. Each layer in a neural network consists of interconnected neurons, that receive inputs, perform computations and pass the results on to the neurons in the next layer. The connections between the neurons have weights assigned to them, indicating their importance to the prediction. Another



parameter in neural networks are biases, constant values added to the output of the neurons to shift the output up or down. The most fitting parameter values for every specific task have to be learned by the model during training. In neural networks, this happens in a cycle consisting of two parts: a forward pass, where a batch of samples is processed and the predictions are compared to the actual labels to compute the loss; and a backward pass (backpropagation) where the loss is reduced by adjusting the weights of the layers. The adjustments in parameters are calculated by gradient descent, a mathematical technique to minimise the loss. The gradient descent is 'packaged' into an optimiser, of which one of the most popular ones is the Adam optimiser. The forward pass-backward pass cycle is iterated over every batch of data, until the entire training dataset is finished. This process, in turn, is repeated for a certain amount of epochs: iterations of the training process.

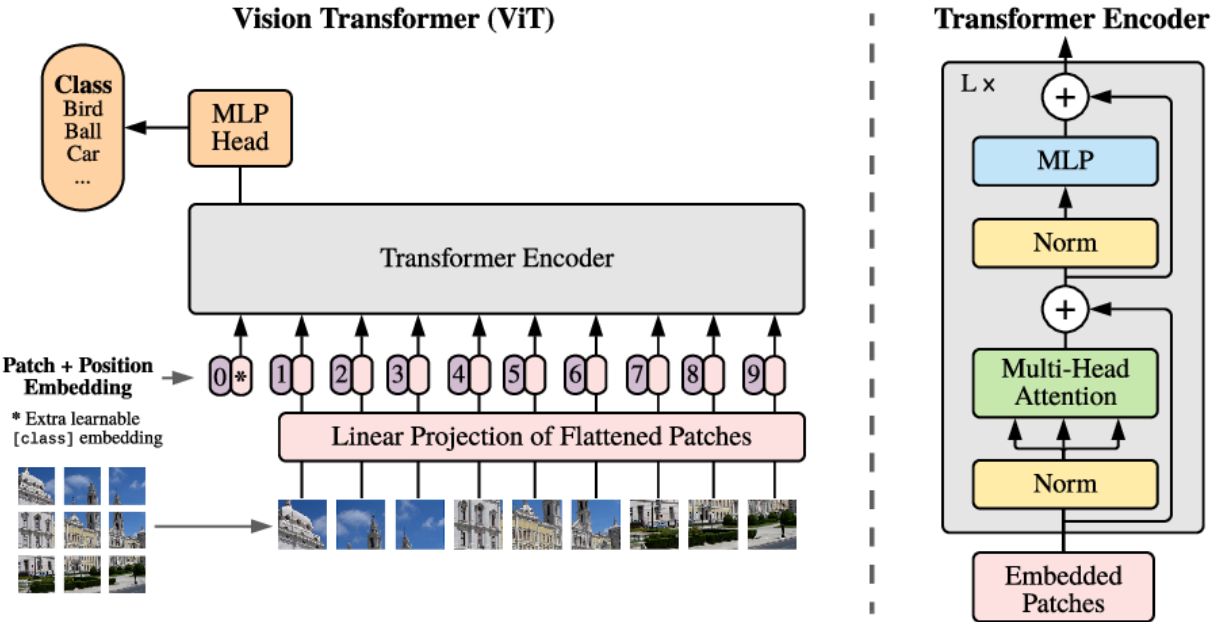


Figure 4 Overview of the Vision Transformer model

Note. Adapted from "An image is Worth 16x16 Words: Transformers for Image Recognition at Scale," by Dosovitskiy et al., (2020), *International Conference on Learning Representations*, <https://arxiv.org/abs/2010.11929>

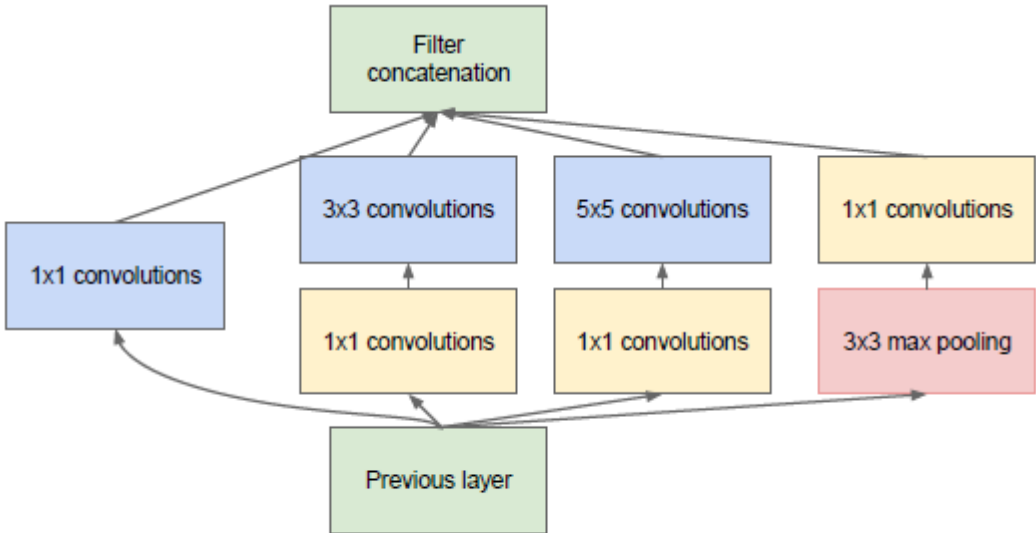
Next to CNN model architectures, there is also the approach of Vision Transformers (ViT), which do not process images pixel-wise like CNN do. Before the ViT, Transformers were almost exclusively used for Natural Language Processing (NLP) tasks, but now ViT can perform very well with both text and images. As introduced by Vaswani et al. (2017), a transformer uses attention mechanisms, which link contextual meaning to more local meaning. It works by using a combination of encoders and decoders, which (in a broad sense) can be compared to the convolutional layer and pooling layer in CNNs. For example, in 'The cat drinks milk, it's sour' and 'The cat drinks milk, it's making a mess.' the word 'it' refers to the milk or the cat depending on the context. Transformers can detect this and group related words or pixels together in encoder blocks. Other well-known transformers are Google's Bidirectional Encoder Representations from Transformers (BERT) and Open AI's Generative Pre-trained Transformer (GPT) (Püttmann, 2023). To process images, ViT breaks down images into patches and uses the attention mechanisms that are specific to transformers to capture both local and global

information (see Figure 4). For example, when trying to identify a cat or a dog, they combine both information on the facial features (local) as on the shape of the animal itself (global). As a result, ViT demonstrates competitive performance across datasets of varying sizes.

By using CNNs and ViTs to classify images, less obvious patterns can be discovered than with traditional image classification algorithms, which can lead to a higher accuracy. It is however, not exactly clear how the classification happens: there are no clear decision branches as with a decision tree, it is a black box system. Furthermore, as the model gets more complex, and therefore often more accurate, it needs substantially more training data (Gu et al., 2018). A deeper network also requires more computational power and related to that, a longer processing time.

2.2.3 State-of-the-art models and their architectures

While all CNN models share the same fundamental building blocks, various architectures introduce unique combinations of layers, balancing computational efficiency and performance. The following sections provide a brief chronological overview of state-of-the-art models and their architectures. Each of these models introduces novel contributions to the field and is accessible through different deep learning frameworks, making them prevalent in both research and practical applications. This variety allows for an evaluation of which architecture is most suitable for the classification task at hand.



(b) Inception module with dimensionality reduction

Figure 5 Inception module with dimensionality reduction within GoogLeNet

Note. Adapted from "Going deeper with convolutions," by Szegedy et al., (2014), *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern recognition*, 7(12), p. 1-9, <https://doi.org/10.1109/CVPR.2015.7298594>

In 2014, Szegedy et al. introduced a CNN building block called an inception module (figure 5), which uses convolutional layers of increasing kernel sizes (1x1, 3x3, 5x5) to simultaneously look at different

scales in the image. Because this results in a lot of connections, they also added a dimensionality reduction. Their resulting new CNN model was **GoogLeNet**, which has multiple inception modules in their 22 layers with parameters. The last pooling layer in the model is Global Average Pooling layer, which takes the average of the 7x7 feature map to reduce it to 1x1. The researchers won the ILSVRC 2014 competition, an image classification competition where the goal is to create the best possible classification algorithm to classify a certain image dataset (Tsang, 2018a).

**VGGNet** gets its name from the creators of the model: the Visual Geometry Group (VGG) from the University of Oxford. It was the first runner-up of the ILSVRC 2014, after GoogLeNet, but it won the localisation task of the competition. It introduced a new approach where multiple smaller filters (5 layers of 3x3 filters) were combined instead of using one big layer (of 7x7 or 11x11), which was the popular choice at the time. This results in less parameters to be learned and therefore it learns faster and is less prone to overfitting (Simonyan & Zisserman, 2014). The architecture introduced by VGGNet has inspired a lot of other model architectures. Its simplicity facilitates easier interpretation of the network's behaviour and allows for straightforward transfer learning and hyperparameter-tuning, making it a popular choice in both research and practical applications.

However, the architecture of VGGNet and other similar networks creates the vanishing/exploding gradients problem: during backpropagation, to adjust the weights, a lot of very small or very large gradients have to be multiplied, resulting in zero (vanishing) or a very large number (exploding). This has a negative influence on the performance of the model (Tsang, 2018b). By creating parameter-less shortcuts (also called identity mapping) He et al. (2015) avoid this negative influence in their model architecture **ResNet**. Adding these shortcuts to the combinations of 3x3 layers adds a lot of complexity and training time (see Figure 6 left), so the architecture was adjusted to bottlenecks of 1x1, 3x3 and 1x1 layers consecutively (see Figure 6 right). ResNet50, which has 50 layers with these bottlenecks, won the ILSVRC 2015.

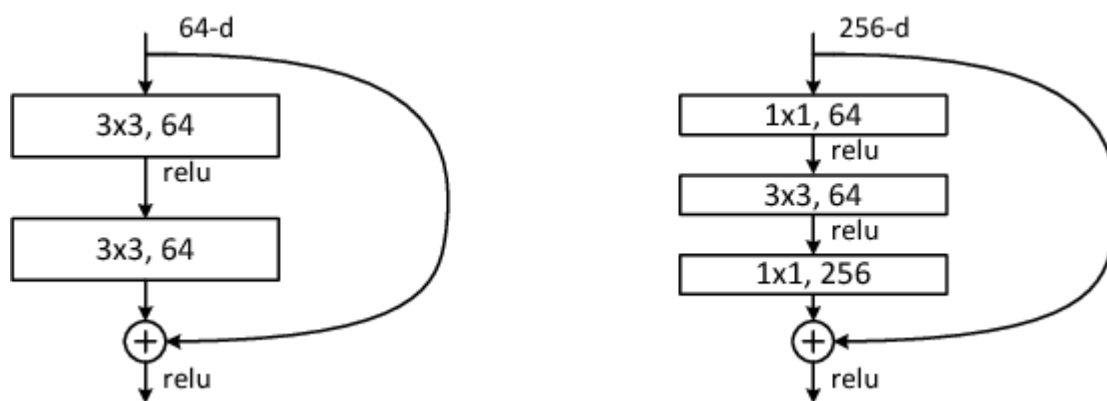


Figure 6 Left: a standard building block, right: a 'bottleneck' building block for ResNet-50/101/152.

Note. Adapted from "Deep Residual Learning for Image Recognition," by He et al., (2015), *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern recognition*, p. 770-778, <https://doi.org/10.1109/CVPR.2016.90>

Another problem in CNN architectures is that information passes through so many layers, that their informational value is lost throughout, like a T-shirt that has been washed so many times that the colours and text fade. To solve this problem, the **DenseNet** architecture passes all information from all layers to all layers after that. The layers are combined in Dense Blocks and followed by a bottleneck layer to reduce the complexity. Because of the density of the network, less layers (and therefore less parameters) are needed: only four Dense Blocks follow on each other (Huang et al., 2016).

A more recent architecture is **MobileNet V1**, a CNN that is designed to work on mobile devices (Howard et al., 2017). To achieve this, depthwise convolution is followed by pointwise convolution to decrease the computation. Depthwise convolution means that a separate operation is performed on every channel of the image (e.g. red, green or blue) and the pointwise convolution combines the information from all channels together by taking a 1x1 filter. Some multipliers are also introduced: multiplier alpha controls the amount of channels that are being used, and multiplier rho controls the image resolution. This way, the speed of the model can be changed. A year later, a second version of MobileNet was introduced by Sandler et al. (2018), introducing combinations of inverted residuals (shortcuts, similar to ResNet) and bottlenecks to improve the performance.

**EfficientNet** (Tan & Le, 2019) also tries to improve the learning speed by scaling its width, depth and resolution. Scaling refers to the process of increasing the size or depth of a neural network to improve its performance. By adding more layers (depth scaling) or more neurons (width scaling) or increasing the resolution of the images (resolution scaling) simultaneously, EfficientNet improves the performance while minimising the computation cost. In 2021, Tan & Le, created some new versions of their EfficientNet architecture, that had similar results but were much smaller and faster than their original architecture. To achieve this, they added Stochastic Depth, which randomly skips layers during training to help regularise the network, and improved the blocks so they have similar results but with less parameters.

Next to these CNN models, **Vision Transformers** are often used to execute image classification. Their architecture (as explained in the previous section) uses attention-based blocks to combine local and global information for more accurate image classification.

## 2.3 Transfer learning and hyperparameter tuning

The goal of training any Deep Learning algorithm is to find the optimal **parameters** of the model to minimise the cost function. These parameters include the weights, coefficients that connect the neurons to calculate the output of the model, and biases, which are added to the weights to allow the model to fit the data better by providing an additional degree of freedom. The **cost function** (or loss function) is the metric that is used to determine how well the model is performing on the given dataset: it measures the discrepancy between the predicted outcomes and the true outcomes. Multiple loss functions exist, like Mean Squared Error (MSE), which is commonly used in regression tasks to penalise the difference

between predicted and actual values squared; Huber loss, a robust alternative to MSE that is less sensitive to outliers; and Kullback-Leibler Divergence, a measure of how one probability distribution diverges from a second, often used in probabilistic models. In this thesis, cross entropy loss is used as a loss function due to its effectiveness in classification tasks. Cross-entropy loss measures how different the predicted probability distribution is from the true probability distribution. It calculates the average log-likelihood of the true labels under the predicted distribution (Goodfellow et al., 2016).

For minimising the cost function, a mix of different factors are at play. The dataset itself is one of the most critical elements - adding more high-quality training data is often the most effective way to reduce the cost function and improve model performance. If the dataset is limited, another powerful technique to use is **Transfer Learning**. Transfer Learning involves using information from one machine learning task to better execute another task, because it is assumed that many of the factors that explain the variations in the first task are relevant to the variations that are needed for the second task (Goodfellow et al., 2016). This approach offers the advantage that learning is faster and results in a better performance. In practice, this means using models that have been pretrained on big datasets, which are available in machine learning libraries. The library that will be used is PyTorch, a widely used open-source machine learning library originally developed by Meta AI and now part of the Linux Foundation (*PyTorch*, n.d.).

Of course, the model architecture also plays a key role in determining the cost function. Certain neural network designs are better suited for specific types of data and tasks. However, the way these models learn from data is determined by their hyperparameters. **Hyperparameters** are high-level settings that cannot be directly learned from the data, but must be manually tuned by the machine learning engineer (Shahul & Aayush, 2023). These include choices like whether to use early stopping to prevent overfitting or not, but also which batch size and optimisation algorithm to use. The optimal hyperparameters depend on the model type and dataset. A brief summary of relevant hyperparameters for CNNs is given below.

A first hyperparameter is the **amount of epochs** (or training iterations) used to train the data. For how long does the program try to improve the classification? After every epoch the entire training dataset has been passed through the model. When too little epochs are used, the model did not have enough time to perfect the parameters, but using too much epochs can lead to overfitting. It is therefore important to find an ideal amount of epochs to train the model for.

The second important hyperparameter is the **batch size**, which refers to the number of samples that are processed before updating the model parameters (Zvornicanin, 2023). A first way to do this is using batch gradient descent, where all of the samples are taken into account during the epoch: the model looks at all images at once to update the parameters and essentially learn how they should classify. In this case the batch size would be equal to the size of the whole dataset. A second option is using stochastic batch gradient descent, viewing only one random sample at a time to learn, thus using a

batch size of one. Lastly, there is mini-batch gradient descent, where a predefined amount of samples is used during every epoch. This approach is commonly used, and the batch size is often in powers of two, like 16 or 32 (Zvornicanin, 2023). This way, the model parameters are adjusted many times during each epoch, but not for every sample, resulting in a more efficient learning process.

The **learning rate** determines the speed at which the model learns, how long earlier knowledge is kept and how quickly the weights are updated: a larger learning rate results in a faster model, but might overlook the optimal solution. It could be compared to following a road by car or by foot: a car goes faster (bigger distances) and allows you to see more quicker, but you might miss the optimal view that you would have seen if you went by foot. The learning rate usually varies between 0.000001 and 1. According to Goodfellow et al. (2016), the learning rate is the most important hyperparameter to tune.

The way learning rates are handled depends on the **optimiser** that is used. Optimisers are algorithms used in machine learning to adjust the parameters of a model during training to minimise the loss function and improve the model's performance. They calculate and combine **gradients** to know in which direction they should go to reduce the loss function. A first possible way of combining gradients in optimisers is taking their average, which is called momentum. This results in an acceleration of the training process and an improvement of the performance. A second technique, Adagrad (Adaptive Gradient), adapts the learning rate for each parameter based on the sum of the squares of the gradients for that parameter. This way, parameters with large gradients will be given a larger learning rate, and for frequently occurring parameters a quickly decreasing learning rate is avoided. It is effective for sparse data but can lead to a rapid decrease in the learning rate. Another technique is RMSProp, which is similar to Adagrad but uses a moving average of the squared gradients instead of the sum. This also helps prevent the learning rate from decreasing too quickly. Adam (Adaptive Moment Estimation) combines the ideas of momentum and RMSProp. It maintains an exponentially decaying average of past gradients (like momentum) and an exponentially decaying average of the squares of past gradients (like RMSProp). Adam is considered the most popular optimiser due to its efficiency, ease of implementation, and good generalisation ability. Therefore, this is the optimiser that will be used here.

One key aspect of optimisers is the concept of **weight decay**, which is a way to prevent overfitting (also called regularisation) by adding a penalty term to the loss function proportional to the square of the weights. The effect of this is that the network prefers to learn small weights and only adds large weights if they considerably improve the cost function. After all, larger weights can make the model's predictions highly sensitive to small changes in the input data, leading to unpredictable behaviour on unseen data. The importance of the size of the weights versus the minimisation of the cost is determined by the value of the weight decay: a small value means minimising the cost function is preferred while a large value minimises small weights (Nielsen, 2019).

Another way of preventing overfitting is using early stopping and dropout. In early stopping, the model has to classify the validation data during the training process as well, and if the validation loss is

increasing too much, the training is stopped. This way, the model will perform better on real examples, not just the training set. How many epochs the validation loss can increase before early stopping is carried out, is determined by the early stopping **patience**. This value is often (as a rule of thumb) set to 10 to 20% of the amount of epochs. Dropout regularisation removes a random selection of a fixed number of the units in a network layer for a single gradient step. This means that during forward propagation, some of the neurons are removed from the process and in backward propagation the weights are calculated. Then, the same technique is applied with another batch and other neurons that are removed. By repeating this process, the possibilities of overfitting are reduced, as every batch there will be a 'different network' classifying the images (Nielsen, 2019).

There are endless possibilities for choosing hyperparameters, so how to adjust them efficiently? The first possible approach is to do grid search, combining all possible values for all hyperparameters and then evaluating them. This is possible for a few combinations (e.g. three different learning rates, four batch sizes and three options for numbers of epochs makes 36 different combinations), but the amount of combinations grows quickly. The second option is random search: making random combinations of options to limit the amount of searches and still have a good solution. Lastly, there is **Bayesian optimisation**, where the chance of a combination improving the performance is calculated based on previous searches and then used to choose new points to evaluate (Gupta, 2020). Another tactic to use is **pruning**: the elimination of less promising combinations of parameters, like cutting some branches of a tomato plant to make the plant focus on growing fruits instead of leaves.

Hyperparameter optimisation can be done manually or (in this case) automatically, and there are a lot of different frameworks to work with, like Spearmint, Scikit-Optimise or Optuna (Shahul & Aayush, 2023). For the purpose of this thesis, **Optuna** will be used, because it is a flexible framework that fits with nearly all machine learning frameworks (among which PyTorch) and it requires little intervention or extra coding. It uses Bayesian optimisation as its default optimisation algorithm (Akiba et al., 2019).

## 2.4 Research gaps

Considering the smaller amount of research on classification methods for cycling road surfaces, an application and comparison of methods as provided in this thesis is needed. The characteristics of the cycling road surfaces are very variable, as local governments might change it to a different type (e.g. to asphalt for higher comfort or to an unpaved type for more water filtration) or heavy floods or erosion may spread over the surface. So, road surface characteristics are often not documented well, partly because these roads can be more separated from the inhabited centres; they are places to pass and not to reach. Therefore, it is needed to develop a system that deducts information from travelling along the road (like street-level images) and turn this information into road surface types. Existing methods have only tried a limited number of classification methods, so a comparison of different CNN models and a more recent Vision Transformer can also shed some light on the further possibilities in this research area.

Using street-level images might have some limitations. First of all, collecting such type of images is time intensive, volunteers have to go cycling onto every road themselves, so a lot of cycling trips and volunteers are needed for a big dataset. In this thesis, the size of the dataset could therefore be a hindrance to the model performance. Secondly, collecting data from a bicycle introduces shakiness and may cause very blurry images, which could also lead to bad performance. However, shakiness can also be used to identify road types, as a proxy to oscillation: more bumpy road surfaces have more blurry images. Lastly, there are no standardised road surface categories: all previously mentioned papers used different classes. In this thesis yet another distinction of road surface categories will be introduced, based on some of the categories of OpenStreetMap (OpenStreetMap Contributors, n.d.). Only six out of the 40 possible standard values for the surface tag in OSM are chosen (see Section 3.2), a choice that is well-motivated but always arguable, due to the overlapping nature of road surfaces.

## 3 METHOD

### 3.1 Overview of methodology

The proposed method (as summarised in Figure 7) first discusses the six different road surface classes that will be used. It will be stated how exactly these surface types are interpreted and distinguished from each other and arguments will be given to support this choice. Section 3.3 then describes the data collection, annotation and splitting. First, the process of gathering, annotating and preprocessing al data is discussed, followed by how the dataset is split into training, testing and validation sets. In section 3.4, every step of the comparison is outlined. First it will be stated which models (and their versions) will be used and why. Then it is important to explain what will be compared: the two different splitting approaches, the influence of hyperparameter tuning and the overall model architecture. To compare these aspects, different evaluation metrics will be calculated and visualised through confusion matrices, learning curves and saliency maps, as discussed in section 3.4.3. Additionally, a small experiment is carried out to analyse the influence of the image capturing method on the performance of the model and the shakiness of the images.

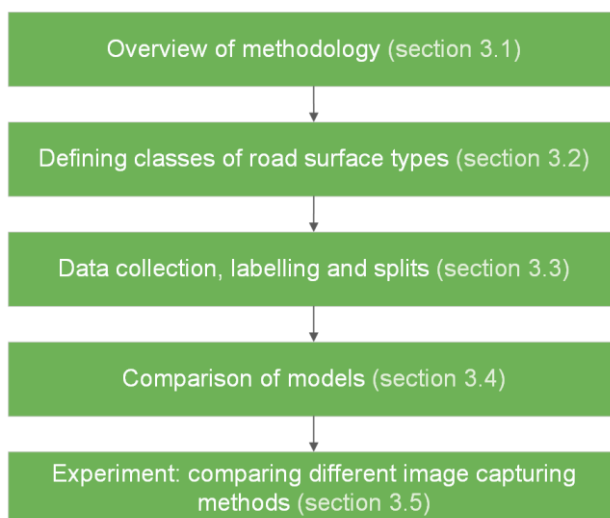


Figure 7 Overview of the different parts in the method



## 3.2 Defining classes of road surface types

Having relevant, exclusive and somewhat standardised classes is essential for having a relevant classification. In existing literature, a range of different classes have been used: Verstockt et al. (2013) created a method to classify road points in three paved categories (asphalt, cobblestone and tiles) and three unpaved categories (gravel, grass and mud), Balcerek et al. (2020) used nine categories (asphalt, concrete, trylinka (a Polish pavement type), concrete paver blocks, granite paver blocks, openwork surface, gravel, sand and grass). Most surface classifications that are car-oriented only use two categories: paved and unpaved (Pereira et al., 2018). For cyclists, just distinguishing paved and unpaved roads is not sufficient, as riding over cobblestones versus asphalt or fine gravel versus muddy ground is a very different experience. Thus, similar to Verstockt et al. (2013), six surface categories are chosen from the OpenStreetMap (OSM) surface categories. This is seen as the closest ‘standardised’ classification available.

OpenStreetMap has more than 40 possible standard values for the surface key, ranging from general terms like ‘paved’ to very specific ones like ‘woodchips’ (OpenStreetMap Contributors, n.d.), but next to these, OSM contributors can also create their own classes. Some classes are subclasses of other classes, such as ‘grass’, ‘clay’, ‘sand’, ‘earth’, ‘gravel’ or ‘pebblestone’ being more specified instances of the class ‘ground’. We want to have mostly general classes that are the most prevalent ones. Table 1 shows the classes and their abundance in OpenStreetMap ways if bigger than 0.5% (OpenStreetMap Contributors, n.d.).

*Table 1 Openstreetmap (OSM) Classes and their abundance*

<b>Class (OSM: Value of Key:Surface)</b>	<b>Abundance in ways (March 2024)</b>
<b>Asphalt</b>	44.59%
Unpaved	18.79%
Paved	6.8%
<b>Concrete</b>	5.42%
<b>Paving stones</b>	5.35%
<b>Ground</b>	5.3%
Gravel	3.44%
Dirt	2.52%
Grass	1.9%
Compacted	1.66%
Sand	0.85%
<b>Sett</b>	0.7%
<b>Fine gravel</b>	0.64%

*Note.* Adapted from *Key:surface*, by OpenStreetMap Contributors, (n.d.), OpenStreetMap Wiki, <https://wiki.openstreetmap.org/wiki/Key:surface>




Classes 'unpaved' and 'paved', are very frequently used in OpenStreetMap data, but they will not be used because they do not give a lot of information about the road surface: a paved road could be very difficult terrain (e.g. cobblestones) and an unpaved road could be easily cyclable (e.g. fine gravel cycle highways) and vice versa. The top most abundant class is '**asphalt**', with 44.59% of the ways in OSM having this value. In other research, it is also always included in the list of classes (or seen as part of the class 'paved'). This indispensable class is therefore also included in this research. The next most common class (next to 'paved' and 'unpaved') is '**concrete**', which differs from asphalt because it is cast in place in a series of plates. These plate transitions can cause discomfort for cyclists and require a different type of maintenance than asphalt, making it important to make a distinction between the two classes. '**Paving stones**' is almost equally as abundant as 'concrete', and is also mostly a smooth road type to cycle on. It looks very different from 'asphalt' and 'concrete' and requires a different maintenance, so a distinction is also needed here. Other researchers also use similar classes for this, like 'tiles' (Verstockt et al., 2013) or 'paver blocks' and 'trylinka' (Balcerek et al., 2020), but 'paving stones' is seen as a overarching class, as these types of pavements can have a lot of different shapes, colours or sizes depending on the city or country. A last paved road surface type is '**sett**' stones, which are rectangular, hewn stones (Nederlands: 'kasseien') that are mostly used for old roads. They only refer to 0.7% of the roads in OpenStreetMap worldwide, but this percentage is likely higher in certain areas (such as Belgium). However, including this class in the classification is needed because these roads have a large impact on the cycling comfort and safety. There is some confusion in the OSM community about whether the word 'cobblestones' or 'sett' should be used, and still volunteers use the two classes interchangeably (henke54, 2018). However, 'cobblestones' refers to spherical, rounded, unhewn rocks that are used for pavement, and 'sett' stones are more rectangular, hewn rocks (OpenStreetMap Contributors, n.d.). Sett stones appear in different colours and sizes (depending on the location) and can also be lain in different patterns (e.g. arches or rows) but they are all irregular and therefore have more space between them than paving stones.


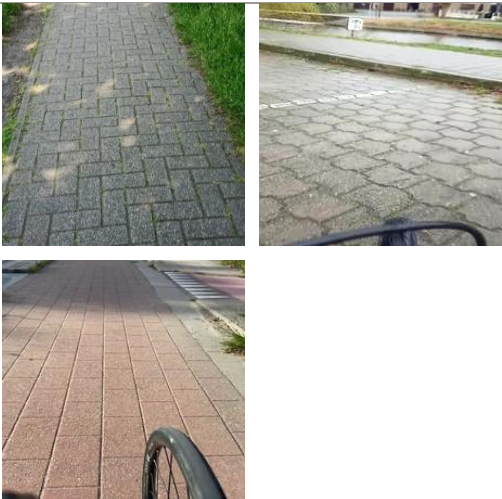

Next to these paved road types, there are of course a lot of unpaved ones. Most of the time they are tagged as 'unpaved', but a more specific type is desired, as the cyclist's experience on these paths will differ based on what that unpaved surface consists of (e.g. grass is more difficult to cycle on than a wide gravel path). The second most common unpaved surface type is '**ground**', which is an overarching class for 'gravel', 'dirt', 'grass', 'compacted', 'sand' and 'fine gravel'. This class is, in this thesis, interpreted as being local soil in various conditions: it could be overgrown with long grass or mown, or the local soil could be exposed and make it muddy or sandy. This combination is used because there is a large variety of soils and stages of maintenance, and the appearance of the road could change any day due to the weather and usage. Finally, one more class, '**fine gravel**' will be used in this research. In the OSM community, there is some confusion about the difference between 'fine gravel' and 'compacted', as it can be hard to distinguish the two (TwoWheelsGood, 2018). Both type of roads consist of small stones, but the first one is loose and the second is pressed down or bound with water. In practice, they are often used interchangeably. It must also be noted that 'gravel', the most abundant 'ground' subclass, and 'fine gravel' are also often confused with each other, as most people use the word 'gravel' regardless of how

large the rocks are. For the purpose of this thesis, 'fine gravel' will be used, because it's difficult to know if the road is compacted or not based on an image, but it is possible to see if it consists of small stones. This particular class is also relevant to cyclists, as some city bikes, gravel bikes and even race bikes might be suited for riding on fine gravel, but not on the rest of the 'ground' trails.

All used classes can be found in Table 2, along with a description and some example images. The definitions are based on those of OpenStreetMap (OpenStreetMap Contributors, n.d.), but are adjusted for the classification task at hand, and a visual description is added based on what features should be looked at to classify images into these classes.

Table 2 Chosen surface types with description and examples

Surface type	Description	Examples
<b>Asphalt</b>	Smooth pavement made out of mineral aggregate bound by asphalt.  <b>Visual:</b> Dark grey, reflective, small rocks visible, irregular cracks, potholes and sometimes fixed with other asphalt	
<b>Concrete</b>	Light cement concrete, often in blocks or lanes.  <b>Visual:</b> Light grey (in sun), has straight edges of blocks, some small rocks visible.	
<b>Fine gravel</b>	Loose (or pressed down) small rocks which are usually put in place.  <b>Visual:</b> Small rock pieces in a regular, wide path. No/little deep traces of bicycles visible.	

<p><b>Ground</b></p>	<p>Exposed local soil, like sand or earth, that can be covered in mud, grass, fallen leaves or other natural deposition.</p> <p><b>Visual:</b> Sand, grass or mud that shows or is a narrow path. Sometimes no clear path is visible.</p>	
<p><b>Paving stones</b></p>	<p>Fitted, geometric and smooth stone pavement, like most sidewalks in Belgium. The stones have been baked and are therefore almost identical.</p> <p><b>Visual:</b> Artificial stones with smooth (straight/curved) edges fitted together in a tight pattern.</p>	
<p><b>Sett</b></p>	<p>Natural stone pavement with an irregular edge. The stones have been hewn.</p> <p><b>Visual:</b> The stones have irregular edges, less tightly fitted than paving stones. The stones are cut and do not cover the surface completely. Images are also often shaky.</p>	

Note. Adapted from *Key:surface*, by OpenStreetMap Contributors, (n.d.), OpenStreetMap Wiki, <https://wiki.openstreetmap.org/wiki/Key:surface>

## 3.3 Data collection, labelling and splits for model building

### 3.3.1 Data collection

To create a reproducible, affordable and accessible method that can be expanded to a larger group of people, data will be collected manually by smartphones and action cameras by four volunteers during cycling trips. The device will be mounted on a bike mount and pivoted to record the road in front of the bike, resulting in a video for every cycling trip, and therefore a collection of images. The images are purposefully taken in different circumstances: on wet and dry roads (which discolour the roads and create a reflection) and with shadows (of trees for example) or bright light. Different devices are also used to collect the images: Android phones, iPhones and a GoPro; and they are mounted on different bikes and under different angles. As a result, on some images the horizon is visible, on others only the road right in front of the wheel and on others even the handlebars of the bike. All these variations are introduced to mimic a real life situation where the classification would be used by volunteers to classify their cycling trip by using what devices and bikes they have on hand to collect the data. Introducing variety makes the model more robust.

Besides the four volunteers who went cycling for this thesis specifically, YouTube content creator Manou Maudgal was contacted to provide additional data for the 'ground' surface category. Since his videos mostly contained shots taken with a GoPro from an optimal angle (which showed the road before the bike), it was a most welcome addition to the dataset. This could also show the potential of using social media content for creating datasets like this. The videos are sent to the author by WeTransfer links, because of their size. Preprocessing of the videos themselves is often needed, because the Apple .mov video standard had to be converted to .mp4, and sometimes it was more efficient to crop some parts of the video out, like parts of the YouTube videos where a person is just talking. When this is completed, the video title is changed in the format of EM01\_2023-06-13\_15-30-00, with 'EM' being the contributor code (Emma), 01 being the video number, 2023-06-13 being the recording date and 15-30-00 being the (estimated) recording start time. By using this format, it is easier to have an overview which videos are already processed and who has contributed to which category. Then, the video frames are extracted by splitting the video at an (arbitrarily chosen) interval of 5 seconds. Before the images are organised, they are resized into a 224x224 format, as this format is necessary for most models. Some images are then discarded because they don't contain the road (e.g. a picture of the sky) or are duplicates (e.g. when waiting for a traffic light). Finally, they are manually put into folders with corresponding surface classes by looking at the images themselves.

There is no one-size-fits-all answer to the question of how many images a good dataset should have. It depends on the model architecture (and specifically how many layers it has) and the amount of classes that are used. A deeper model architecture requires more data. The deepest one used in this paper is (by far) the VisionTransformer, and according to Martinez-Noriega & Yokota (2023) they typically require over 100 million images. Tensorflows image dataset CIFAR10 contains 60 000 images divided into ten classes (*Convolutional Neural Network (CNN)*, 2024), so extrapolating this to a dataset with six classes

would mean that around 30 000-50 000 images would be needed. Goodfellow et al., (2016) mention in their Deep Learning book introduction that a rough rule of thumb is to have 5000 images per class to achieve an acceptable performance with the models of that time. However, the models are in this case not trained from scratch, but Transfer Learning is used. In an experiment by Zanotti (2020) three sizes of the Category Flower Dataset (Nilsback & Zisserman, 2008), which has 102 classes of flowers, were used to train and finetune a pretrained dataset. With the smallest training dataset of 818 images, a F1-score of 79% was achieved, with 3275 images 91% and with the largest set of 6551 images 94%. This indicates that even a smaller dataset of less than 5000 images in total can be used to classify a number of classes.

With the available resources and time, the goal was to collect at least 200 images for each surface class, resulting in a balanced dataset of more than 1200 images in total. It is expected that there will be more images for the 'asphalt' class, as this is also the most prevalent class in practice (see Table 1 in the previous section). The collected dataset will be augmented by using Pytorch Transformers, a library to flip, rotate and crop the images. This way, the actual images that will be trained upon will be an augmented version of the original ones.

### *3.3.2 Data split ratio and sampling*

In order to effectively train the model, adjust its parameters and evaluate its performance on unseen data, it is necessary to split the total dataset into separate subsets: a training set, a validation set and a test set. Various methods exist to create these subsets, including random splitting, K-fold cross-validation, and manual splitting. K-fold cross-validation involves splitting the data into k subsets (folds) and using each fold as a testing set while the rest are used for training. This process is repeated k times, with each fold used once as a validation while the k-1 remaining folds form the training set. This approach is not used, as it complicates the comparison of hyperparameters and model performance across different folds. For comparability, a fixed training set, validation set and test set will be created: every image will only be part of one subset. When splitting, the ratio of training set to validation and test set is influenced by several factors inherent to the modelling process. Having numerous hyperparameters to tune necessitates a larger validation set, a large enough test set is necessary to have a correct evaluation and at the same time there has to be enough training data left, otherwise the training will have a high variance in performance (Baheti, 2021). Given the relatively modest size of the dataset in this study, keeping a sufficiently large training dataset will be prioritised. As such, the chosen split ratio allocates 70% of the data to the training set, 20% to the validation set and 10% to the test set. Allocating the images to the subsets could be done by random splitting, dividing the images into the subsets randomly, but for simplicity a manual splitting approach is used.



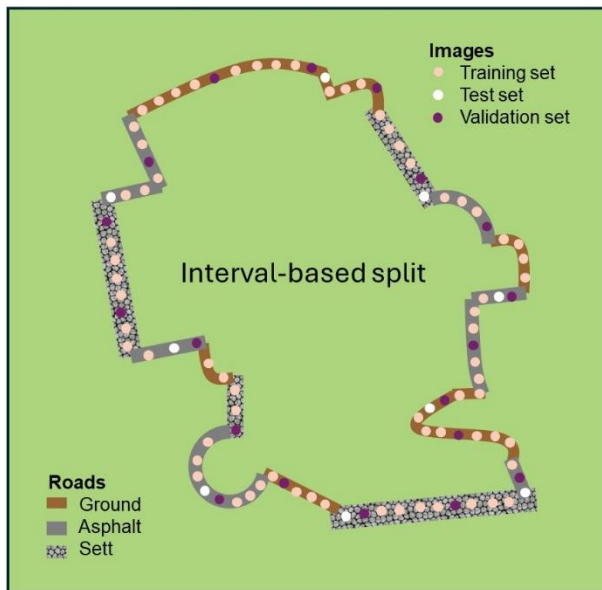


Figure 8 Simplified example of the interval-based splitting approach

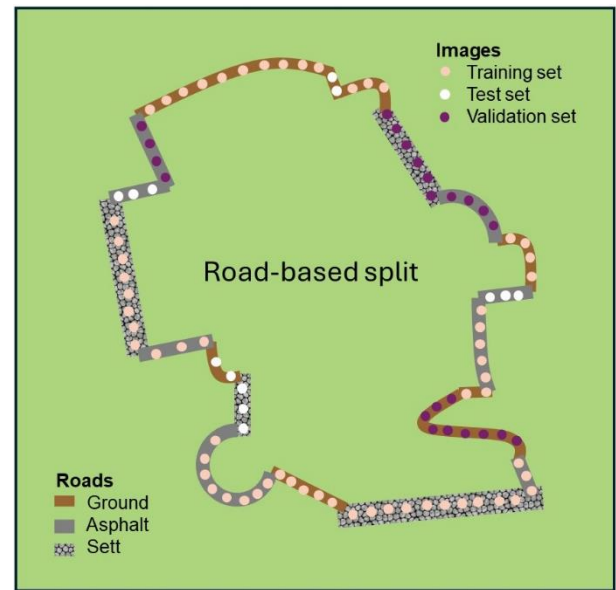


Figure 9 Simplified example of the road-based splitting approach

This raises the question: 'Which images will be put in which subset?'. In the initial formulation of the dataset, referred to as the **'interval-based' dataset**, a sequential splitting approach was employed for the construction of the test and validation sets. This method involved selecting every 5th and 10th image within each class, ensuring a structured and diverse representation of the dataset (see Figure 8). However, upon further examination, it became apparent that this approach led to the inclusion of highly similar images across the training, test, and validation sets. After all, having more than five images per road was not uncommon, as can be seen in the simplified example. This could potentially result in overfitting and inflated accuracy metrics. To mitigate this issue and enhance the generalisability of the model, a revised dataset, termed the **'road-based' dataset**, was introduced (see Figure 9). With this splitting approach, entire roads comprising approximately 2 to 20 similar images were grouped together and put into the respective subsets. By adopting this approach, it was ensured that no identical or highly similar images were present across different subsets, thereby promoting a more robust evaluation of model performance and enhancing its ability to generalise to unseen data. At the same time, it was made sure that the distributions between subsets (70% training data, 20% validation and 10% testing) remained the same.

Both the interval-based dataset and the road-based dataset will be used to train the models, to be able to compare their different splitting methods. As the road-based training set is less diverse, it is expected that models trained on this set will have more difficulty finding optimal parameters, and the accuracy will be lower.

## 3.4 Comparison of methods

### 3.4.1 Models used in this thesis

In this thesis the following model versions are used and compared:

- **GoogLeNet** (Inception V1)
  - GoogLeNet introduces the idea of an inception module, which allows for more efficient computation by combining multiple convolutional filter sizes into an inception module. GoogLeNet (Inception V1) is chosen because it was the first to introduce this concept.
- **VGG16**
  - This is a very straightforward model and therefore a popular choice in both research and practical applications. Its simplicity and depth have made it a benchmark model for many tasks.
- **ResNet101**
  - ResNet introduces the novel idea of bottlenecks to solve the vanishing/exploding gradient problem. ResNet101 is selected because it provides a good compromise between depth and computational requirements, as it has significantly better performance than shallower ResNet versions without being as resource-intensive as ResNet152.
- **DenseNet201**
  - DenseNet introduces dense connections between layers, ensuring maximum information flow between layers. This helps in solving the vanishing gradient problem and improves feature reuse, making the model highly efficient and accurate. DenseNet201 is chosen for its balance between depth and efficiency, providing robust performance without the excessive computational requirements of even deeper DenseNet variants.
- **MobileNet V2**
  - This CNN is lightweight and can therefore be used on mobile devices. Additionally, MobileNet V1 has been used in related research by ... and can therefore be compared with this research. The newer version, MobileNet V2, also introduces inverted residuals and linear bottlenecks, which improve performance and efficiency.
- **EfficientNet V2 M**
  - EfficientNet is one of the most recent high-performing CNNs. Out of the three sizes of EfficientNetV2 (S, M and L), EfficientNetV2\_L is the largest one, and achieves the highest accuracy, but it uses too much memory and can't be run with the available resources, but the medium version can. EfficientNet V2 incorporates a compound scaling method that balances network depth, width, and resolution, offering state-of-the-art performance with optimised resource usage.
- **ViT-B/32**
  - The Vision Transformer is a very recent image classification model with a different architecture than CNNs, and therefore interesting to compare. ViT-B/32 uses the Base



(B) model architecture as opposed to the Large (L) or Huge (H) versions that have more parameters and complexity. The "/32" refers to the input resolution, indicating that each image is divided into patches of size 32x32 pixels (Dosovitskiy et al., 2021). This state-of-the-art ViT is chosen because it is available via PyTorch and requires only a moderate computation time.

For all models, pre-trained version will be used, which have been trained on a large dataset and therefore have adjusted weights. Using Transfer Learning is more interesting than making the model learn from scratch, as pre-trained models usually perform better and are easily accessible through Python libraries (Milleville, 2024). In this thesis, the pre-trained models will be accessed through the PyTorch library, as it is commonly used and has stable versions of most of the recent CNNs (*PyTorch*, n.d.).

### 3.4.2 Aim of the evaluation

The objective of comparing different models, data splitting methods and stages of hyperparameter tuning is to determine how to create the best possible model and understand how these different aspects affect performance. The initial comparison focuses on the performance of various models on two differently split datasets: the interval-based dataset and the road-based dataset. For this comparison, all pre-trained models are trained on the same two datasets using the following hyperparameters:

- 50 epochs,
- a learning rate of 0.001,
- a batch size of 32,
- weight decay of 0.0005,
- early stopping patience of 10,
- dropout of 0.5
- and the Adam optimiser.

The purpose of this comparison is to evaluate how different model architectures impact performance in classifying cycle road surface types and to identify the most suitable architecture for this task, while also examining if the data splitting method leads to overfitting.

In a second comparison, the hyperparameters of the best-performing models will be tuned and applied to the most suitable dataset configuration. As mentioned in section 2.3 Optuna will be used for automated hyperparameter optimisation. The following search ranges will be used for the hyperparameters:

- learning rate: 0.00001 to 0.005 on a logarithmic scale,
- number of epochs: 10 to 100,
- batch size: 8, 16, 32 and 64.

Optuna requires defining the number of trials (set to 30), which is how many combinations will be searched, the number of jobs (set to one), which means how many trials can be carried out at the same time and the timeout duration (set to 20 000 s), how many seconds to run the program. These values were chosen to maximise hyperparameter optimisation with the available time and computing resources: CUDA, an external device that is used by PyTorch to run large and powerful models, has a limit on how much memory can be used for free, and this limit is commonly reached if the hyperparameters are set to higher values. By comparing the models with tuned hyperparameters to the models without, the influence of hyperparameter tuning can be assessed, leading to the identification of the optimal classification model.

### 3.4.3 Evaluation metrics

Having established the aspects to be compared, it is necessary to explain the methodology of the comparison. In brief, the comparison of every model configuration will be based on the following evaluation forms:

- a range of performance metrics for the total model: train and test accuracy, Cohen’s kappa and the F1 score,
- the performance per class as visualised in a confusion matrix,
- the evolution of the performance per epoch as visualised on learning curves
- and the use of different areas in the image to decide the classes as visualised on saliency maps.

#### Performance metrics

To evaluate the model, some metrics will be created. First of all, the amount of true positive (TP) and true negative (TN) cases will be collected. The former are instances when an image is rightly classified as a certain class and the latter are cases that are rightly not classified as that class. When an image is wrongly classified as the class in question, it is called a false positive (FP). A false negative (FN) is when the opposite happens (mistakenly not classified in that class). From those, one of the most frequently used metrics, the **accuracy** (equation 1), can be collected. It is the ratio of true positive and true negative cases over all the instances (Hossin & Sulaiman, 2015). This performance metrics will be calculated for both the training set and testing set, to be able to compare both accuracies and detect overfitting. Another way to evaluate the performance, is with precision (equation 2): how many of the cases that are classified as positive, are really positive? The recall (equation 3) shows how many of the originally positive cases are truly classified as such. Taking the average of the recall and the precision results in the **F1-score** (equation 3), a metric that is commonly used for imbalanced datasets (Hossin & Sulaiman, 2015).

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$precision = \frac{TP}{TP + FP} \quad (2)$$

$$recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = \frac{2 * recall * precision}{recall + precision} \quad (4)$$

Additionally, Cohen's Kappa will be calculated. This metric shows how well the model is performing compared to a basic 'random' classification. It is calculated using the equation 5, where  $p_0$  represents the overall accuracy of the model, and  $p_e$  measures the agreement between the model predictions and the prediction by chance (McHugh, 2012).

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \quad (5)$$

Cohen's Kappa ranges from 0 to 1, with 0 being that the model is classifying as bad as a random model, and 1 being that the model is classifying everything perfectly (and way better than the random model). The metric will be calculated for the test set, to show how the model performs on new data.

In total, there will be four metrics that are calculated every epoch while training: train accuracy, test accuracy, F1- score and Cohen's Kappa. They are calculated with the commonly used python library Scikit Learn and saved into CSV-files (Comma-Separated Values file).

## Confusion Matrices

To better understand the factors influencing model performance, it is essential to examine the road surface classes individually and identify any confusion between them. This is achieved by visualising the true and predicted classes for each category in a confusion matrix. The matrix displays the number of images that fall into each combination of true and predicted classes. The diagonal from the top left to the bottom right represents the correctly classified images.

## Learning Curves

All performance metrics are calculated per epoch during training and saved into a CSV file (Comma-Separated Values file) for each model. This allows the evolution of model performance to be visualised on graphs and enables comparison between training and testing accuracy. The learning curves, along with the confusion matrices, are created using the Python library Matplotlib.

## Saliency maps

The way a neural network classifies images into classes is not clear to the user. The models are therefore often called a black-box system, in which you cannot look inside the box what is going on. One of the ways researchers have been trying to inspect the workings of the model, is with class activation maps (CAM) or saliency maps, post-hoc explainable AI methods which identify the “*portions or clusters of pixels within an image that significantly influenced the model’s ultimate classification*” (Bhagya et al., 2023). The result of CAMs is often a visualisation of the picture with a heat map overlay that shows which pixels were more relevant. There are a number of CAMs developed, which all have different approaches. Examples are Grad-CAM (Selvaraju et al., 2017) and its modified version Grad-CAM++ (Chattopadhyay et al., 2018), Ablation-CAM (Desai & Ramaswamy, 2020), Score-CAM (Wang et al., 2020), Eigen-CAM (Muhammad & Yeasin, 2020) and LayerCAM (Jiang et al., 2021). But which CAM to use? According to Bhagya et al. (2023), who did a comparative study on all of these CAMs (except for LayerCAM), **Score-CAM** explains the decision-making process of their Neural Network the best, because it uses a weighted average of different activation maps. These CAMs (except for LayerCAM) are also available in a PyTorch package (Jacob Gildenblat & contributors, 2021), which make them easier to implement. In this thesis, Score-CAM will be used to explain the model and explore why some samples are wrongly classified. A saliency map will be made for six examples (one of every road surface class) based on the last layer (or module) of each model, which is the part of the model that makes the final classification decision.

## 3.5 Experiment: comparing different image capturing methods

Using smartphones to collect image data on moving bicycles could result in some blurred images. In most research these images are discarded, but not in this case, because it is inherent to the way the data is collected and a more blurry image could even say something about the road surface type. However, to understand the influence of blur on the model performance and its connection to how the data is collected, a small experiment is carried out.

In a first part of the experiment, a separate dataset will be created by copying the images in the test set and putting them through a blur filter of five different intensities, with 0 being the original image and 5 being a blurry (but still recognisable) copy of the image. These different sets of blurred images will be classified by the best performing model, and their performance metrics will be compared per surface class by visualising a histogram. Then, the mean Laplacian score will be calculated and visualised for every surface class in every blur level. The Laplacian filter detects edges with zero-crossing detection, resulting in a score for every image. The higher this score is, the more edges there are on the image and therefore more fine details and texture and less blur (Moraru et al., 2018). By calculating this, it can be deducted how the surface type influences the blurriness, and how this influences the model performance.

To see how this blurriness relates to the real-life ways of collecting the images, a second experiment is carried out. Two kinds of bicycles (a city bike and a mountain bike) are combined with two types of cameras (a smartphone and a GoPro action camera) mounted in different ways, resulting in the combinations summarised in Table 3. The idea is that collecting data with a mountain bike could result in less blurry images than with a city bicycle, because of the suspension on the bike. Also, the latest GoPro action cameras have a technology called ‘Hypersmooth’, where the camera’s motion is detected and the image sensor is cropped a little, resulting in a less shaky video (GoPro, 2023). Several degrees of hypersmooth correction can be chosen: minimal (‘On’), automated (‘Autoboost’) and maximal (‘boost’), of which the ‘boost’ option is chosen for this thesis. So, it is expected that the image quality of the GoPro will be very good compared to the smartphone camera. The different ways of mounting the GoPro, with a chest mount on a person and an arm mount on the handlebars, and the phone, flat on the handlebars or with an arm mount, are demonstrated on the images below (Figure 10).



Figure 10 Different ways of mounting a recording device data collection while cycling, from left to right: GoPro on chest mount, GoPro mounted with an arm, Phone on a flat bike mount and Phone on a bike mount an with arm

Table 3 Camera, bicycle and attachment combinations with specifications

Camera	Bicycle	Attachment	Specifications
OPPO A91 (smartphone)	City bicycle	Bike mount with arm	30 frames per second, 1080P resolution
OPPO A91 (smartphone)	Mountain bike Canyon	Bike mount with arm	30 frames per second, 1080P resolution
OPPO A91 (smartphone)	City bicycle	Bike mount flat	30 frames per second, 1080P resolution
OPPO A91 (smartphone)	Mountain bike Canyon	Bike mount flat	30 frames per second, 1080P resolution
GoPro 11	City bicycle	Bike mount with arm	25 frames per second, 4K resolution*, hypersmooth: boost
GoPro 11	Mountain bike Canyon	Bike mount with arm	25 frames per second, 4K resolution*, hypersmooth: boost
GoPro 11	City bicycle	Chest mount	25 frames per second, 4K resolution*, hypersmooth: boost

GoPro 11	Mountain bike Canyon	Chest mount	25 frames per second, 4K resolution*, hypersmooth: boost
----------	-------------------------	-------------	--

\*GoPro images are taken in 4K but converted to 1080P for data storage and computing reasons.

All combinations are used to record the road of the same testing course, a 2,5 km long course in Kanegem which includes four of the road surface types used in this thesis: asphalt, fine gravel, ground and sett (Figure 11). With these ‘wobbly’ roads, the stabilisation of all options is tested. The images are gathered from the videos by extracting a frame every five seconds. These are then put into their corresponding road surface type folders. For every combination, the accuracy will again be tested on the best performing model, and the mean Laplacian score will be calculated. This way, the different combinations can be connected to the first part of the experiment, and their influence on the performance can be compared.

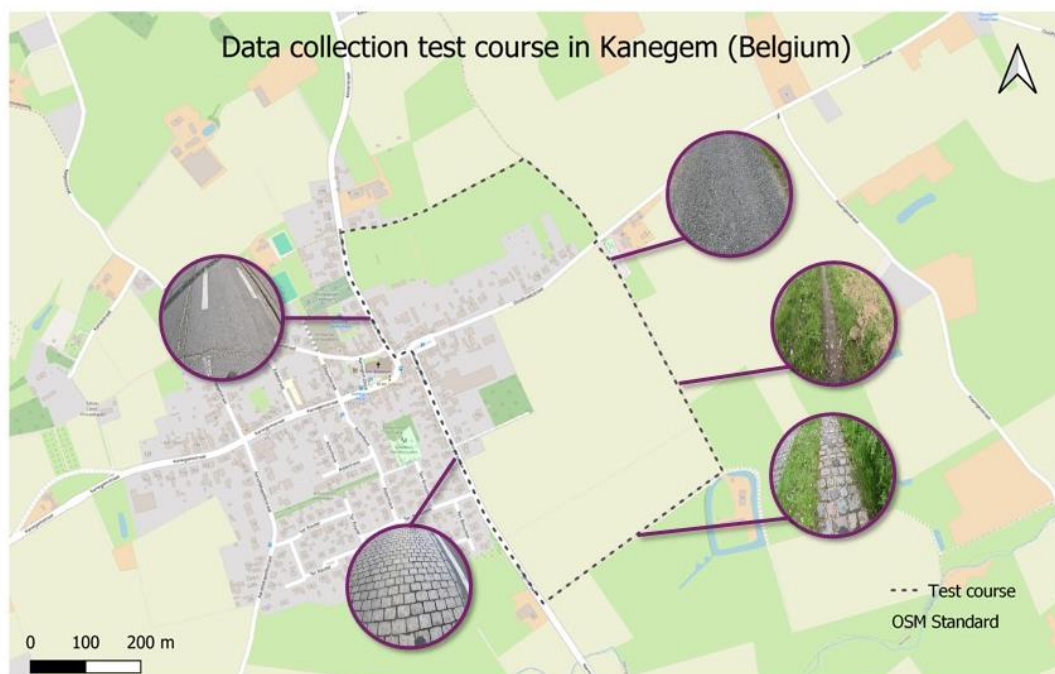


Figure 11 Map of data collection test course in Kanegem (Belgium) with examples of the different road surface types

## 4 RESULTS

The proposed method for bicycle road surface classification results in several significant findings, which are detailed in the following sections. . In Section 4.1, the resulting dataset of 1595 images is discussed, including a few example images. Section 4.2 then compares various model configurations, demonstrating that model architecture significantly impacts performance, with CNN models generally outperforming the Vision Transformer for this classification task. It is also noted that any method of dataset splitting results in overfitting, though this is less apparent with the interval-based dataset. The



top three performing models—GoogLeNet, DenseNet, and MobileNet—undergo hyperparameter optimisation in Section 4.3, leading to further performance improvements. DenseNet emerges as the best model, as summarised in the comprehensive comparison in Section 4.4. DenseNet is then used in the image collection experiment (Section 4.5 where results indicate that while very blurry images degrade performance, blurriness helps distinguish certain classes like sett and fine gravel. The primary factor influencing model performance is not the type of bike or blurriness, but rather the angle at which the camera is mounted on the bicycle.

## 4.1 Resulting dataset

The composition of the two final datasets is described in Table 4.








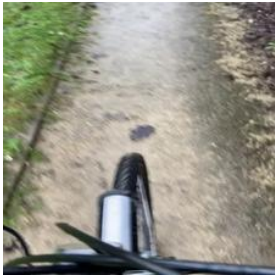







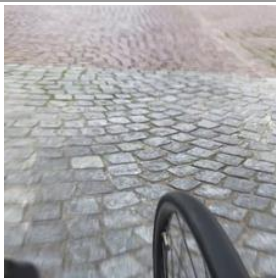


*Table 4 Dataset summary per class and subset*

	<b>Asphalt</b>	<b>Concrete</b>	<b>Fine gravel</b>	<b>Ground</b>	<b>Paving stones</b>	<b>Sett</b>	<b>Total</b>
<b>Training</b>	288	153	158	171	203	141	1117
<b>Validation</b>	82	44	45	49	59	40	319
<b>Test</b>	42	22	22	24	29	20	159
<b>Total</b>	412	219	225	244	294	201	1595

As mentioned in Section 3.3.2, all of the samples were divided into their subsets (training, validation and test set) in two different ways: using an interval-based approach and a road-based approach. In the first case, every 5<sup>th</sup> image is allocated to the validation set, every 10<sup>th</sup> to the test set and the rest to the training set. The second approach was more manual: the images that show the same roads were put into the same subset, at the same time keeping the right number of images per class and subset.

Consequently, both datasets have 1117 images in the training dataset, 319 in the validation set and 159 in the test set, which results in a total of 1595 images. 412 of the images in the dataset are in the asphalt class, 219 in concrete, 225 in fine gravel, 244 in ground, 294 in paving stones and 201 in the sett class. These numbers are the same for both datasets, but the actual images are not, because of the way they were split. The goal was to create a very diverse dataset by having different weather conditions, using different cameras and angles to approach a real-life collection method. This variety is visible in the examples in Table 5, where for each type there is an example image in dry condition, wet condition and with shadows.

Table 5 Example images in various conditions for the six road surface types

	Dry	Wet	Shadow
<b>Asphalt</b>			
<b>Concrete</b>			
<b>Fine gravel</b>			
<b>Ground</b>			
<b>Paving stones</b>			
<b>Sett</b>			



## 4.2 Comparison of different models and datasets without hyperparameter tuning

As discussed in Section 2.2.2 and Section 3.4.1, the different models that will be compared in this thesis are GoogLeNet, VGG16, ResNet101, DenseNet201, MobileNet V2, Efficientnet V2 M and ViT-B/32. This comparison aims to understand the influence of different model architectures, the difference between ViT and CNN models and the potential for overfitting. Additionally, the two data splitting methods will be evaluated.

### 4.2.1 Performance metrics

The resulting performances of the models in the different phases with the different datasets can be found in Tables 6 (for the interval-based dataset) and 7 (for the road-based dataset). The best performing models are indicated in bold, these were used for the further hyperparameter tuning with the road-based dataset. With the interval-based dataset, most models achieve high performance (Table 6), with most F1-scores above 85%. GoogLeNet and EfficientNet emerge as the top performers for this dataset, representing the oldest and newest CNN models in the comparison. Contrary to expectations, the VisionTransformer underperforms significantly, with an accuracy of less than 60% and a Cohen's Kappa of only 0.4479 for the interval-based dataset. VGG16 also performs below par, although not as poorly as the VisionTransformer. However, the difference between the train and test accuracies of all models (ranging from 0.01 to 0.12), suggests some degree of overfitting.

Table 6 Result for the seven models after training on the interval-based dataset with the following hyperparameters: epochs=50, lr=0.001, batch size=32, early stopping patience=10, dropout= 0.5

Model name	Train accuracy	Test Accuracy	Cohen's Kappa Testset	F1 Score
<b>GoogLeNet</b>	<b>0.9660</b>	<b>0.9245</b>	<b>0.9082</b>	<b>0.9244</b>
VGG16	0.8935	0.7799	0.7294	0.7775
ResNet	0.9579	0.8742	0.8463	0.8750
DenseNet	0.9534	0.8679	0.8392	0.8664
MobileNet	0.9472	0.8742	0.8448	0.8654
EfficientNet	<b>0.9928</b>	<b>0.8994</b>	<b>0.8774</b>	<b>0.8993</b>
VisionTransformer	0.5685	0.5597	0.4479	0.5095

When trained and tested with the road-based dataset, the models' performance deteriorates (Table 7). The test accuracy is consistently and significantly lower than the train accuracy, indicating that the models are overfit. Here, the VisionTransformer and VGG16 again underperform, but the top-performing models are GoogLeNet, DenseNet and MobileNetV2, differing from the interval-based dataset. When comparing the CNNs to the ViT, a clear difference is visible: the train accuracy is on average 0.33 higher for the CNNs, and the F1 score 0.43.

Table 7 Result for the seven models after training on the road-based dataset with the following hyperparameters: epochs=50, lr=0.001, batch size=32, early stopping patience=10, dropout= 0.5

Model name	Train accuracy	Test Accuracy	Cohen's Kappa Testset	F1 Score
<b>GoogLeNet</b>	<b>0.9463</b>	<b>0.7233</b>	<b>0.6576</b>	<b>0.7107</b>
<b>VGG16</b>	0.8694	0.5660	0.4753	0.5660
<b>ResNet</b>	0.9320	0.6667	0.5981	0.6726
<b>DenseNet</b>	<b>0.9821</b>	<b>0.7736</b>	<b>0.7228</b>	<b>0.7642</b>
<b>MobileNet</b>	<b>0.9382</b>	<b>0.7358</b>	<b>0.6792</b>	<b>0.7334</b>
<b>EfficientNet</b>	0.9355	0.6981	0.6346	0.7013
<b>Vision Transformer</b>	0.6007	0.2453	0.1019	0.2598

The F1 scores are, on average, 0.19 lower for the road-based dataset compared to the interval-based dataset, which is similar for the other performance metrics, showing that the way the data is split definitely impacts the performance. With both datasets, GoogLeNet is one of the top performing models, but the other top performers, EfficientNet for the interval-based dataset and DenseNet and MobileNet for the road-based dataset, differ. However, the difference between their performance metrics is not significant enough to state that these model architectures are more suitable for different datasets. More data and training would be needed for such a claim.

#### 4.2.2 Confusion matrices

To dive deeper into the model's classification mistakes, it is useful to look at the classes separately on the confusion matrices, which show the difference between the predicted and true labels of the test sets (Figures 12 and 13 on the next pages). The Vision Transformer stands out, because it has less of a clear diagonal line in its matrix, it actually confuses almost all classes with all other classes. What is also striking, is that the asphalt class consistently has a high number (and a darker green), but this is because the asphalt class is overrepresented in the test set: there are 42 asphalt test samples, but 20 to 29 samples of the other classes. A more informative phenomenon that can be deduced from the confusion matrices (especially those of the road-based dataset), is that often (and not surprisingly), similar looking surface types are confused with each other: asphalt and concrete, fine gravel and ground, and paving stones and sett. This is especially visible in the better-performing models, like GoogLeNet, DenseNet and MobileNetV2. Furthermore, most models have a difficulty classifying the concrete class: only around half of them are classified as their true class, the rest are often confused with asphalt roads but also all other road surface types.

When the matrices of the interval-based and road-based dataset are compared, the difference in accuracy is also visible: with the interval-based dataset the test set is classified more accurate and the diagonal is distinctly visible, opposed to the road-based dataset results where there is more confusion.

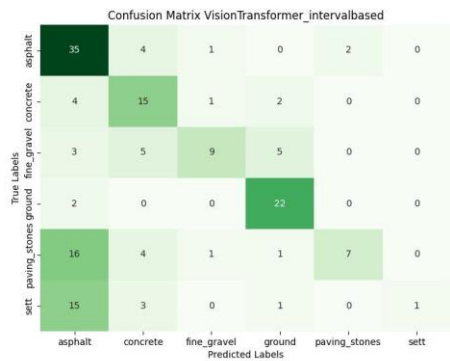
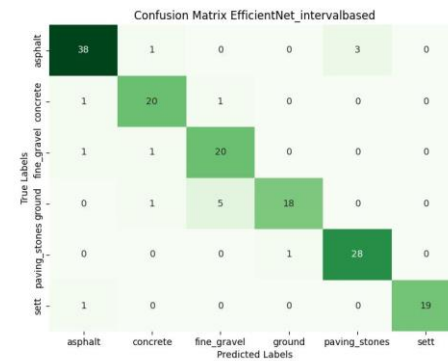
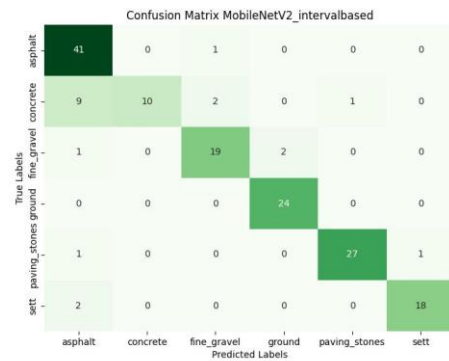
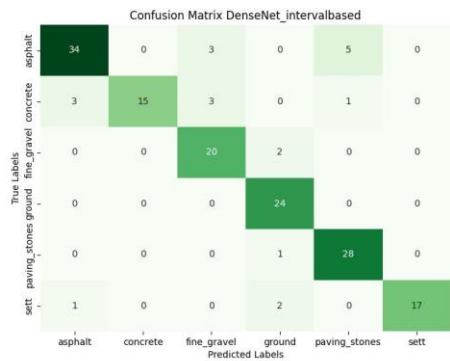
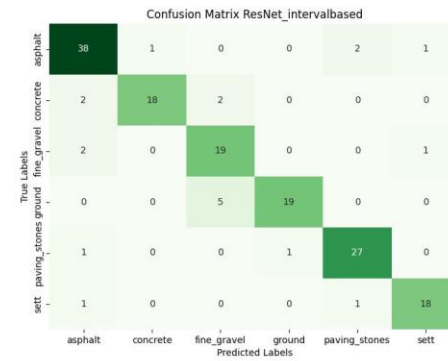
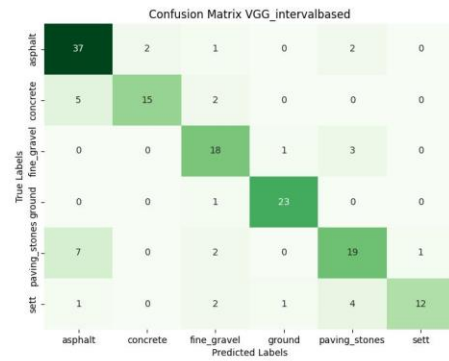
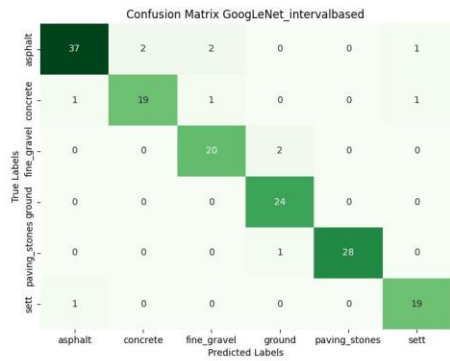


Figure 12 Confusion matrices for each model without optimised hyperparameters for the interval-based dataset

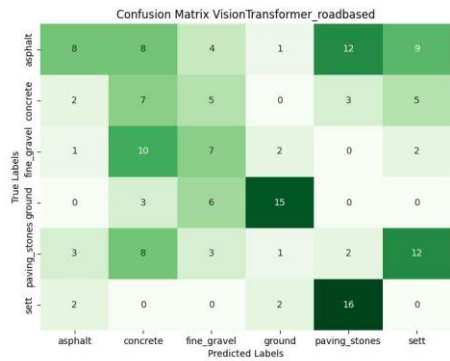
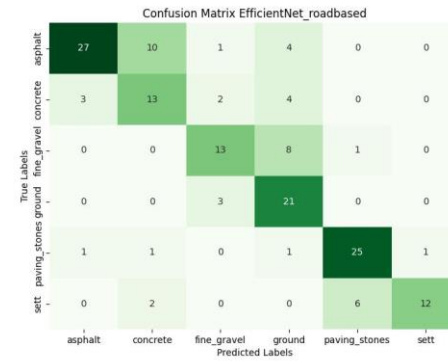
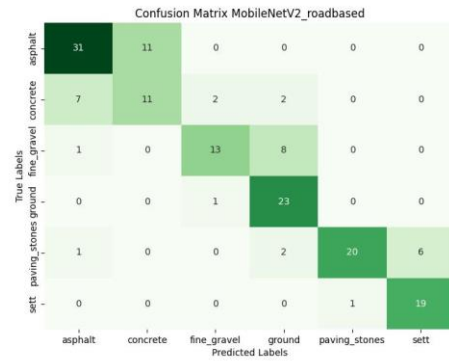
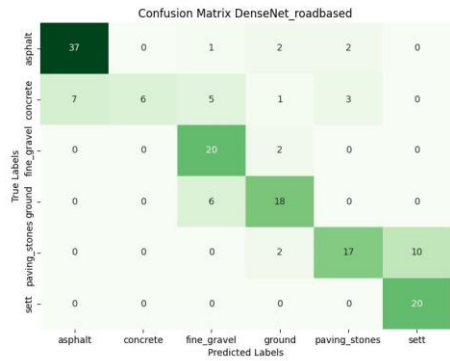
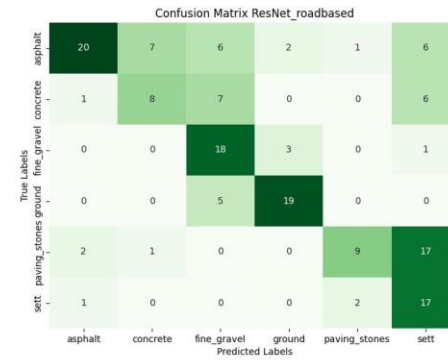
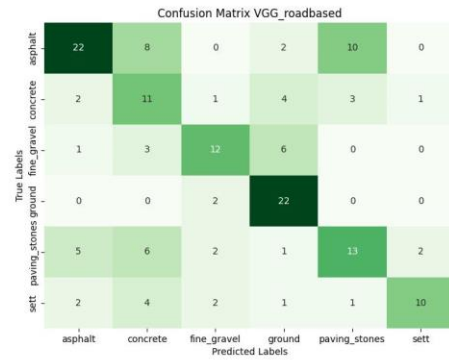
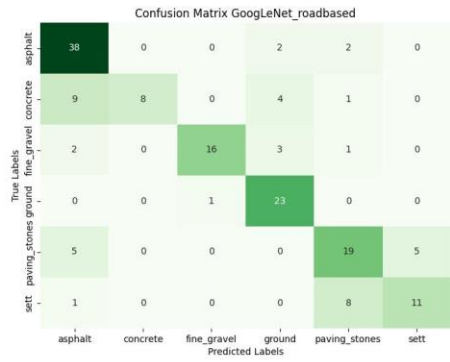


Figure 13 Confusion matrices for each model without optimised hyperparameters for the road-based dataset

### 4.2.3 Saliency maps

In Figures 14 and 15 the saliency maps for six examples for the interval-based and the road-based dataset are displayed, together with their predicted and true classes<sup>1</sup>. More examples of saliency maps, for the two worst performing models VGG16 and the Vision Transformer, and for the best performing models, GoogLeNet, DenseNet and MobileNet, can be found in Appendix 1 to 5. It is clear that the better performing models have a focus that is broad and around the bicycle wheel, which is where the image should be most clear. However, it could also be that they use the bicycle itself as input to classify the road, which would make sense, because on unpaved roads more gravel- and mountain bikes are used, but this will not always be right. The worse performing models, have a very narrow focus (VGG16) or look somewhere else entirely (Vision Transformer). It even seems that the Vision Transformer classifies some images as 'asphalt' (class 0) when they see a car. Some models also seem to classify an image as 'ground' when trees are visible. What is also striking is that on the concrete class image, where there is a clear sunny part and a shadow of a tree, some models focus on the sunny side to classify it, especially with the road-based dataset.

When visually comparing the two datasets, there doesn't seem to be a significant difference that can explain the different performance. On some saliency maps it seems that in the road-based dataset more pixels are used (the red area is a little wider). The same models seem to be making similar mistakes with both datasets.

---

<sup>1</sup> The classes correspond with the numbers in the following way: 0 = asphalt, 1 = concrete, 2 = fine gravel, 3 = ground, 4 = paving stones and 5 = sett.



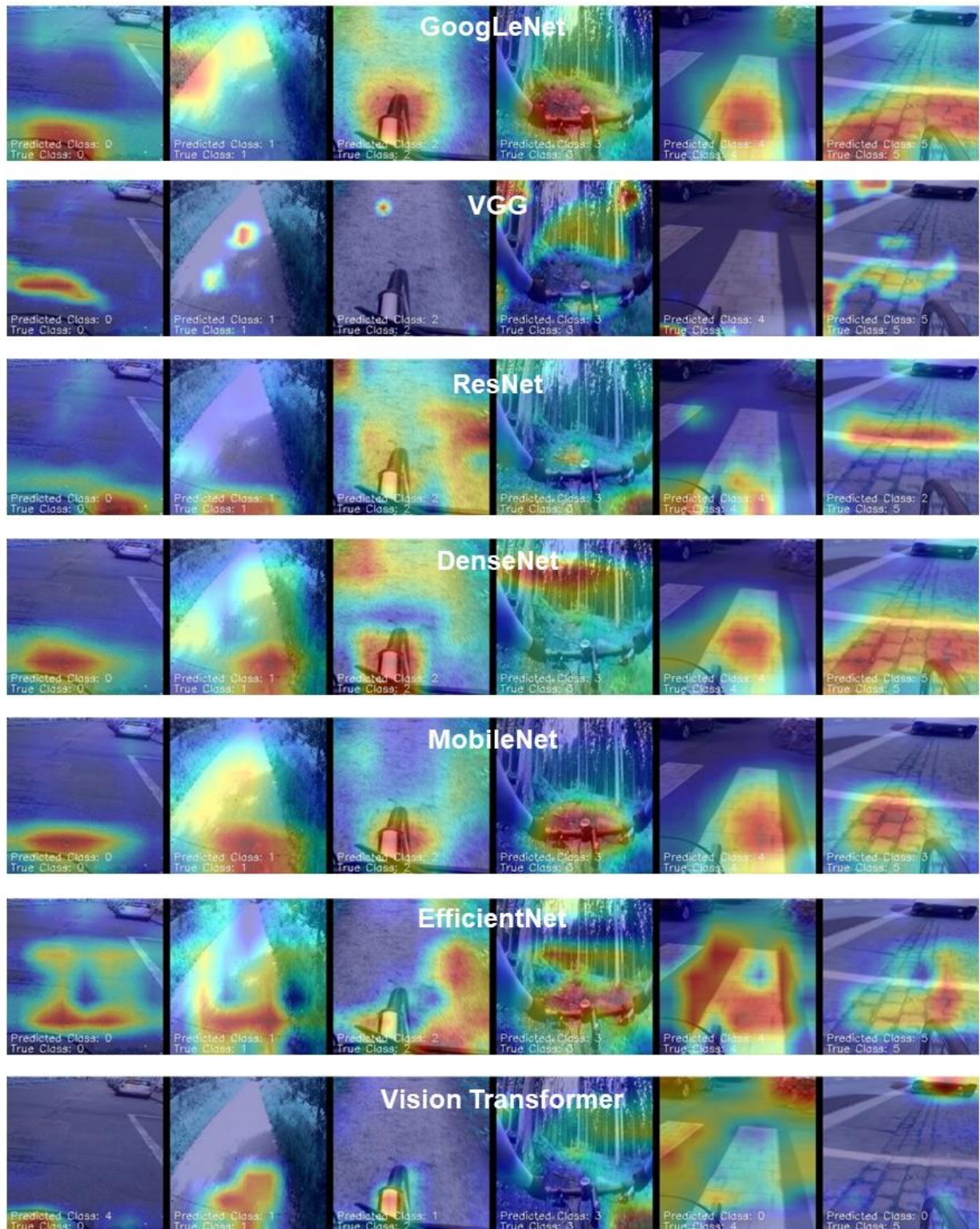


Figure 14 Saliency maps for the seven models with the interval-based dataset



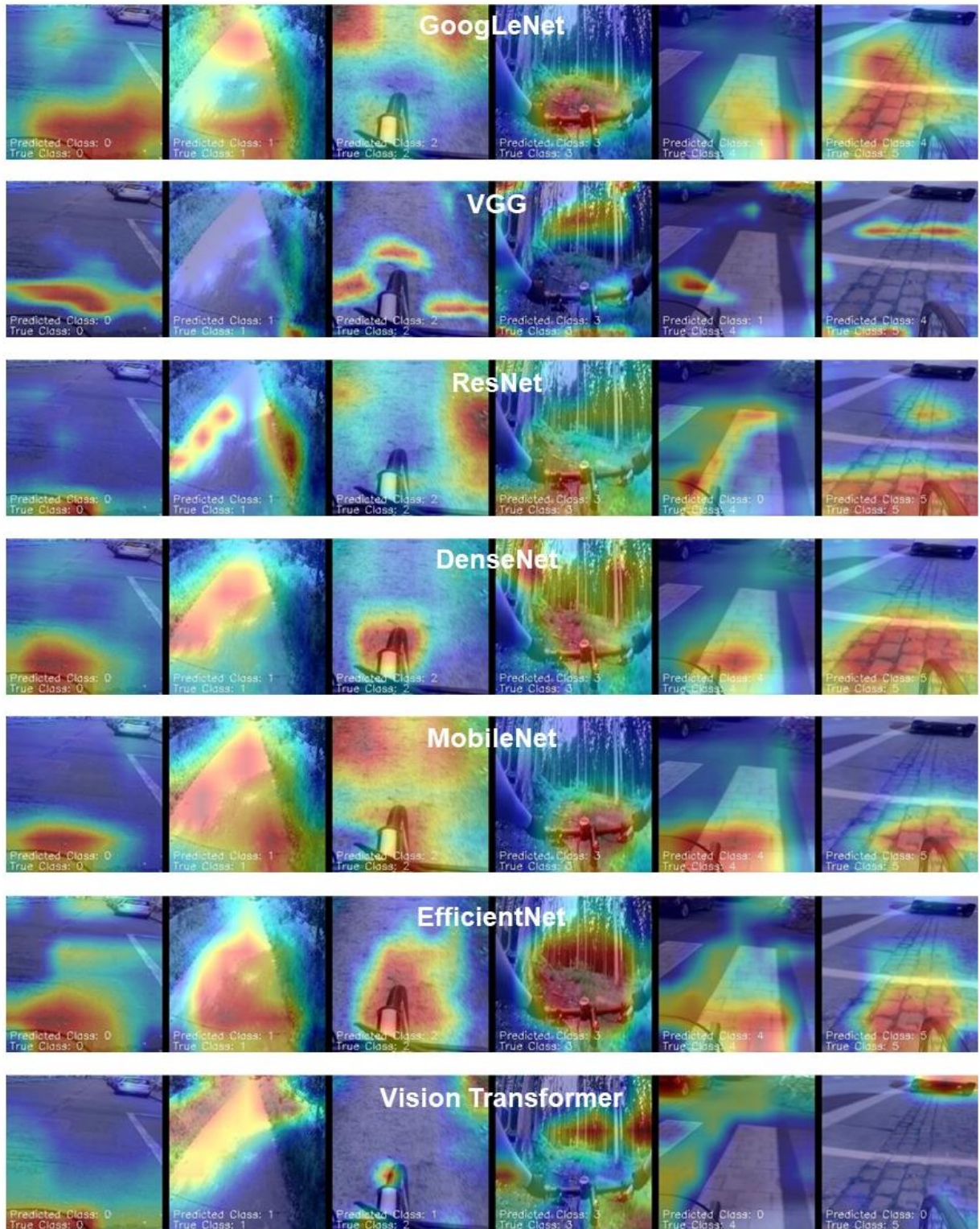


Figure 15 Saliency maps for the seven models with the road-based dataset

#### 4.2.4 Learning curves

Upon examining the learning curve for each model with the interval-based dataset (Figure 16), several things are noteworthy. Firstly, some models stop training earlier than others due to early stopping criteria. For instance, EfficientNet already halts after 20 epochs, while VGG16 stops after 48 epochs, with none of the models reaching the maximum of 50 epochs. EfficientNet stops early because it experienced a lower test loss than training loss ten times by epoch 20. Similarly, DenseNet's training ceases at 31 epochs, as shown in Figure 17 comparing train and test accuracy. Additionally, VGG16 and the ViT exhibit poorer performance compared to the other models: their learning curves begin with a low accuracy (below 0.3) and improve more slowly. Unlike the other models, their accuracy continues to increase slightly towards the end, suggesting that continued training might further enhance their performance.

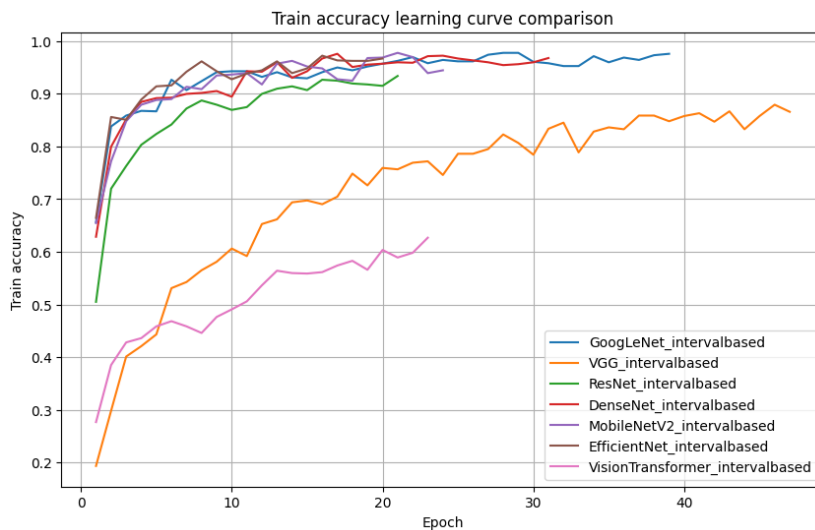


Figure 16 Train accuracy learning curve comparison for the seven models without optimised hyperparameters using the interval-based dataset

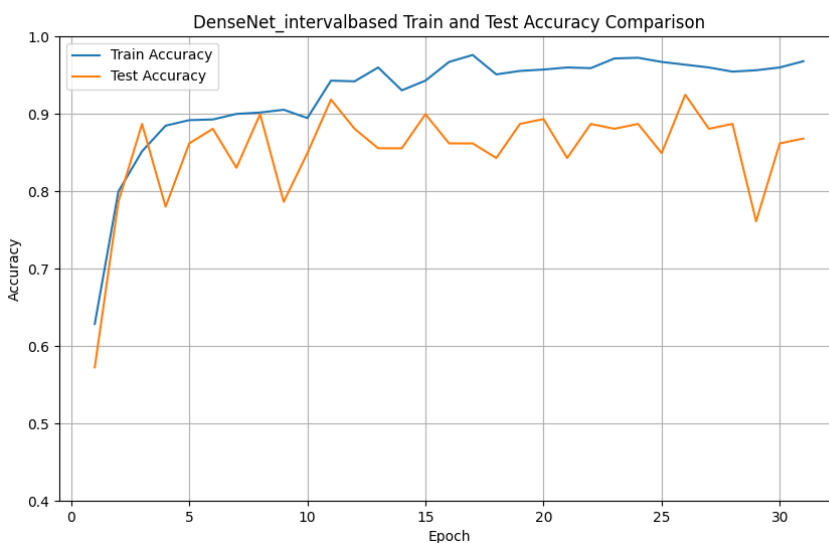


Figure 17 DenseNet train and test accuracy comparison with the interval-based dataset



The learning curves for the road-based dataset (Figure 18) look very similar to those of the interval-based dataset: again, the VGG and ViT model start lower and improve slower than the other models, which start with an accuracy between 0.6 and 0.7 and end above 0.9. However, with the road-based dataset the models stop training 10 epochs earlier than with the interval-based dataset. This indicates that the early stopping is triggered earlier, because the validation performance was worse than the train performance earlier and more often.

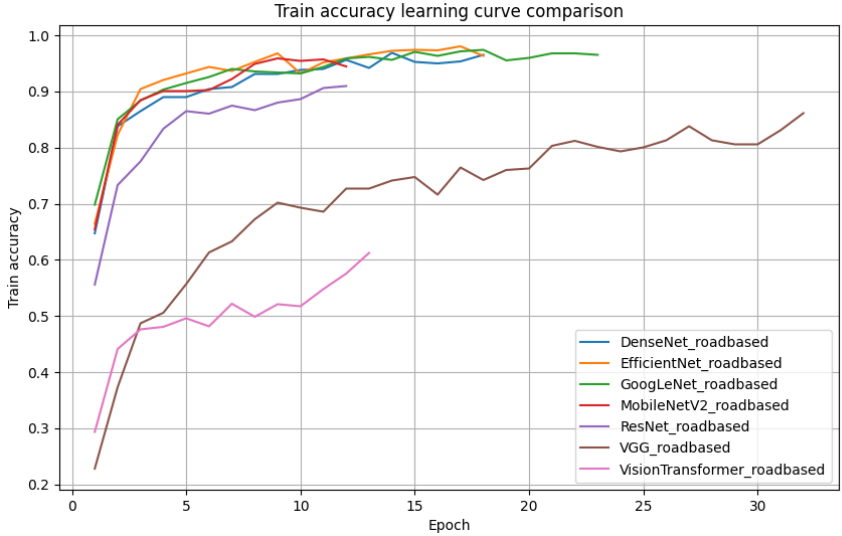


Figure 18 Train accuracy learning curve comparison for the seven models without optimised hyperparameters using the road-based dataset

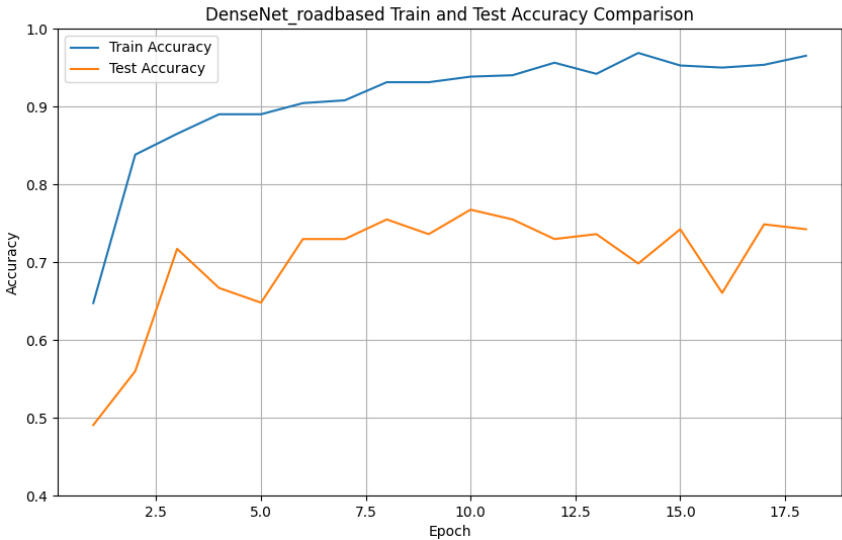


Figure 19 DenseNet train and test accuracy comparison with the road-based dataset

By comparing the train and test accuracy for the interval-based and the road-based dataset for DenseNet as an example (Figures 17 and 19), the influence of the test set sampling methods is clear and visible. The test accuracy is consistently much lower than the train accuracy in the road-based dataset, fluctuating around 0.75, but it stays relatively constant, similar to the interval-based dataset.

This shows again why the early stopping is triggered earlier for the road-based dataset: the validation loss (where the early stopping trigger is based on) often follows the same evolution as the test loss. With the interval-based dataset, the two accuracies converge at the beginning of the training process, which promises good results. However, this also shows the similarity of the training set and the test set: their images are easily recognised by the model because of their similarity. Other models follow a similar pattern in difference between train and test accuracy for the different datasets.

It is clear that the way the dataset is split into subsets, based on an interval or based on the difference in roads, has a big impact on how the model performance is evaluated. For the interval-based dataset there is clearly some temporal correlation between the training, testing and validation sets, which is avoided in the road-based dataset (see discussion). Therefore, the road-based dataset is used for the analysis of the models with hyperparameter tuning.

### 4.3 Comparison of different models with hyperparameter tuning

#### 4.3.1 Hyperparameter optimisation results

The hyperparameter optimisation for the road-based dataset was done with a timeout of 20 000 seconds, for 30 trials and the interval of possible learning rates was set to [0.00001 to 0.005]. The three models completed all 30 trials, of which 23 (MobileNet and Densenet) or 24 (GoogLeNet) were pruned, because the hyperparameters they were testing were not achieving great performance (Table 8). There seems to be no one answer as to what the best batch size or number of epochs is, but the learning rate is low (0.00002842 – 0.00009738) for all models. This shows how big an influence the learning rate has on the training process. The models best validation accuracies are all between 0.85 and 0.9, which looks promising. However, in the optimisation process the validation data is used to adjust the hyperparameters, so to have an accurate way of comparing the optimised models to those models without hyperparameter tuning, the accuracy has to be based on data the model has not already trained on, the test set.

*Table 8 Results of hyperparameter optimisation with Optuna for the road-based dataset with max 30 trials and a timeout of 20 000 seconds*

<b>Model</b>	<b>Number of trials</b>	<b>Number of pruned</b>	<b>Best learning rate</b>	<b>Best epochs</b>	<b>Best batch size</b>	<b>Best validation accuracy</b>
<b>GoogLeNet</b>	30	24	0.00002842	51	32	0.8809
<b>Densenet201</b>	30	23	0.00009027	22	8	0.8589
<b>MobileNet V2</b>	30	23	0.00009738	65	16	0.8683

### 4.3.2 Performance metrics

With their optimal hyperparameters (as explained in the previous section), GoogLeNet, DenseNet and MobileNet were trained again on the road-based dataset. The results are significantly better compared to the models without hyperparameter optimisation (Table 9). The train accuracies have all increased to over 99%, with GoogLeNet having the highest train accuracy. It is, however, more important to look at the test accuracies and other metrics, as they show how the models perform on unseen data. Compared to their equivalents without finetuning, all models had better test accuracies: without hyperparameter tuning GoogLeNet had 0.7233; DenseNet had 0.7736 and MobileNet had 0.7358. Now these have increased with 0.0440, 0.0566 and 0.0189 respectively. Despite this increase, the differences between train and test accuracies are still over 0.15 for all models, meaning that they are overfit. Yet, the train and test accuracies of DenseNet converged: before hyperparameter tuning their difference was 0.2085 but now the difference is only 0.1680. This is also shown in the Cohen's Kappa and F1 scores for the DenseNet model, which are around 0.80, a strong performance that is different from the other models.

Table 9 Result for the three best models after training on the road-based dataset with their optimal hyperparameters (as specified in 4.3.1); early stopping patience=10; dropout= 0.5; optimiser= Adam, weight decay=0.00005

Model name	Train accuracy	Test Accuracy	Cohen's Kappa test set	F1 Score
GoogLeNet	0.9991	0.7673	0.7138	0.7559
DenseNet	<b>0.9982</b>	<b>0.8302</b>	<b>0.7913</b>	<b>0.8233</b>
MobileNetV2	0.9982	0.7547	0.7010	0.7471

### 4.3.3 Confusion matrices

With optimal hyperparameters, the confusions that happened in most models are not solved: similar surface types are still confused with each other (Figure 20). With DenseNet, all 42 asphalt images are classified as such, which could have an impact on the total accuracy of the model. But, if the largest road surface class would be preferred over other classes to improve the accuracy, there would be many other classes confused with this one. This does not happen: less images are even confused with asphalt for the DenseNet model than for the other models, and the diagonal with the correctly predicted classes is even more clear than for the other models. The most difficult surface type to predict for all models with hyperparameter optimisation seems to be concrete, only half or less of the images in this class is classified correctly, and it is mostly confused with asphalt and sometimes with fine gravel or ground. This confusion also happened with the models without hyperparameter optimisation, only DenseNet has improved in this prediction, showing that the concrete class is difficult to predict for the models and perhaps needs more sample images to be distinguished.

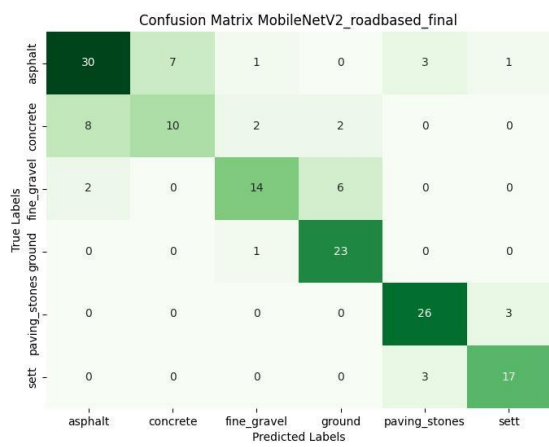
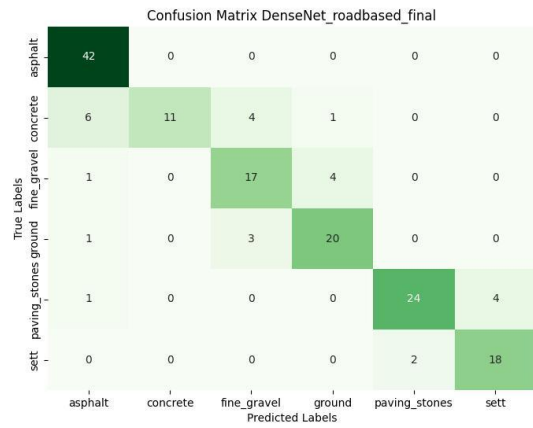
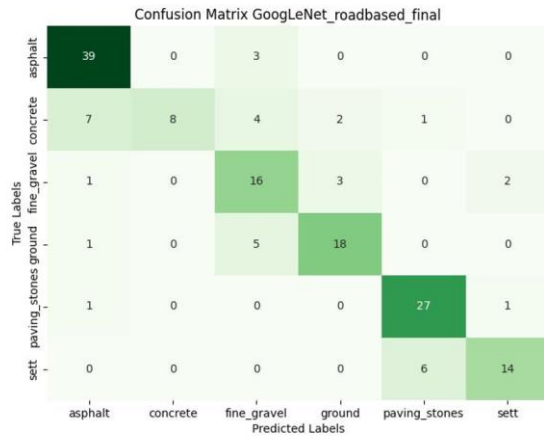


Figure 20 Confusion matrices for the models with hyperparameter optimisation using the road-based dataset

#### 4.3.4 Saliency maps

Based on the saliency maps, it seems that the models with hyperparameter optimisation do not necessarily look at different features compared to the default models to classify the images (Figure 21). The field of focus is close to the bicycle wheel, the area where the structure of the road surface is the most clear, and is relatively wide, meaning it uses an adequate amount of pixels to classify the image. Nevertheless, the models might be making some of the same possible mistakes as made without hyperparameter optimisation. Again, it seems like the trees in the background are used to identify a ground image, which could be a problem if the cyclist is driving through the forest on a paved cycling track, and sometimes (with asphalt, fine gravel and ground images) the bicycle wheel or handlebars are within the focus field, implying that they could also be used to classify the images. In short, the saliency maps do not explain the difference in model performance due to hyperparameter optimisation.

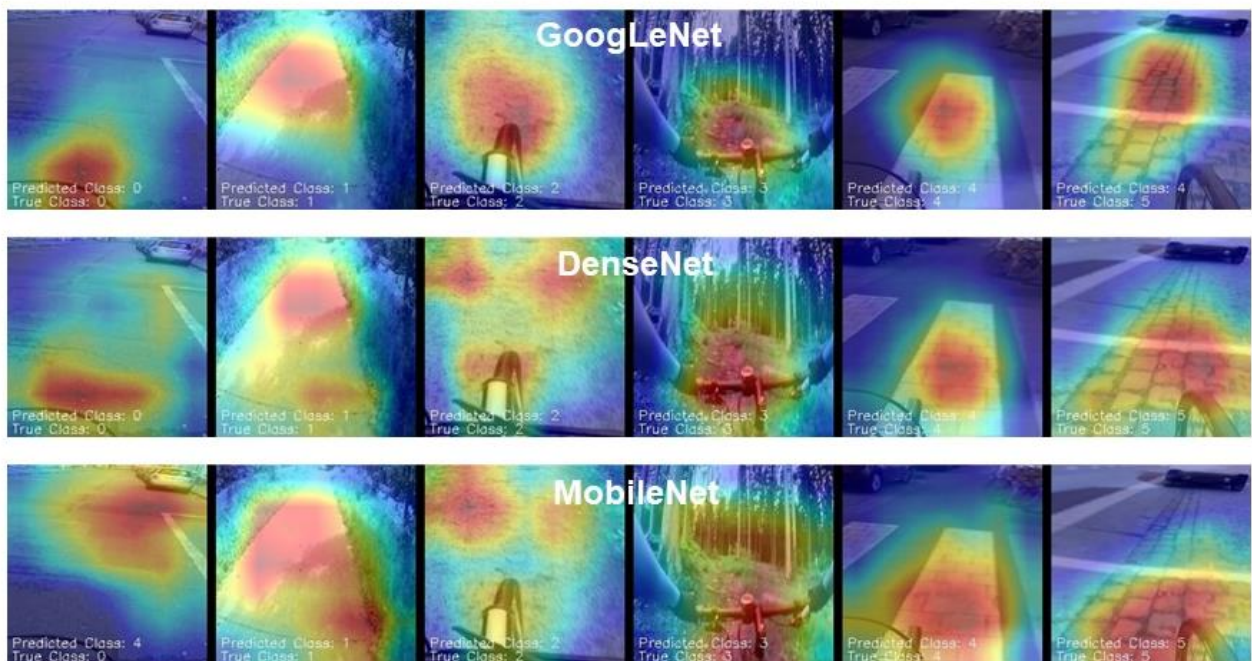


Figure 21 Saliency maps of models with hyperparameter optimisation with the road-based dataset

#### 4.3.5 Learning curves

When DenseNet is hyperparameter-tuned, the learning curve exhibits similar patterns as seen with the models before tuning (Figures 17 and 19), but with notable improvements. The training accuracy increases further, and the test accuracy stabilises around 0.8, compared to 0.75 previously. As mentioned in section 4.3.2, the gap between train and test accuracies has narrowed, which is also visible on the learning curve (Figure 22). In the initial epochs, the train and test accuracies are similar, which is typically a sign of good performance. This improvement demonstrates the impact of optimal hyperparameter selection on model performance. However, the persistent gap between train and test accuracies still indicates overfitting. The training process stops after 15 epochs, three epochs earlier

than the untuned model, suggesting that the validation loss was consistently worse than the training loss early in the process, triggering early stopping sooner.

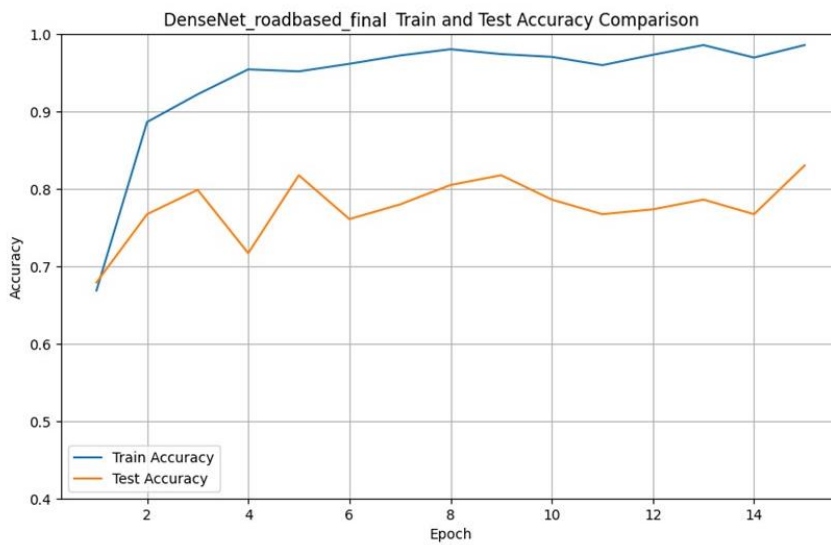


Figure 22 Hyperparameter-tuned DenseNet train and test accuracy comparison using the road-based dataset

#### 4.4 Total comparison

All different configurations and their F1 scores are summarised in Figure 23. This demonstrates clearly the performance difference between the interval-based and the road-based dataset. It also shows the increase of the F1 score after hyperparameter tuning, where DenseNet improves the most. Therefore, it can be concluded that the DenseNet model with hyperparameter optimisation performs the best on the given classification task, with a test accuracy of 83.02% and a F1 score of 82.33%. Before tuning, these values were 77.36% and 76.42 respectively. Hence, the resulting trained DenseNet model will be used for the following data collection experiments.

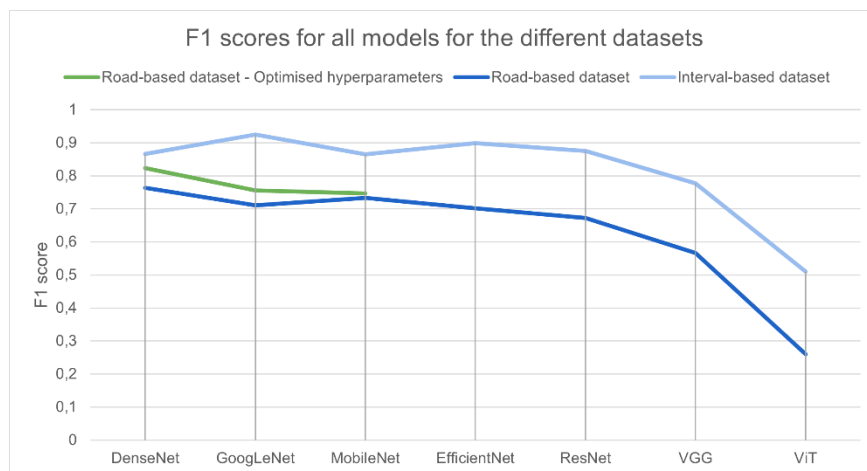


Figure 23 F1 scores for all models for the different datasets with or without hyperparameter optimisation, sorted from higher to lower score

## 4.5 Experiment: data capturing methods

### 4.5.1 Influence of blur on performance

For the first part on the data quality experiment, 159 images were put through five stages of blurring, resulting in five datasets. An example of a blurred image is given for every blur level in Table 10, with level 0 being the original image, and level 5 the most blurred version of the image. When tested with the DenseNet model with hyperparameter optimisation, the interval-based dataset achieved an accuracy of 83.65%, which stayed the same for the first level of blurring. As the images were more blurred, the accuracy gradually got worse: 75.47% at level 2, 62.26% at level 3, 56.60% at level 4 and 55.35% at level 5 (Figure 24).

Table 10 Example images of the ground class in different blur levels

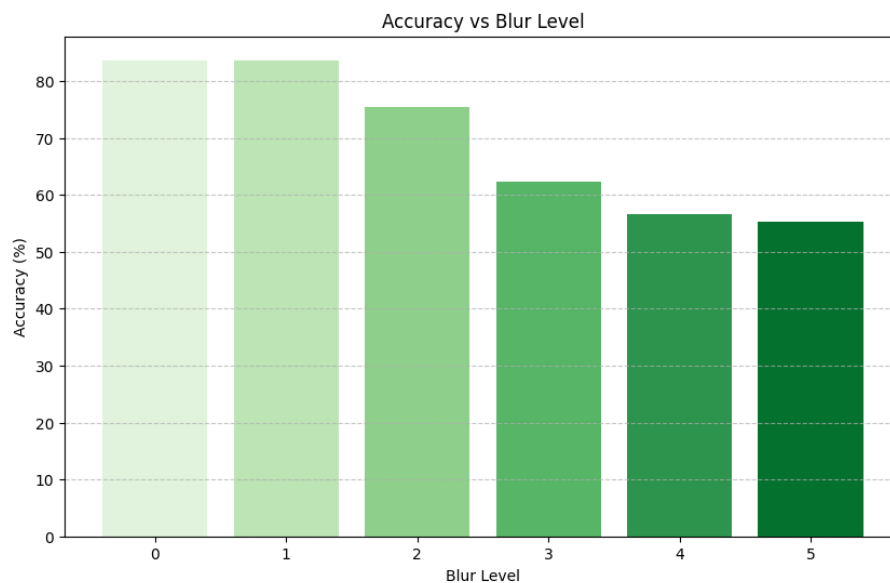


Figure 24 Accuracy per blur level

Putting the confusion matrix for level 0 next to that of level 5 (Figures 25 and 26) provides insight on the way the blurring affects each class. For most classes, the blurring results in a less accurate prediction, confusing them with similar classes (e.g. asphalt with concrete) but also with very different classes (e.g. asphalt with sett). However, for the sett and fine gravel surface types the accuracy actually increases, implying that the model uses the blur (absence of edges) to decide that images belong in these classes. This finding is similar to that in the experiment with different data collection methods. This also results



in the other classes being confused with sett and fine gravel more often, even if they don't seem very similar.

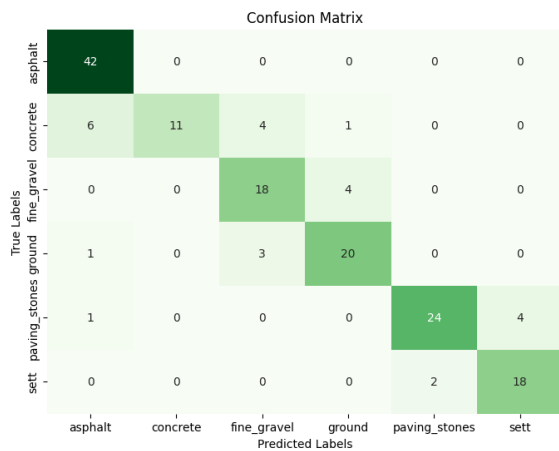


Figure 25 Confusion matrix for blur level 0

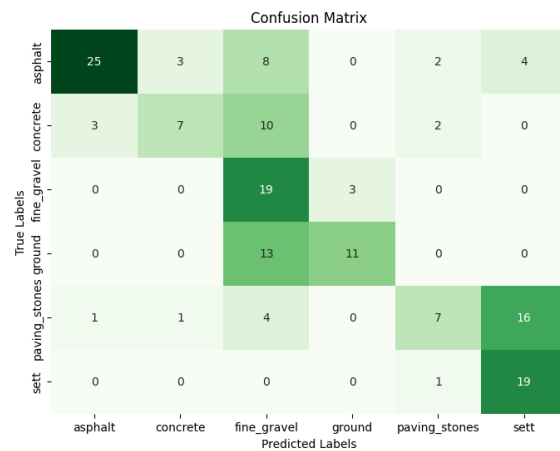
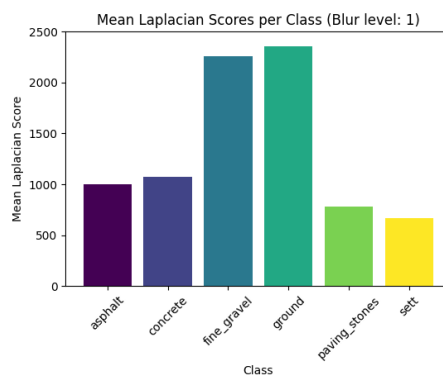
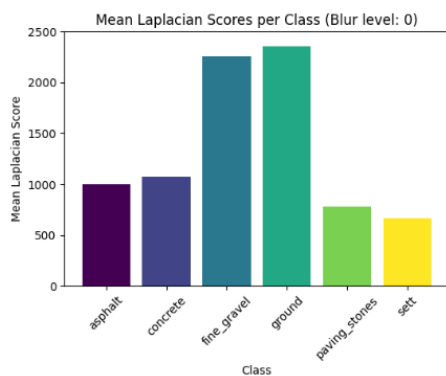


Figure 26 Confusion matrix for blur level 5

The Laplacian score changes after blurring the images, as can be seen on the bar graphs in Figure 27. Blur level 0 and 1 are nearly identical, with a maximum Laplacian score of more than 2000, but as the images get more blurry the score drops to below 200 and even below 20 for the severest blurred images. There is a big difference in Laplacian scores between the different road surface types: the classes 'fine gravel' and 'ground' have almost double the score of the other classes, and in the unblurred images the 'sett' class is lower. This could indicate that the unpaved classes have higher quality images (perhaps because more of them are shot with a GoPro and a iPhone) or, more likely, because of the inherent characteristics of the classes: with grass or stones there are more edges on the image. As mentioned above, it is likely that the model uses edge detection to classify the sett and fine gravel class, causing other images to be confused with the class.



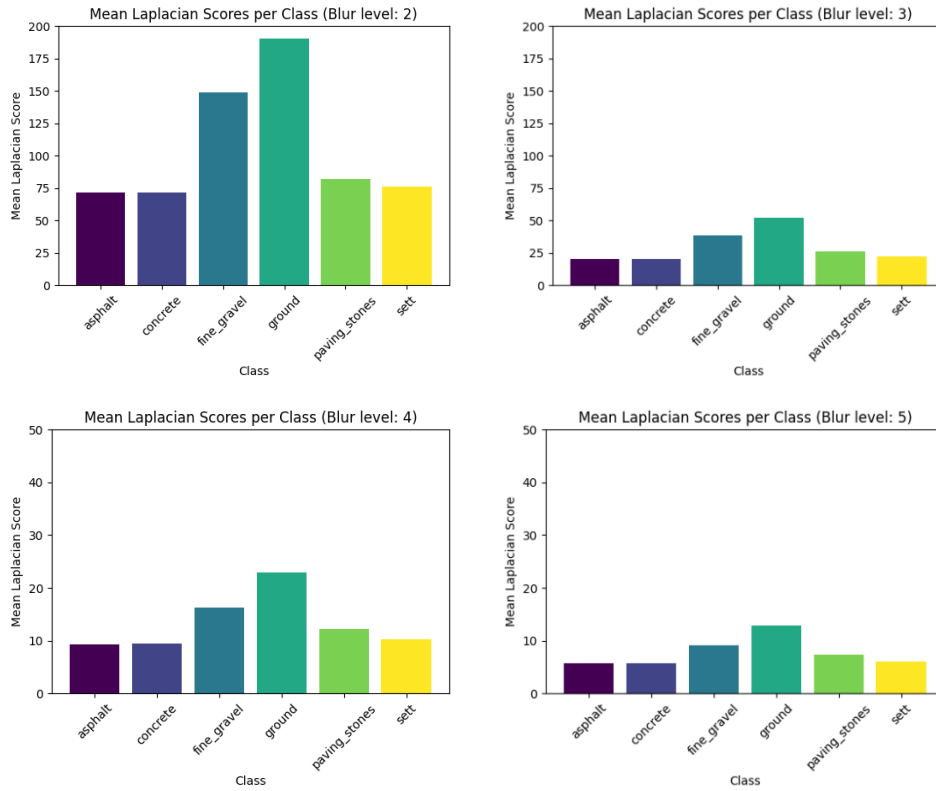


Figure 27 Mean Laplacian scores per class for the different blur levels

To link these results to the images used for classification, the Laplacian scores were also calculated for the total dataset (Figure 28). This shows an even bigger difference between the classes: the ground class has by far the most edges, distinctively more than the fine gravel class. The least edges are again in the sett class, but also in the paving stones class.

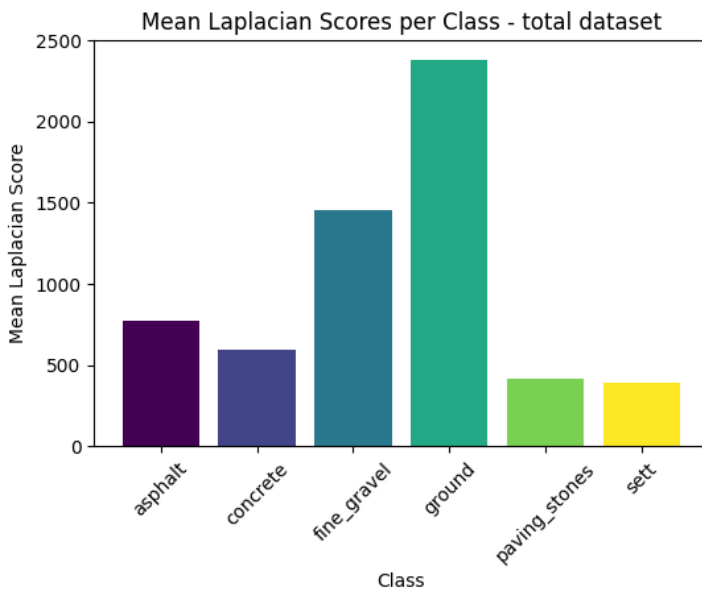



Figure 28 Mean Laplacian scores per class for the total dataset

#### 4.5.2 Influence of data collection methods on blur and performance

In total, eight combinations of collection methods were used, their total accuracy, the mean Laplacian score and an example (taken on the same road) can be found in Table 11. There is some variety in accuracy depending on the combination, ranging from 34.85% to 81.25%, showing that the way the data is collected definitely has an impact on the performance. This variance does not seem correlated with the Laplacian score, but this score seems dependent on the way the data is collected. It is higher when using a GoPro action camera, possible because the higher position captures more of the scenery, but also because of the image quality.

Table 11 Different camera, bicycle and attachment combinations with their resulting accuracy, mean Laplacian scores and an example

Camera	Bicycle	Attachment	Accuracy (%)	Mean Laplacian scores	Example
GoPro 11	Mountain bike	Bike mount with arm	76.81	2972	
GoPro 11	Mountain bike	Chest mount	34.85	3240	
OPPO A91	Mountain bike	Bike mount with arm	76.06	656	
OPPO A91	Mountain bike	Bike mount flat	81.25	660	

GoPro 11	City bicycle	Bike mount with arm	50.43	2893	
GoPro 11	City bicycle	Chest mount	70.09	3845	
OPPO A91	City bicycle	Bike mount with arm	77.14	760	
OPPO A91	City bicycle	Bike mount flat	62.26	440	

When zooming in at the performance of the individual classes by looking at the confusion matrices of each combination (Figure 29), the difference in accuracy is also visible. Every combination of bicycles, cameras and position has some difficulty distinguishing between ground and fine gravel, a confusion that has already been noted in the overall results of the models (Section 4.3). Most collection methods also confuse the surface types sett and ground, possibly because there is a road where both types are combined: two small lanes of cobblestones are bordered by tall grass on the sides. However, on another road segment asphalt and sett road surface types are combined (with cobblestones on the main road and asphalt on the side as parking spaces), but there does not seem to be a remarkable confusion between those. This could relate to the fact that the sett road is much wider and the asphalt is only visible on the very side, and not directly next to the sett road. Based on the confusion matrices, there is no important influence of the type of bicycle on the accuracy of the classes, but the influence of the type of device and the way it was mounted is significant.

There are only three to five images in the fine gravel class for every combination, but for phone-recorded images they are classified almost exclusively as asphalt, while with an action camera, some are recognised as their true class. The phone-recorded images are more blurred, but also closer to the ground, two aspects that could result in this confusion. Another notable phenomenon, is that when using the GoPro mounted on the chest and tilted straight down, some ground-type images are mistaken for asphalt. With the action camera, most sett images are taken for paving stones or ground. This shows the influence of the camera angle: the amount of grass that is shown in the combined road is bigger, resulting in a prediction as ground; the cobblestones are less shaky and more distinguishable, making them look like paving stones; and the ground tracks are wider and have clearer and smoother looking rocks, resulting in a confusion with the asphalt road type. A similar thing happens with the chest-mounted GoPro on the city bicycle (where the angle is tilted more up): a lot of the sett images are classified as ground and paving stones. However, the ground category performs remarkably well in this scenario (48/55 cases are classified correctly). This may indicate that the model uses this kind of view of the bicycle handlebars to classify ground images, as a lot of the training data for the ground category had a similar angle. For the combinations with a phone, the sett category is predicted more accurately.

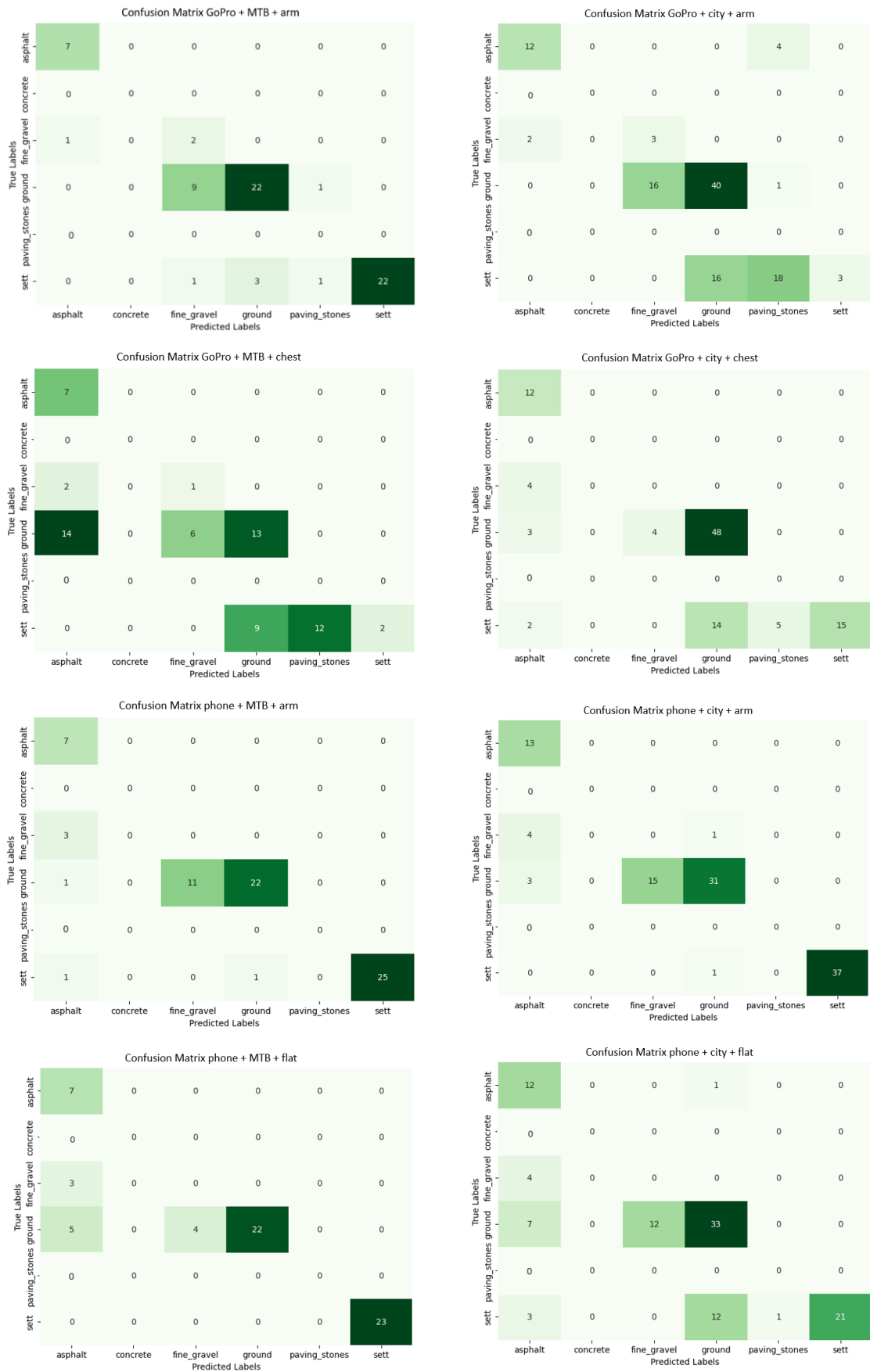


Figure 29 Confusion matrices for the different combinations of camera, bicycle and attachment method with the best performing model and the road-based dataset

By conducting this experiment it becomes apparent how the model uses edge detection to identify the classes, especially the sett and fine gravel class. The amount of blurring definitely has an impact on the accuracy of the model, but because the training set also contains some blurry images, which could be compared to blur level 2 (with a Laplacian score between 50 and 200), using blurred images as input for the model is not necessarily dissuaded.

## 5 DISCUSSION

### 5.1 Main findings

This thesis set out to determine the potential of using cameras on bicycles to collect a road surface image dataset and using CNNs (and ViT) to classify the data. The performance of different pre-trained models on this task was compared and combined with different levels of hyperparameter optimisation and dataset splitting. Ultimately, the best option was used to evaluate the influence of the data collection method on the performance. In the following paragraphs, the research questions posed in the introduction of this thesis are answered.

**How do different state-of-the-art image classification models perform on this task, and which model(s) are the most suitable? How does hyperparameter tuning affect the model performance?**

In this thesis, two kinds of deep learning architectures were used: Convolutional Neural Networks and Vision Transformer. There was a big difference in the performance of these types: the train accuracy of the CNNs for the road-based dataset without tuning was on average 0.33 higher than the ViT, and the F1 score 0.43. As mentioned in Section 3.3, a Vision Transformer typically requires over 100 million images (Martinez-Noriega & Yokota, 2023). This could explain why the Vision Transformer performed so badly on the provided datasets. The best performing models, in turn, were not necessarily the most recent ones: GoogLeNet was one of them, but was the oldest model in the comparison. The others were MobileNet and DenseNet. In the end, the best performing model turned out to be DenseNet with optimised hyperparameters, achieving a test accuracy of 83.02% and a F1 score of 82.33%. This is an improvement compared to the same model before hyperparameter optimisation, which had a test accuracy of 77.36% and a F1 score of 76.42. The other tuned models also improved, showing that hyperparameter tuning does indeed improve model performance.

In related research, similar models show up as the best performing ones. Zhang et al. (2022) also decided DenseNet was the best model architecture for a similar classification task. They applied different CNN models to a dataset of road surface images with different weather conditions in the dark, namely their own CNN, SqueezeNet, VGG, ResNet50, and DenseNet121. Here, DenseNet was chosen as the best model, because it was fast and achieved a validation accuracy of 94.08% for the images with ambient light.



### **How do the models perform under different training/validation/test splitting methods?**

Using the interval-based dataset most models performed very well, with training accuracies (mostly) above 90%. The test accuracies were lower, but still similar, and comparing the test and train accuracy learning curves showed a very parallel trajectory, which is a sign of promising model performance. However, these results were too optimistic, as the train and test sets were very similar and actual unseen data would show a worse performance. The reason for this similarity was the splitting of the data: the total dataset was created by cyclists recording videos of the road, and extracting frames from those videos every five seconds. Because the road does not change every five seconds, several images of the same road were put into the dataset. The validation set and test set were extracted from those frames by taking every fifth and tenth one respectively, the rest ended up in the train set. Therefore, some images in the three subsets are very similar to each other. This results in the models trained on the interval-based dataset being overfit and the performance being overestimated. This issue happens in every dataset to some extent. In ImageNet, a frequently used image dataset for Deep Learning, 890 out of 50K validation images have near-duplicate images in the training set (Sun et al., 2017). These images are often deleted, to avoid overfitting. For that reason, a second dataset was made: the road-based dataset is not split systematically every few images, but every road (or video) is put into a different subset (training, test or validation dataset). This dataset resulted in a worse performance compared to the interval-based dataset, but provided results that are much more reflective of real-world conditions.

### **Do the data collection methods and image quality influence the model performance?**

One of the motivations for creating a model based on phone images, was to have an affordable and scalable method. To determine if this was a reasonable approach for data mining, an experiment was conducted where different ways of recording the road surface (with a GoPro or phone, mounted directly on the handlebars, on an arm or on the chest, with a city bicycle or a mountain bike) were compared based on their accuracies and their Laplacian scores. Next to this, the test set was put through five stages of blurring transformations and were also used to test the model and calculate the Laplacian scores. This resulted in some insight on the DenseNet model with hyperparameter tuning: it predicts images as fine gravel when they have more edges and as sett when they have less (and are more blurry). Artificially blurring the images also influenced the accuracy, but most images are not that intensely blurred when collected. Therefore, collecting images with phones mounted on the handlebars is no problem, but it could be a wise idea to apply a minimum Laplacian score threshold of 50 (which corresponds to being at least blur level 2) when upscaling the collection method to avoid too blurry images. As to the combinations of collecting images: the kind of bike has no influence on the classification, but the position of the device to the ground does. The current method works best if the device (phone or action camera) is mounted on the handlebars, as a lot of the training images are also collected this way. However, if a larger, more diverse dataset would be created and trained on, this effect would likely also change.

## **Is it feasible to use Deep Learning methods to classify the surface types of cycling roads using street-level images?**

While the performance of the proposed CNN and ViT models was not optimal, an examination of their image analysis methodologies shows potential for establishing a robust framework for surface classification. Compared to related research, the performance of the proposed DenseNet model with optimal hyperparameters on the road-based dataset with six surface classes was similar or less. However, the data split and amount of classes in these models also has a major influence on the performance. The model of Balcerek et al. (2020), achieved a mean accuracy of 85.4%, with similar classes. Verstockt et al. (2013) achieved an accuracy of 90% with a visual model based on a decision forest predicting six classes, and 92% when the images were combined with acceleration data. For data collection, the researchers also extracted images from bicycles every five seconds, but it is not clear how many images are used and how they are split, so it could just as well be that the data leakage problem is also present here. Similarly, Beecken & Reinhardt (2019), which reached an accuracy of 99.51%, mention that one of the reasons they achieve this accuracy is because “training and testing data were collected from almost the same environment (little geographical diversity)”. When using only two or three surface classes (like ‘paved’ and ‘unpaved’), related research achieved even higher accuracies (Pereira et al., 2018; Verstockt et al., 2013; Zhang et al., 2022). So, even though the proposed method performed below expectations, it is not entirely possible to compare it to other research, because of the different classes and data leakage problems, which are widely present.

## **5.2 Limitations**

### **Overfitting**

The hidden overfitting problem that happens when having too similar data is a form of data leakage, “the introduction of information about the data mining target that should not be legitimately available to mine from” (Kaufman et al., 2012). In this case, some temporal and geographical correlations are introduced to the dataset, as the images that are extracted from the videos 5 seconds after each other are more similar because they are of the same road. Data leakage is one of the biggest data mining mistakes, and is certainly not unique to this thesis. For example, based on the limited number of images (497 for 9 classes) and the 72 examples provided in the research of Balcerek et al. (2020), they might also be running into the same problem. As a method to split their data into training, test and validation sets, they used random splits that changed every iteration, with 60% of the data in the training set and 20% in both the test and validation set. After training their model, which was based on AlexNet, 10 times, they reached a mean test accuracy of 85.4%, but this might be a too optimistic result. The data leakage problem is not very visible and without introducing some new images to the model it can't be checked, but it is important to keep in mind.

By using the road-based data splitting approach and the resulting dataset in this thesis, this data leakage is avoided, so the results are more true to the reality. This way, however, the resulting training set is less diverse, which causes worse performance and overfitting: there is a significant gap between the training and testing accuracy. A larger and correctly split dataset is therefore needed to close this gap and improve the overall performance.

### **Class distinction and multi-class images**

The results of the models per class (as seen on the confusion matrices) showed how there is some confusion between similar classes (especially for the road-based dataset): asphalt was often confused with concrete, fine gravel with ground and paving stones with sett. A possible solution could be to merge those together, but whether this should be carried out depends on the classes and the application of the classification method. When applying this method to acquire information for bicycle road maintenance, for example, it is important to know the difference between asphalt roads and concrete roads. For an analysis of the bicycle road infrastructure per country, a difference between 'paved' (asphalt, concrete, paving stones and sett roads) and 'unpaved' (fine gravel and ground roads) might suffice, but a difference between a road of the sett surface type and a road with paving stones is important for an analysis of cycling comfort. OpenStreetMap can be used for all of these applications, so it would be best to keep as much classes as possible. However, the surface class fine gravel could be seen as a subclass of ground, so merging this would make sense. Even for annotating the images these were the most difficult classes to distinguish from each other.

As can be seen on the saliency maps, the models sometimes have problems with images that have multiple surface types next to each other. This happens, for example, when cycling path has a different surface type than the road, there is another surface type in the distance or the road is simply surrounded by grass (which is part of the 'ground' class). One way of making the model look only at the relevant surface, would be to use semantic segmentation. Semantic segmentation is a kind of CNN framework where each pixel of an image is classified in a certain category, instead of the whole picture being classified as one class (Cornille & Rogge, 2022). Because semantic segmentation is heavily influenced by the size of the dataset, some networks have been developed to get as much information as possible from few shots. This is called Few-Shot Learning. One-Shot Summary Prototypical Network (OSPNet) is a semantic segmentation framework developed by Wang et al. (2021), where three branches with their own combination of layers (a support branch, a query branch and a summary branch) are used to classify pixels of images of unpaved roads into two categories, drivable and not drivable. This is just one example of the many ways semantic segmentation has been used for street-level images. However, using semantic segmentation requires every image to be labelled with polygons for every class, and this is a very laborious process. This method also focuses on the location of the class on the image, which is useful for autonomous driving, detecting potholes, cracks or road signalisation, but not necessarily for classifying the road surface types.

### 5.3 Future research

In future research, the model could definitely be improved by adding more and more diverse data, that could be collected and sampled in a similar way. However, the frame extraction intervals could be set to more than five seconds, resulting in less pictures of the same roads. Too similar images could also be manually deleted, to have just one image per road. In that case, a systematic splitting method, such as with the interval-based dataset, would also be sufficient to use, as the similar image problem would not appear. Future research initiatives could focus on the collection of more diverse data samples from a bigger group of people, potentially with an application capable of geolocation tracking and collecting and preprocessing the videos automatically. Such an approach would result in a substantially larger dataset, thereby possibly improving the model performance. Having multiple people annotate the images would also enhance the distinction between difficult classes, like fine gravel and ground. This thesis could then serve as a foundational reference for model selection and the determination of hyperparameters.

Next to a precise classification model, it would also be useful to connect the classified images to the road network again. This could be carried out by collecting both GPS location and road surface imagery at the same time. By linking these inputs based on the collection timestamp, the coordinates can be retrieved, and then snapped to the nearest known road network, like the roads of OpenStreetMap. Though this seems a straightforward method, some issues could emerge when doing this in practice. The tracking location is sometimes not very accurate, so it could be automatically snapped to the wrong roads. Also, on one road there could be multiple predictions that are not the same, for example four predictions of asphalt, two of concrete and one of paving stones. In these cases, the most frequently predicted class can be used, but sometimes small parts of the road are in a different surface class. Future research should take these issues into consideration.

## 6 CONCLUSION

The goal of this thesis was to investigate the possibilities of using Convolutional Neural Networks (CNNs) and Vision Transformers for image-based cycle road surface classification. Cyclists need quality information about the roads they plan to cycle on, but such information is often lacking. Additionally, methods for obtaining these necessary road properties are not systematically and realistically developed for cyclists, as the focus tends to be more on car-oriented classifications. Various methods exist for image classification, such as Random Forest (RF) and Support Vector Machines (SVM), but Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) are currently top favorites due to their excellent performance. This study compares the way different CNNs and a Vision Transformer perform when given the task of classifying a road surface dataset into six classes. The chosen classes - asphalt, concrete, fine gravel, ground, paving stones and sett – are chosen from frequently occurring categories in OpenStreetMap and because they significantly impact cycling experience and influence cyclists' route choices.

To collect this data, videos were collected using smartphones mounted on bicycles, which were then converted into images. These images were split into training, test, and validation sets using two methods, resulting in two datasets. In the interval-based dataset, every fifth picture was placed in the validation set, every tenth in the test set, and the remainder in the training set. The road-based method, however, assigned entire roads to each subset, maintaining a 70% training set, 20% validation set, and 10% test set ratio. Models trained on these datasets included GoogLeNet, VGG16, ResNet, DenseNet, MobileNet, EfficientNet, and a Vision Transformer. Next to the model architecture, the influence of hyperparameter optimization (finding the best settings that determine the learning process) on model performance was also explored. For the purpose of comparison, a set of performance metrics were calculated, which were interpreted further using learning curves, confusion matrices, and saliency maps, a sort of heat maps that show which pixels a model looks at to classify an image. The potential impact of image quality, resulting from using smartphone-mounted cameras, on model performance was investigated in an additional experiment. In this experiment, Laplace scores, representing image edge sharpness, and the performance with the best model were evaluated across different cameras, bikes, and mounting methods.

The results revealed how the Vision Transformer emerged as one of the worst-performing models, while GoogLeNet, MobileNet, and DenseNet showed the best performance. Additionally, a significant difference was observed between the interval-based and road-based datasets: for the former, the training and test accuracies were high and very similar, while the latter showed a wider gap between the two. The interval-based splitting method appeared to ensure better performance; however, due to that splitting method, the images in the training and test set were much too similar (they often came from the same roads). This introduced hidden overfitting due to data leakage, where geographical location and time information leaked into the model training process. The road-based dataset thus provided a more accurate picture of model performance on real data and was used for the remainder of the study.

Furthermore, hyperparameter optimization significantly improved model performance. For instance, DenseNet achieved a test accuracy of 83.02% with optimized hyperparameters compared to 77.36% without optimization. Similarly, GoogLeNet and MobileNet showed improvements from 72.33% to 76.73% and 73.58% to 75.47%, respectively. The image capturing experiment also demonstrated that image blurriness affected performance only when the images were extremely blurry, and the mounting method had a more significant impact on performance than the camera or bike used.

For a more robust and accurate model, a larger and more diverse dataset should be collected, ensuring that each image corresponds to one road. This dataset can be gathered using similar methods described in this thesis, but applying a minimum Laplacian Score of 50 to filter out blurry images, as experiments have shown that excessively blurry images lead to poorer performance. If the interval between extracted frames would be set to more than five seconds, there would also be less images per road and a smaller chance at data leakage. The data collection could be combined with geolocation tracking in a mobile application, making it possible to collect more data and develop the next step: linking classification results to their locations.

This study provides valuable insights into improving information on cycling infrastructure by comparing different models, data splitting methods, and hyperparameter optimisations. By demonstrating the effectiveness of using bicycle-mounted cameras for data collection and applying advanced deep learning models for image classification, this research has paved the way for complementing information on cycling infrastructure.



## 7 REFERENCES

- Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- Baheti, P. (2021, September 13). *Train Test Validation Split: How To & Best Practices [2023]*. V7 Labs. <https://www.v7labs.com/blog/train-validation-test-set>
- Balcerek, J., Konieczka, A., Piniarski, K., & Pawłowski, P. (2020). Classification of road surfaces using convolutional neural network; Classification of road surfaces using convolutional neural network. In *2020 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*.
- Beecken, J., & Reinhardt, A. (2019). How Smooth is my Ride? Detecting Bikeway Conditions from Smartphone Video Streams. *2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019*, 65–70. <https://doi.org/10.1109/PERCOMW.2019.8730869>
- Bhagya, J., Thomas, J., & Raj, E. D. (2023). Exploring Explainability and Transparency in Deep Neural Networks: A Comparative Approach. *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023*, 664–669. <https://doi.org/10.1109/ICICCS56967.2023.10142255>
- Biddle, S. J. H., Ciaccioni, S., Thomas, G., & Vergeer, I. (2019). Physical activity and mental health in children and adolescents: An updated review of reviews and an analysis of causality. *Psychology of Sport and Exercise*, 42, 146–155. <https://doi.org/10.1016/J.PSYCHSPORT.2018.08.011>
- Chattopadhyay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, 2018-January*, 839–847. <https://doi.org/10.1109/WACV.2018.00097>
- Convolutional Neural Network (CNN)*. (2024). Tensorflow Core. <https://www.tensorflow.org/tutorials/images/cnn>
- Debasish, K. (2024, April 19). *Basics of CNN in Deep Learning*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>
- Denneman, E. (2022, October 10). *Exploring OpenStreetMap's potential as a cycling infrastructure database*. European Cyclists' Federation. <https://ecf.com/news-and-events/news/exploring-openstreetmaps-potential-cycling-infrastructure-database>
- Desai, S., & Ramaswamy, H. G. (2020). *Ablation-CAM: Visual Explanations for Deep Convolutional Network via Gradient-free Localization* (pp. 983–991).
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Hounsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*. <http://arxiv.org/abs/2010.11929>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. <https://www.deeplearningbook.org/>

- Google. (n.d.). [Satellite image of a road near Neringenstraat, Kanegem]. Retrieved May 27, 2024, from <https://www.google.be/maps/@51.009524,3.4081662,203m/data=!3m1!1e3?entry=ttu>
- GoPro. (2023, September 6). *What Is HyperSmooth?* [https://community.gopro.com/s/article/What-is-HyperSmooth?language=en\\_US](https://community.gopro.com/s/article/What-is-HyperSmooth?language=en_US)
- Gupta, L. (2020, November 21). *Comparison of Hyperparameter Tuning algorithms: Grid search, Random search, Bayesian optimization*. Analytics Vidhya. <https://medium.com/analytics-vidhya/comparison-of-hyperparameter-tuning-algorithms-grid-search-random-search-bayesian-optimization-5326aaef1bd1>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016-December*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- henke54. (2018, December). @Tordanik(once again); about SETT and COBBLESTONE. OpenStreetMap Community Forum. <https://community.openstreetmap.org/t/tordanik-once-again-about-sett-and-cobblestone/85564>
- Hoffmann, M., Mock, M., & May, M. (2013). Road-quality classification and bump detection with bicycle-mounted smartphones. *Proceedings of the 3rd International Conference on Ubiquitous Data Mining*, 39–43. <https://dl.acm.org/doi/10.5555/2907177.2907186>
- Hossin, M., & Sulaiman, M. N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2), 01–11. <https://doi.org/10.5121/ijdkp.2015.5201>
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. <https://arxiv.org/abs/1704.04861v1>
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January*, 2261–2269. <https://doi.org/10.1109/CVPR.2017.243>
- IBM. (n.d.). *What is Computer Vision?* Retrieved May 24, 2024, from <https://www.ibm.com/topics/computer-vision>
- Jacob Gildenblat, & contributors. (2021). *PyTorch library for CAM methods*. GitHub. <https://github.com/jacobgil/pytorch-grad-cam>
- Jiang, P. T., Zhang, C. Bin, Hou, Q., Cheng, M. M., & Wei, Y. (2021). LayerCAM: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing*, 30, 5875–5888. <https://doi.org/10.1109/TIP.2021.3089943>
- Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4). <https://doi.org/10.1145/2382577.2382579>
- KeyLabs. (2024, February 16). *Semantic Segmentation vs Object Detection: A Comparison*. <https://keylabs.ai/blog/semantic-segmentation-vs-object-detection-a-comparison/>
- OpenStreetMap Contributors. (n.d.). *Key:surface*. OpenStreetMap Wiki. Retrieved April 23, 2023, from <https://wiki.openstreetmap.org/wiki/Key:surface>
- MapComplete. (n.d.). Retrieved May 21, 2024, from <https://mapcomplete.org/>

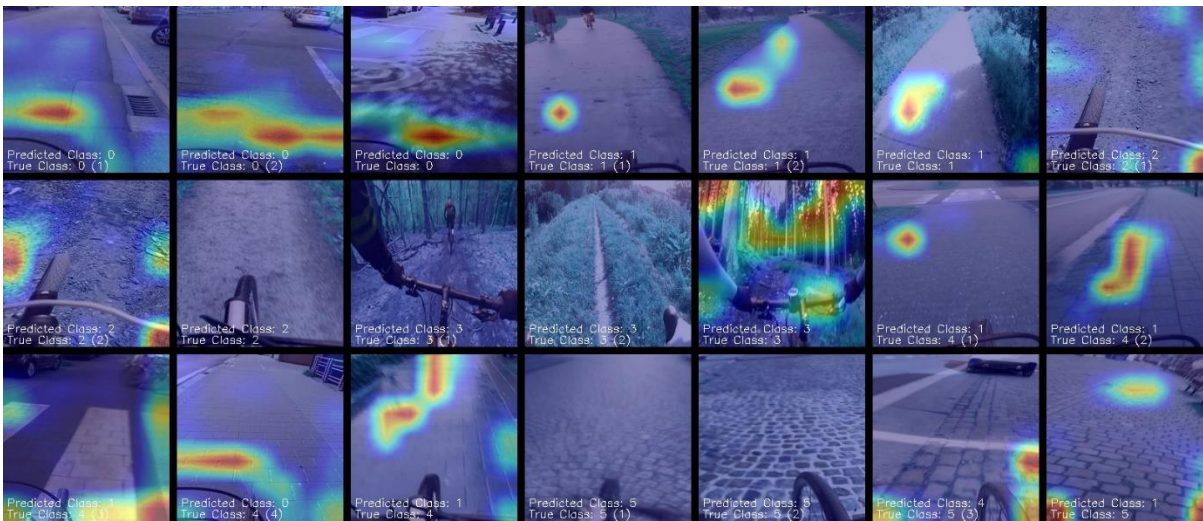
- Martinez-Noriega, E. J., & Yokota, R. (2023). Towards Real-time Formula Driven Dataset Feed for Large Scale Deep Learning Training. *IS and T International Symposium on Electronic Imaging Science and Technology*, 35(11). <https://doi.org/10.2352/EI.2023.35.11.HPCI-243>
- McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia Medica*, 22(3), 276. <https://doi.org/10.11613/bm.2012.031>
- Milleville, K. (2024, February 29). *Deep Learning for Image Segmentation*. Universiteit Gent.
- Ming Wen, L., & Rissel, C. (2008). Inverse associations between cycling to work, public transport, and overweight and obesity: findings from a population based study in Australia. *Preventive Medicine*, 46(1), 29–32. <https://doi.org/10.1016/J.YPMED.2007.08.009>
- Moraru, L., Obreja, C. D., Dey, N., & Ashour, A. S. (2018). Dempster-Shafer Fusion for Effective Retinal Vessels' Diameter Measurement. *Soft Computing Based Medical Image Analysis*, 149–160. <https://doi.org/10.1016/B978-0-12-813087-2.00008-7>
- Muhammad, M. B., & Yeasin, M. (2020). Eigen-CAM: Class Activation Map using Principal Components. *Proceedings of the International Joint Conference on Neural Networks*. <https://doi.org/10.1109/IJCNN48605.2020.9206626>
- Nielsen, M. (2019). *Neural networks and deep learning*. <http://neuralnetworksanddeeplearning.com/chap3.html>
- Nilsback, M.-E., & Zisserman, A. (2008, December). *Automated Flower Classification over a Large Number of Classes*. Indian Conference on Computer Vision, Graphics and Image Processing. <https://www.robots.ox.ac.uk/~vgg/publications/2008/Nilsback08/>
- Oja, P., Titze, S., Bauman, A., de Geus, B., Krenn, P., Reger-Nash, B., & Kohlberger, T. (2011). Health benefits of cycling: a systematic review. *Scandinavian Journal of Medicine & Science in Sports*, 21(4), 496–509. <https://doi.org/10.1111/J.1600-0838.2011.01299.X>
- OpenStreetMap Contributors. (n.d.). *OpenStreetMap*. Retrieved April 13, 2023, from <https://www.openstreetmap.org/about>
- Pereira, V., Tamura, S., Hayamizu, S., & Fukai, H. (2018). Classification of Paved and Unpaved Road Image Using Convolutional Neural Network for Road Condition Inspection System. *ICAICTA 2018 - 5th International Conference on Advanced Informatics: Concepts Theory and Applications*, 165–169. <https://doi.org/10.1109/ICAICTA.2018.8541284>
- Püttmann, L. (2023, February 9). *GPT and BERT: A Comparison of Transformer Architectures*. DEV Community. <https://dev.to/meetkern/gpt-and-bert-a-comparison-of-transformer-architectures-2k46>
- PyTorch. (n.d.). Retrieved March 2, 2024, from <https://pytorch.org/>
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. <https://doi.org/10.1109/CVPR.2018.00474>
- Sanghvi, K. (2020, May 8). *Image Classification Techniques*. Analytics Vidhya. <https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac>

- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). *Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization* (pp. 618–626). <http://gradcam.cloudcv.org>
- Shahul, E., & Aayush, B. (2023, August 30). *Hyperparameter Tuning in Python: a Complete Guide*. Neptune.Ai. <https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide>
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. <https://arxiv.org/abs/1409.1556v6>
- StreetComplete*. (n.d.). Retrieved May 21, 2024, from <https://streetcomplete.app/>
- Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 843–852).
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07(12)*, 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
- OpenStreetMap Contributors. (n.d.). *Tag:surface=sett*. OpenStreetMap Wiki. Retrieved May 25, 2024, from <https://wiki.openstreetmap.org/wiki/Tag:surface%3Dsett>
- Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *36th International Conference on Machine Learning, ICML 2019, 2019-June*, 10691–10700. <https://arxiv.org/abs/1905.11946v5>
- Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. *Proceedings of Machine Learning Research, 139*, 10096–10106. <https://arxiv.org/abs/2104.00298v3>
- Tsang, S.-H. (2018a, August 24). *Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification)*. Coinmonks. <https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7>
- Tsang, S.-H. (2018b, September 15). *Review: ResNet — Winner of ILSVRC 2015 (Image Classification, Localization, Detection)*. Towards Data Science. <https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8>
- TwoWheelsGood. (2018, March). *Surface type: gravel vs fine\_gravel vs compacted*. OpenStreetMap Community Forum. <https://community.openstreetmap.org/t/surface-type-gravel-vs-fine-gravel-vs-compacted/82578>
- Varona, B., Monteserin, A., & Teyseyre, A. (2020). A deep learning approach to automatic road surface monitoring and pothole detection. *Personal and Ubiquitous Computing, 24(4)*, 519–534. <https://doi.org/10.1007/s00779-019-01234-z>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems, 2017-December*, 5999–6009. <https://arxiv.org/abs/1706.03762v7>
- Verstockt, S., Slavkovikj, V., De Potter, P., Slowack, J., & Van De Walle, R. (2013). Multi-modal Bike Sensing for Automatic Geo-annotation Geo-annotation of Road/Terrain Type by Participatory Bike-

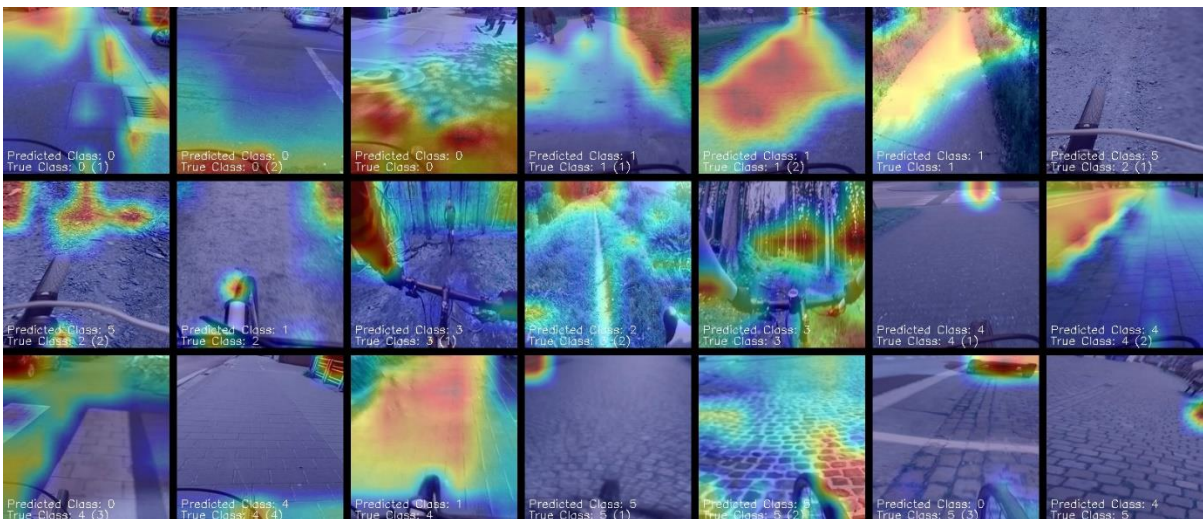
- sensing. In *2013 International Conference on Signal Processing and Multimedia Applications (SIGMAP)*.
- Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., & Hu, X. (2020). *Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks* (pp. 24–25). <https://github.com/haofanwang/Score-CAM>
- Winters, M., Brauer, M., Setton, E. M., & Teschke, K. (2010). Built environment influences on healthy transportation choices: bicycling versus driving. *Journal of Urban Health : Bulletin of the New York Academy of Medicine*, *87*(6), 969–993. <https://doi.org/10.1007/S11524-010-9509-6>
- Zanotti, M. (2020, June 30). *Transfer Learning in Image Classification: how much training data do we really need?* Towards Data Science. <https://towardsdatascience.com/transfer-learning-in-image-classification-how-much-training-data-do-we-really-need-7fb570abe774>
- Zhang, H., Sehab, R., Azouigui, S., & Boukhniifer, M. (2022). Application and Comparison of Deep Learning Methods to Detect Night-Time Road Surface Conditions for Autonomous Vehicles. *Electronics 2022, Vol. 11, Page 786, 11*(5), 786. <https://doi.org/10.3390/ELECTRONICS11050786>
- Zvornicanin, E. (2023, March 16). *Relation Between Learning Rate and Batch Size* . Baeldung. <https://www.baeldung.com/cs/learning-rate-batch-size>



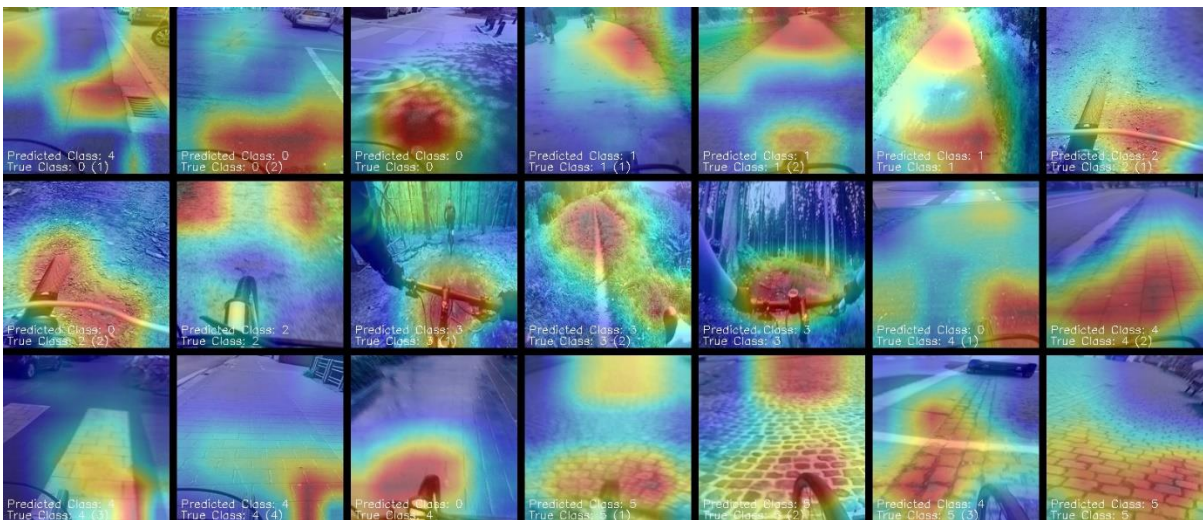
## 8 APPENDICES



Appendix 1 Saliency map for VGG16 with the road-based dataset

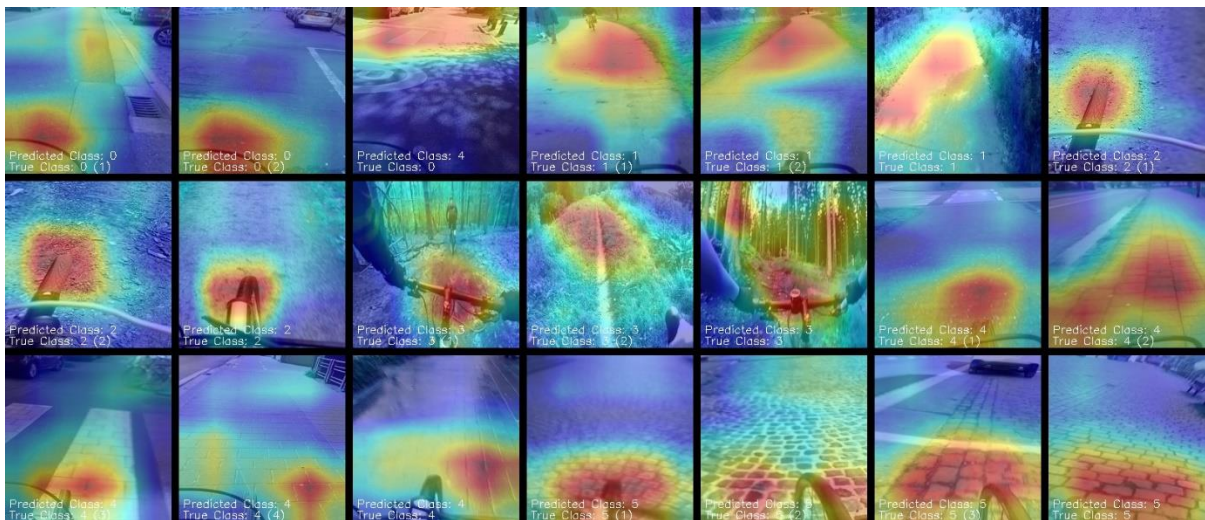


Appendix 2 Saliency map for Vision Transformer with the road-based dataset

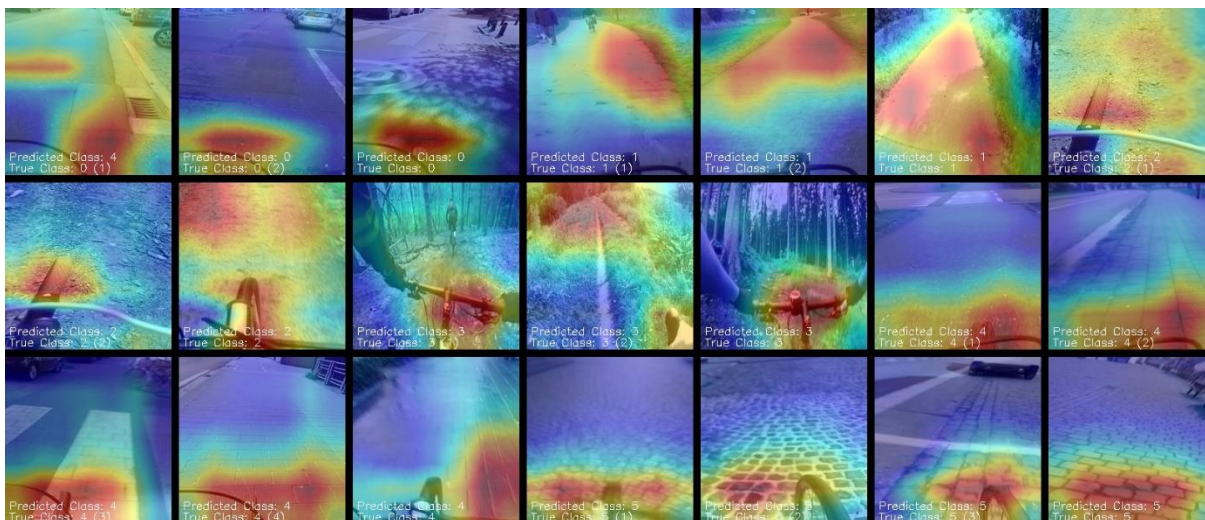


Appendix 3 Saliency map for GoogLeNet with the road-based dataset





Appendix 4 Saliency map for DenseNet with the road-based dataset



Appendix 5 Saliency map for MobileNet V2 with the road-based dataset