CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering

Road Surface Condition Monitoring System

Systém sledování stavu povrchu vozovky

Diploma Thesis

Author:	**Bc. Soňa Drocárová**
Supervisor:	**Ing. Adam Novozámský, Ph.D.**
Academic year:	2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Drocárová** Jméno: **Soňa** Osobní číslo: **486256**

Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**

Zadávající katedra/ústav: **Katedra matematiky**

Studijní program: **Aplikované matematicko-stochastické metody**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém sledování stavu povrchu vozovky

Název diplomové práce anglicky:

Road Surface Condition Monitoring System

Pokyny pro vypracování:

[1] Seznamte se s klasifikací stavu povrchu vozovky pomocí moderních technik zpracování obrazu. Prozkoumejte různé přístupy a vyberte alespoň dvě metody, které budete v práci dále zkoumat.
[2] Vyhledejte v literatuře volně dostupné soubory dat používané při trénování algoritmů a vytvořte nový podobný dataset. Prozkoumejte možnosti využití předtrénovaných neuronových sítí.
[3] Natrénujte vybrané metody na získaných datech a prozkoumejte jejich chování při různém nastavení parametrů.
[4] Definujte vhodné metriky pro hodnocení kvality modelu. Na základě odpovídajících metodik hodnocení kvality klasifikace porovnejte vybrané metody.

Seznam doporučené literatury:

[1] R. C. Gonzalez, R. E. Woods, Digital Image Processing (4th ed.). Pearson, 2018.
[2] I. Goodfellow, Y. Bengio, A. Courville, Deep learning. MIT press, 2016.
[3] D. V. Godoy. Deep Learning with PyTorch Step-by-Step: A Beginner's Guide. Independently published, 1 2022.
[4] Ma, Yao et al. "Current Non-Contact Road Surface Condition Detection Schemes and Technical Challenges." Sensors (Basel, Switzerland) vol. 22,24 9583. 7 Dec. 2022, doi:10.3390/s22249583
[5] K. Cordes, C. Reinders, P. Hindricks, J. Lammers, B. Rosenhahn and H. Broszio, "RoadSaW: A Large-Scale Dataset for Camera-Based Road Surface and Wetness Estimation," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 2022, pp. 4439-4448, doi: 10.1109/CVPRW56347.2022.00490.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Adam Novozámský, Ph.D. katedra matematiky FJFI

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.10.2023** Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **31.10.2025**

[illegible]	Masáková	[illegible]
Ing. Adam Novozámský, Ph.D.	prof. Ing. Zuzana Masáková, Ph.D.	doc. Ing. Václav Čuba, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

16.11.2023	[illegible]
Datum převzetí zadání	Podpis studentky

Acknowledgment:
I want to thank my supervisor, Ing. Adam Novozámský, Ph.D., for his guidance and helpful advice. Additionally, I express my gratitude to Porsche Engineering for providing the data for this thesis, and to my colleagues who were essential in the collection process, especially Bogdans for assisting me when needed. Finally, a big thanks to my boyfriend for his assistance with the language correction and emotional support.

Název práce:

Systém sledování stavu povrchu vozovky

Autor: Bc. Soňa Drocárová

Obor: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Adam Novozámský, Ph.D., Ústav teorie informace a automatizace, Pod Vodárenskou věží 4, 182 00, Praha 8

Abstrakt: Cieľom tejto diplomovej práce je preskúmať použitie techník počítačového videnia pre detekciu stavu povrchu vozovky v reálnom čase. Porovnávané boli dve architektúry konvolučných neurónových sietí, MobileNet a EfficientNet, z hľadiska ich výkonnosti a rýchlosti pri klasifikácii stavov povrchu na mokré, suché a snehom pokryté cesty. Modely boli trénované a porovnávané na verejne dostupnom datasete (RoadSaW) obsahujúcom presné anotácie z monitorovacieho senzoru výšky vodnej hladiny na vozovke, upraveného na klasifikáciu požadovaných stavov a na datasete vytvorenom z reálnych scén zachytených kamerou testovacieho vozidla, ktorý obsahuje takmer 55 000 obrázkov. V práci boli preskúmané rôzne trénovacie hyperparametre a techniky regularizácie na trénovanie modelov a následne boli vyhodnotené ich schopnosti generalizácie na predošle nepozorovaných dátach. Oba modely sa ukázali ako efektívne pre túto úlohu, pričom výsledky ukazujú, že MobileNet prekonal EfficientNet z hľadiska presnosti a rýchlosti spracovania, čo naznačuje potenciál pre jeho reálne aplikácie.

Klíčová slova: detekcia stavu povrchu vozovky, bezkontaktná detekcia, autonómne riadenie, počítačové videnie, hlboké učenie, konvolučné neurónové siete

Title:

Road Surface Condition Monitoring System

Author: Bc. Soňa Drocárová

Abstract: This thesis aims to investigate the application of computer vision techniques for real-time road surface condition (RSC) detection. Two convolutional neural network architectures, MobileNet and EfficientNet, were evaluated for their performance and inference speed in classifying surface conditions into wet, dry and snow covered roads. The models were trained and evaluated on a public dataset (RoadSaW) containing precise annotations from a sensor monitoring the water film height, with modifications to fit the specific RSC classification task defined in this thesis and a created dataset of real-world driving scenarios recorded from the test vehicle camera consisting of almost 55 000 images. This work explored several training parameters and regularization techniques to optimize model performance and evaluated generalization capabilities on unseen data. Both models were shown to be effective in this task, with the results indicating that MobileNet outperformed EfficientNet in accuracy and processing speed, demonstrating potential for real-world applications.

Key words: road surface condition detection, non-contact detection, autonomous driving, computer vision, deep learning, convolutional neural networks

Contents

Introduction

Harsh weather, especially heavy rain or snow, adversely affects driver capabilities, vehicle handling, and the condition of road infrastructure. Several studies have shown that these conditions negatively influence the roadway capacity, speed, and density of the traffic and increase the risk of traffic accidents [1]. Monitoring surface conditions offers valuable information that could be used as a reference to understand vehicle braking behavior, potentially decreasing the risk of dangerous accidents. Of critical importance in this regard is the road friction coefficient, which impacts vehicular braking control. Low-friction road surfaces pose significant risks of high-side slip accidents, where the brakes become less effective, increasing the possibility of traffic accidents. Various factors influence this coefficient, including vehicle characteristics, road surface, and environmental variables. Among these, weather conditions such as rainfall, snow, and ice have the most substantial impact, causing road surfaces to become slippery, leading to sudden reductions in friction coefficients. Table 1 shows the change in the estimate of the road friction coefficient based on the different road conditions.

Table 1: Friction coefficient for different road surface conditions according to [2].

Road Surface Condition	Dry	Damp	Wet	Black Ice	Snow
Friction coefficient estimate	1 ~ 0.80	~ 0.75	~ 0.65	~ 0.50	~ 0.40

Road surface condition (RSC) sensing is an important tool for detecting states that affect the estimation of friction coefficients and provides key information on vehicle braking dynamics [2]. Therefore, both active safety systems and self-driving cars could greatly benefit from the ability to predict these conditions in real-time [3].

Several sensors already exist for this purpose that are able to determine RSC in both contact and non-contact forms. Although precise, contact sensors face limitations due to their short service life, high installation and maintenance requirements, and the inability to provide real-time measurements. Non-contact methods such as infrared spectroscopy, optical polarization, radar detection, and computer vision meet the portability and real-time data support demands. Despite the fact that many sensors for RSC detection are already available, utilizing a front-facing in-vehicle camera remains the most cost-effective and readily accessible option, since it often comes as part of basic vehicle hardware [2].

The aim of this thesis is to train a fast and accurate computer vision model to detect road surface conditions suitable for use in a test vehicle which is a Porsche Cayenne equipped with the ZED 2 camera system installed behind the rear-view mirror. The first Chapter gives a brief overview of the different non-contact RSC detection methods and further focuses on the previous work in the computer vision approach to this task. Chapter 2 provides an introduction to the concept of machine learning with an emphasis on classification task metrics and common regularization techniques, while Chapter 3 delves deeper into the specifics of artificial neural networks, mainly the convolutional neural network architectures, which are widely used in computer vision. Chapter 4 presents the approach to RSC detection taken in this thesis, providing information about the dataset created from the test vehicle's recordings used to examine the model performance in real-world conditions. The final Chapter describes the experiments and evaluation of computer vision models trained for the purpose of RSC detection. These were trained and tested on public recordings and custom data created for the objective of this thesis using recordings from the test car camera. The classification metrics showed strong performance for both trained models, while the assessment of prediction speed demonstrated their ability to operate in real-world scenarios.

Chapter 1

Road Surface Condition Detection

As already mentioned in the Introduction, the condition of the road surface strongly influences the friction coefficient. These conditions can be examined through contact or non-contact measurements. According to [2] the precision of contact sensors for measuring RSC in addition to their high maintenance and installation requirements cannot offer real-time measurements, which is critical for autonomous driving and active safety systems.

Given the focus of this thesis on computer vision, a short description of the main non-contact RSC detection methods will be presented in this Chapter mainly emphasizing the computer vision approach and the previous researches in this field. Some of the common non-contact road surface condition detection approaches are represented in Figure 1.1 and include:

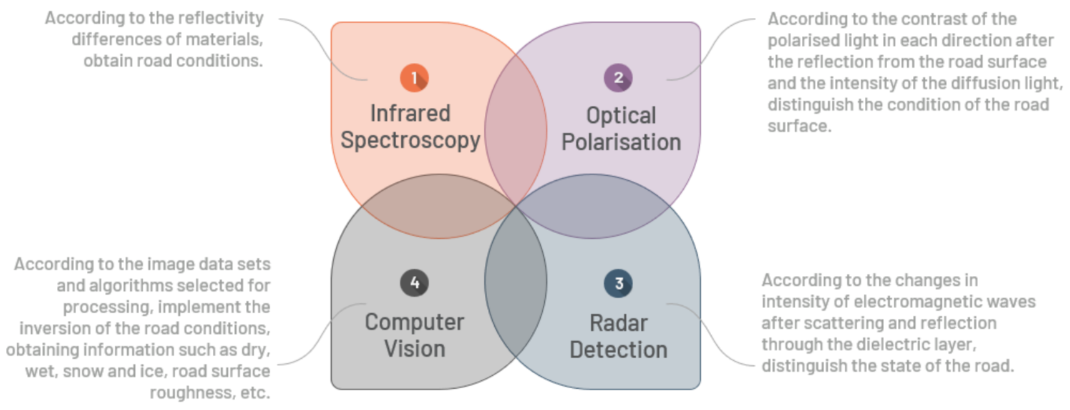

According to the reflectivity differences of materials, obtain road conditions.
1
Infrared Spectroscopy
2
Optical Polarisation
According to the contrast of the polarised light in each direction after the reflection from the road surface and the intensity of the diffusion light, distinguish the condition of the road surface.
According to the image data sets and algorithms selected for processing, implement the inversion of the road conditions, obtaining information such as dry, wet, snow and ice, road surface roughness, etc.
4
Computer Vision
3
Radar Detection
According to the changes in intensity of electromagnetic waves after scattering and reflection through the dielectric layer, distinguish the state of the road.

Figure 1.1: Diagram of common non-contact RSC detection methods [2].

- **Infrared Spectroscopy:** This technique measures the absorption of infrared (IR) frequencies by the sample placed in the path of the IR beam. The main idea is that different chemical functional groups in the sample absorb different characteristic wavelengths of IR radiation. Based on this knowledge, the presence or absence of these groups can be identified [4]. The various reflectancies of water, snow and ice make this approach suitable for RSC detection, where several characteristic wavelengths sensitive to these different water states are used as the basis for discrimination. Several IR spectroscopy-based sensors developed for this purpose are described in [2], both vehicle-mounted and stationary. Vehicle-mounted sensors can be suspended on either side of the car as well as on the roof and provide real-time RSC information. Stationary sensors can be installed at different heights and angles above the road. These tend to have better performance but come at higher cost. Although IR spectroscopy is a fast and highly accurate RSC detection approach, capable of providing detailed information on road conditions, its accuracy is strongly impacted by different light sources and backgrounds. In addition to this shortcoming, the sophisticated technology involved leads to higher costs of these systems.

- **Optical Polarization:** This approach relies on the use of a photodetector that measures the intensity of scattered and reflected light. The polarization parameters are then used to distinguish between dry and waterlogged, as well as wet and dry surface conditions. While this principle is easy to implement, due to the difference in refractive indices of ice and water being very small, a high degree of precision in the installation angle of the detector is required. Among its main drawbacks is also the fact that environmental changes have a large impact on RSC detection results [2].

- **Radar Detection:** Some of the commonly used radar-like technologies for autonomous driving capable of RSC detection are millimeter wave radar and LiDAR (Light Detection and Ranging) [2]. LiDARs use one or more laser beams to scan the field of view of the vehicle, resulting in a three-dimensional point cloud of the scanned environments and the intensities corresponding to the reflected laser energies. This output is commonly used in autonomous driving for tasks such as object detection, classification, tracking, and intention prediction. Although LiDARs have proven to be effective in various tasks, their functionality is limited in adverse weather conditions [5], which makes them less suitable for RSC detection. Millimeter wave radar utilizes radio waves in the millimeter wave spectrum to identify different conditions based on changes in intensity after scattering and reflection through the layer of water, ice, or snow. This radar technology outperforms LiDAR in challenging weather conditions; however, it falls short in terms of resolution [2].

- **Computer Vision:** Deep learning algorithms have been successfully applied in various classification tasks. Many vehicles come equipped with a monocular front camera as part of their basic hardware. Using the information from these cameras, computer vision methods have the ability to explore the texture and other characteristics of the surface to determine road conditions. Existing approaches for RSC detection using convolutional neural networks were able to differentiate between wet, dry, and snowy conditions, in addition to providing a distinction between different types of road surface [6]. Computer vision methods utilize existing sensors to predict road surface conditions, offering a cost-effective solution. However, this approach is dependent on the quality of the data set, which impacts the accuracy of the predictions [2]. Nevertheless, the extraction of information about the surroundings of the vehicles from the front-facing camera is a common computer vision task in autonomous driving applications [7].

1.1 Computer Vision Datasets

As already mentioned, computer vision methods depend on the datasets used for training, this section is going to present various public computer vision datasets employed in automotive tasks, which have been or have the potential to be used for training machine learning models for RSC detection.

BDD100K

This dataset consists of 100 000 driving videos (40 seconds each) recorded during more than 50 000 rides using a camera, collected in various regions of the United States, including New York and the San Francisco Bay area. Provided are also annotations for different computer vision tasks including image classification, object detection, semantic segmentation, instance segmentation, lane detection, drivable area segmentation, and more (see examples in Figure 1.2). These recordings were taken under various conditions, such as different times of the day (including nighttime recordings) but also in diverse weather conditions (rain, snow, clear, partly cloudy or overcast) [8].

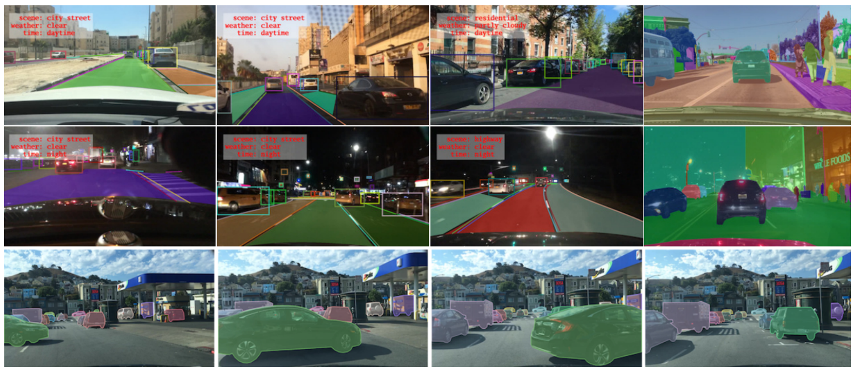


Figure 1.2: BDD100K dataset sample images with annotations [8].

KITTI

This dataset was recorded in Karlsruhe, Germany, using various sensors, including color and grayscale stereo cameras mounted on the roof of a vehicle. The images contain 6 hours of real-world traffic scenarios, including highways, inner cities, or rural areas. It provides evaluation benchmarks for computer vision tasks such as optical flow, semantic and instance segmentation, and 3D object detection visualized in Figure 1.3, as well as many more [9]. However, KITTI does not provide any information on the weather conditions during recording or any indication that there was any diversity in the conditions on the road surface during the recorded scenarios, making it less ideal for RSC detection task.

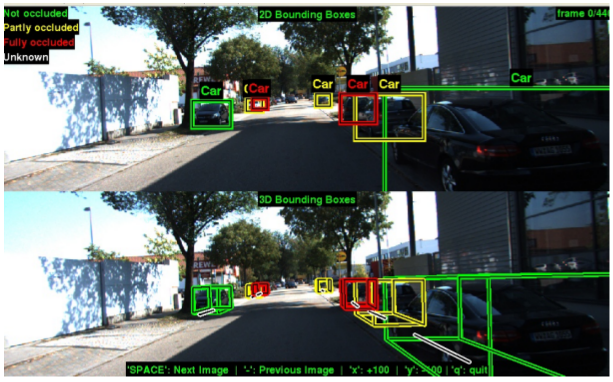


Figure 1.3: Examples of images with annotations from KITTI dataset [9].

RobotCar

RobotCar consist of over 100 drives of the same route in Oxford, UK. Data were collected from May through December, capturing different weather conditions, such as light and heavy rain, direct sunlight, and snow, using 6 cameras along with LiDAR and GPS. The dataset contains raw data with tags provided for different weather, road, or GPS conditions [10]. Despite its considerable size and the wide variety in lighting and weather conditions, a drawback of RobotCar with respect to RSC detection could be the limited number of different types of roads. Since the information is captured on one particular route that is traversed many times, training a computer vision model on these data might result in the model being only reliable on the same road.

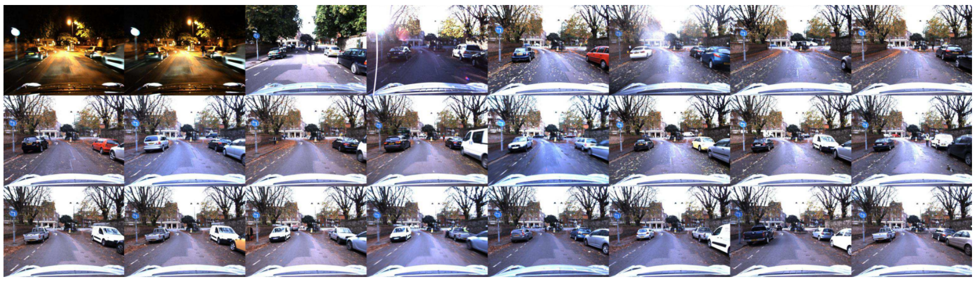


Figure 1.4: Example of image data from RobotCar dataset corresponding to one particular spot in different lighting and weather conditions [10].

CityScapes

This collection of driving videos was recorded over a period of several months in around 50 cities in Germany. With various annotations of over 30 classes, it primarily focuses on the semantic understanding of urban street driving scenes. CityScapes provides a benchmark for tasks such as semantic labeling or 3D vehicle detection (see Figure 1.5). However, all drives in CityScapes were recorded under good or medium weather conditions [11], making these data less suitable for RSC detection when used alone.

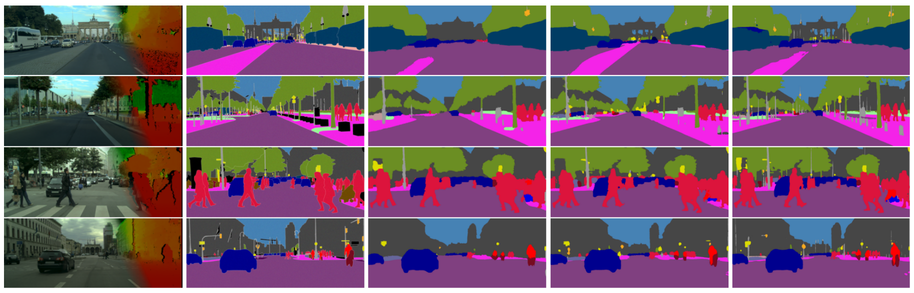


Figure 1.5: CityScapes examples of annotated data [11].

Ithaca365

Ithaca365 consists of 40 traversals of the same 15km route (in Ithaca, USA), with the objective of capturing diverse driving scenes, weather conditions (visualized in Figure 1.6), and traffic scenarios. In addition to images from four cameras, it provides LiDAR point clouds and GPS data. With 7000 annotated frames, it is intended for purposes such as amodal road and object segmentation, depth estimation, and 3D object detection [12]. Similarly to previously mentioned RobotCar, using this data alone could result in a model that is unable to generalize well on unseen roads.

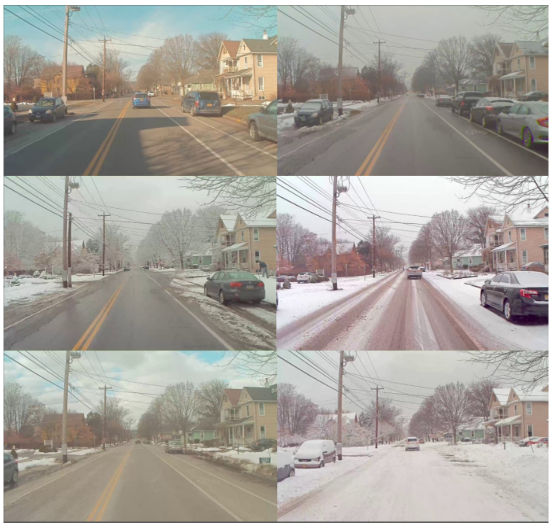


Figure 1.6: Example of images from Ithaca365 taken in the same location in varying weather conditions [12].

ACDC

ACDC or Adverse Conditions Dataset with Correspondences (shown in Figure 1.7) provides benchmarks for semantic, panoptic segmentation, and object detection. It consists of 4006 manually selected images split into classes such as fog, snow, rain, or nighttime, each paired with corresponding "normal conditions" images based on GPS information. Data were collected mainly by driving in urban settings, with occasional highways and rural areas in Switzerland, and captured on a camera mounted in front of the windshield in normal conditions and behind it during rain, fog or snow [13].

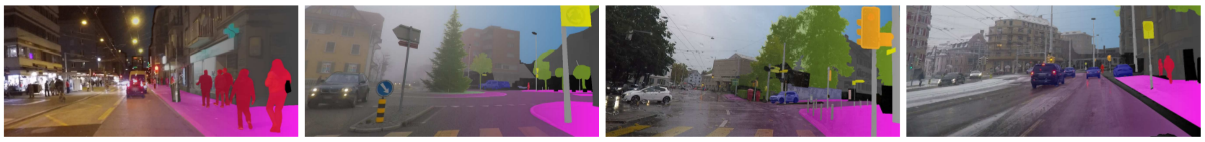


Figure 1.7: ACDC visualization [13].

RoadSaW

This dataset was created with the objective of developing apporaches for road condition estimation. It provides detailed annotations for road surface types (classes: *asphalt*, *concrete* and *cobblestone*) and conditions (classes: *dry*, *damp*, *wet*, and *very wet*). The ground truth provided for the wetness estimation is determined by the so-called Mobile Advanced Road Weather Information Sensor or MARWIS which is one of the previously mentioned sensors based on infrared spectroscopy. Accurate measurements of this sensor allow for fine segregation of wetness classes on the basis of the water film height.

The recordings for this dataset were taken over a span of 10 days using cameras mounted on vehicles (truck or car) along with the reference sensor. Wet conditions were simulated using sprinklers, and the final wetness classes are based on water film height thresholds that were established for each of the surface types. This controlled setting allowed the authors to create a balanced dataset with images split nearly equally into each of the classes. In addition to the four levels of RSC, a set of only two levels (*wet*, *dry*) is provided. The final resulting data is available in the form of region of interest cut-outs mapped to the bird's eye view of the captured images (see Figure 1.8) placed at four different distances from the vehicle that can be chosen as desired (7.5 m, 15 m, 22.5 m, 30 m) as well as 3 different cut-out sizes ranging in from 12.96 m^2 to 2.56 m^2 (labeled as small to large). The RoadSaW data set can also be merged with a compatible balanced snow coverage dataset RoadSC, which provides three types of snow: *fresh fallen*, *fully packed* and *paritally covered*. The authors in [6] state that the dataset is the first to provide accurate water film height measurements combined with road surface types.

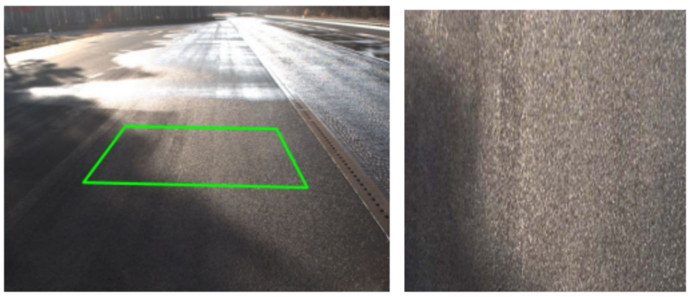


Figure 1.8: Example of the final road patch (rights) extracted from the image (left) which is contained in the RoadSaW dataset [6].

RSCD

RSCD offers manually created annotations for road surface friction (*dry*, *wet*,*water*,*fresh snow*,*melted snow*, *ice*), unevenness (*smooth*, *slight uneven*, *severe uneven*), and material (*asphalt*, *concrete*, *mud*, *gravel*) across 1 million image samples, gathered in Beijing between January and July 2022, spanning approximately 700 kilometers of roads, captured by a camera mounted on the vehicle's bonnet. These are provided in a form of 360 × 240 pixel road patches cut from the original images covering the road area through which the tires would pass (see Figure 1.9).

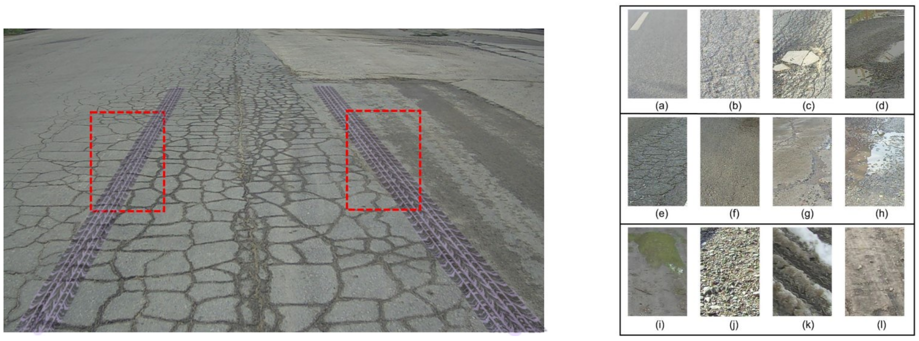


Figure 1.9: RSCD sample cut-out patches for (a)-(l) showing different classes (right) with the placement of the patch in terms of the original image (left) [14].

The above mentioned datasets vary in multiple aspects, such as the equipment utilized, the locations explored, or the recording conditions, making them valuable for a diverse range of tasks. Table 1.1 outlines the aspects that could be considered when selecting the most suitable ones for RSC detection. Also included for comparison is the dataset created for the purposes of this thesis used later in the experiment section.

Table 1.1: Comparison of public computer vision datasets employed in automotive tasks with regards to the RSC detection task.

Dataset	BDD100K	KITTI	RobotCar	CityScapes	Ithaca365	ACDC	RoadSaW	RSCD	Created Dataset
Images	12 000 000	14 999	N/A	5 000 (+2 000)	>680 000	8 012	20 244	1 000 000	54 952
Sequences	100 000	22	>100	50	40	N/A	>250	N/A	1 854
Diverse RSC	✓	N/A	✓	×	✓	✓	✓	✓	✓
Diverse Locations	✓	✓	×	✓	×	✓	×	✓	✓

1.2 Computer Vision Methods for RSC detection

The aim of this section is to explore the different computer vision strategies that have previously been used for the detection of RSC as documented in the literature. This analysis focuses on the models and various datasets constructed for this task, as well as on the variety of road condition classes targeted for recognition.

In [3] a two-step approach was developed to estimate road surface friction consisting of a convolutional neural network that classifies RSC (into classes *dry asphalt*, *wet/water*, *slush*, *snow/ice*) followed by patch segmentation and quantization with classification into three levels of road friction (low, medium, and high). For the first stage concerning RSC detection, four convolutional neural networks (three of which were custom and a SqueezeNet architecture) were tested along with a feature-based model. The data used in this article were collected from YouTube videos and manually annotated, resulting in 3 750 training and 1 550 test samples. As Figure 1.10 shows, the training and testing images were of the whole road scenes not just the road surface. From the models compared in [3], the best test accuracy was achieved with SqueezeNet (97.36%). Considering that the classification is based on entire road scenes, it suggests that it could introduce bias toward surrounding conditions, potentially misrepresenting the RSC. These could be instances, for example, with a clear sky after recent heavy rain or salted roads amidst snowy surroundings, failing to accurately reflect current road conditions.

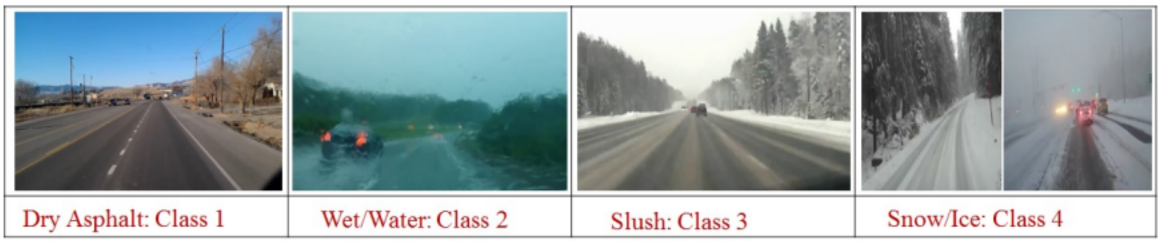


Figure 1.10: Example of training images in [3].

The best performing architecture in [3] was also employed in [15], where training data were extracted from previously described datasets (KITTI, Robotcar, Cityscapes, BDD100K) and some self-recorded data, resulting in 26 109 images. For the model to focus on the road ahead rather than the surroundings, trapezoidal regions of interest in front of the vehicle (shown in Figure 1.11) were used for training and evaluation. In addition to the detection of RSC, the type of road surface was also included, resulting in five classes (*cobblestone wet*, *cobblestone dry*, *asphalt wet*, *asphalt dry* and *snow*). This article also presents the results of pruning the network architecture to reduce its complexity. The best performance in terms of the $F1$ score for the pruned network is reported to be 95.3% on the test data.

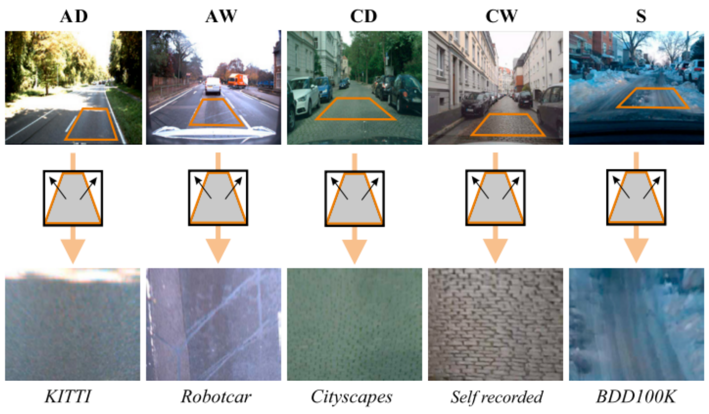


Figure 1.11: Preprocessing of images used in [15].

The evaluation of MobileNetV2 on RoadSaW data is presented in [6]. The test results are provided for each distance from the vehicle at medium patch size, as well as for all patch sizes at the 7.5 metre distance. From these, it is evident that the accuracy increases with the patch size and with the decreasing distance from the vehicle. The reported $F1$ score on the set with six classes is approximately between 80.32% and 92.65% and for twelve classes it ranges from 57.43% to 64.44%.

For RSCD data, multiple convolutional neural network architectures were examined in [14]. The highest accuracy was achieved by the EfficieneNet-B0 model, reaching around 92% for accuracy, precision, and recall. These metrics were further improved by a fusion model that combines predictions from multiple partially overlapping regions for the final classification, resulting in an accuracy of 97.5% on the test set.

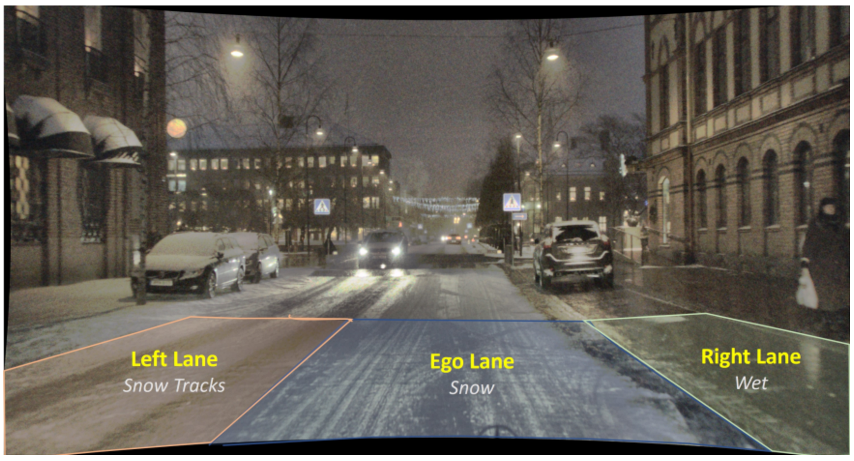


Figure 1.12: Example of data used in [16] for RSC detection in all lanes.

In [16] the RSC is detected not only in the ego-lane but also in the adjacent lanes (see Figure 1.12). Each of the three lanes gets assigned one class (*dry*, *wet*, *snow*, *snow tracks*) resulting in three predictions for each image. The dataset created for this purpose consisted of public benchmarks, as well as some self-recorded data with 25 714 images in total. The proposed network consisted of a backbone architecture (among the tested ones were EfficientNet, MobileNet, ResNet, Vision Transformer and Swin Transformer), three classification heads (used for RSC classification in each lane) and two regression heads (used for adjacent lane availability estimation). The results showed that in RSC detection, EfficientNet achieved the best performance metrics in all lanes except for the ego-lane with an F1 score on the test data of average 90.7% in the left and 90.8% in the right lane. In the ego-lane the best performing model was the Swin Transformer with an average F1 score of 90.9%.

A dataset constructed without error-prone manual annotations was created for the purposes of [17]. The images in this experiment were cropped to cover 1.5 m $\times$ 1.5 m region in front of the vehicle and transformed into a bird's eye view. Annotations for these samples in terms of road surface friction estimates were created using vehicle responses during driving. Around 3 000 images labeled in this manner were used to train a convolutional neural network predictor (with ResNet as the backbone) for friction estimation. The results of this trained predictor, provided on a test set of 800 images, showed a correlation coefficient between the predictions and the ground truth of 0.9824.

Other studies such as [18], [19], or [20], have taken an alternative approach to estimating RSC, focusing particularly on detection from the road-side cameras.

Chapter 2

Introduction to Machine Learning

Machine learning is used in many computer vision tasks and from the previous Chapter it is evident, that RSC estimation is not an exception. This Chapter provides a brief review of this field and its basic principles with a focus on evaluation metrics and regularization techniques applicable to computer vision for classification tasks. This discipline combines various diverse fields of research that are visualized in

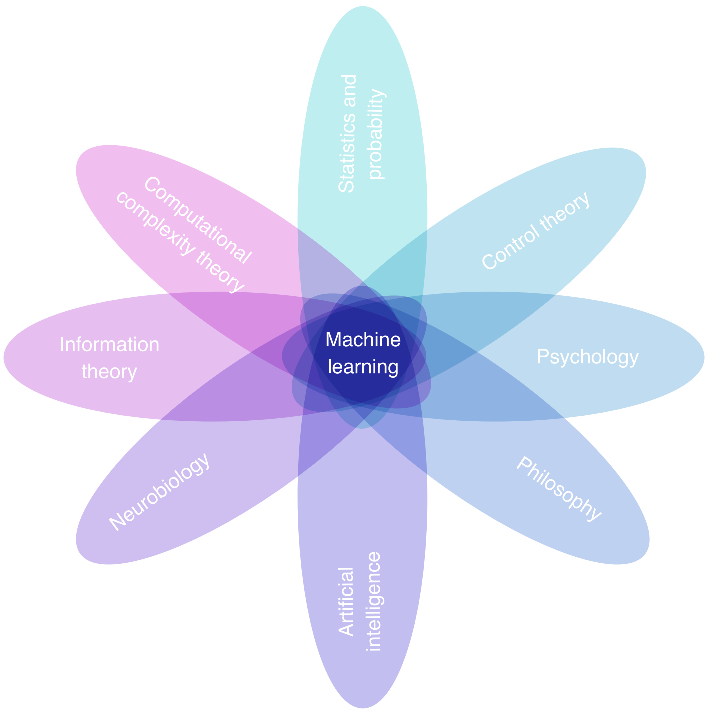


Figure 2.1: Various fields of research which are incorporated into machine learning [21].

Figure 2.1. It models algorithms based on empirical data, from which it pursues to extract rules and patterns. The main idea behind these algorithms is that they learn to perform given tasks based on acquired experience. A formal definition of learning by Tom M. Mitchell (1997) states: "A computer program is said to learn from experience E with respect to some class of tasks T and a performance measure P if its performance at tasks in T, as measured by P, improves with experience E." [22]. Some

of the most common machine learning tasks are *classification*, *regression*, *feature reduction*, *anomaly detection* or *denoising*.

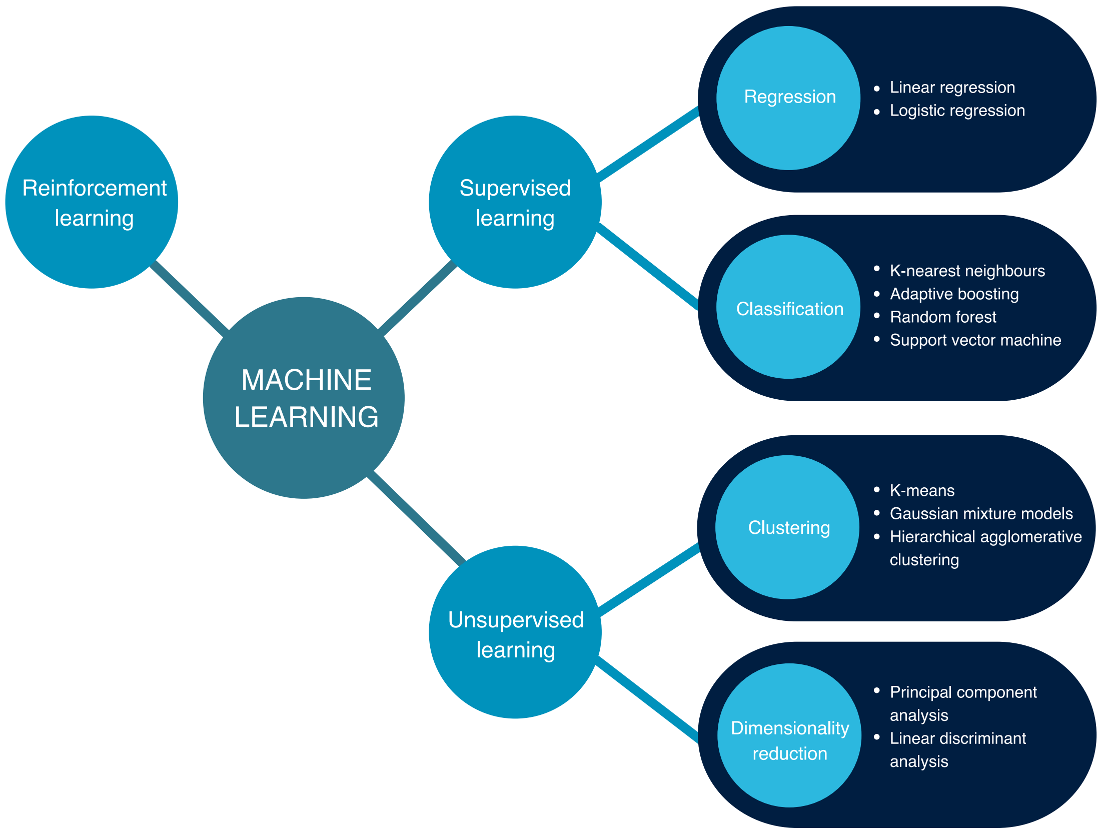


Figure 2.2: Main categories of machine learning algorithms based on experience.

The experience represents what kind of data we present the algorithm with and what it experiences [22]. Based on different learning techniques (experiences), machine learning can be divided into three main subcategories [23] (see Figure 2.2):

- **Supervised learning**: In this type of learning, examples of labels for input data are given to the computer. It is expected to associate features based on given knowledge and make decisions accordingly. Algorithms that experience this kind of data are, for example: linear or logistic regression, decision trees, naive Bayes classifier.
- **Unsupervised learning**: These techniques experience data given by features. Since no labels are provided, the training is done by finding similar patterns and grouping the data according to those patterns. Examples of this category include k-means clustering, probabilistic clustering, or hierarchical clustering.
- **Reinforcement learning**: Algorithms that use reinforcement learning maximize their performance by receiving feedback (called reinforcement signal or reward). Therefore, they do not experience a fixed dataset as the previously mentioned approaches. A reinforcement learning algorithm (also known as an agent) makes decisions, observes outcomes, and adjusts its strategy to improve future results. The agent interacts with the environment, taking actions based on its current state, which result in rewards that guide its behavior. The resulting behavior of the agent should aim to

maximize the long-term sum of the rewards. These reinforcement signals can also be delayed in time, meaning that they are not awarded immediately after taking the action which was crucial in gaining them [24].

2.1 Performance Evaluation

To evaluate the ability of the machine learning algorithm, a quantitative measure (P) is required to determine its performance. Since optimizing the performance measure is sometimes difficult, many of these algorithms act indirectly and optimize performance during training by minimizing a different cost function $J(\boldsymbol{\theta})$ [22]. Since RSC detection is a classification problem, only classification performance metrics are presented in this section.

In the case of binary classification, the evaluation consists of computing:

- *true positives* or *TP* - the count of observations classified as positive that are truly positive,
- *true negatives* or *TN* - the number of the count of observations classified as negative that truly belong to the negative class,
- *false positives* or *FP* - the count of observations classified as positive that are actually negative (misclassification of a negative instance as positive),
- *false negatives* or *FN* - the number of samples classified as negative that are actually positive (misclassification of a positive instance as negative)

In multi-class classification tasks, these counts are computed separately for each group C_i and constitute the so-called *confusion matrix* (see Figure 2.3). From these calculations, the actual performance metrics are computed.

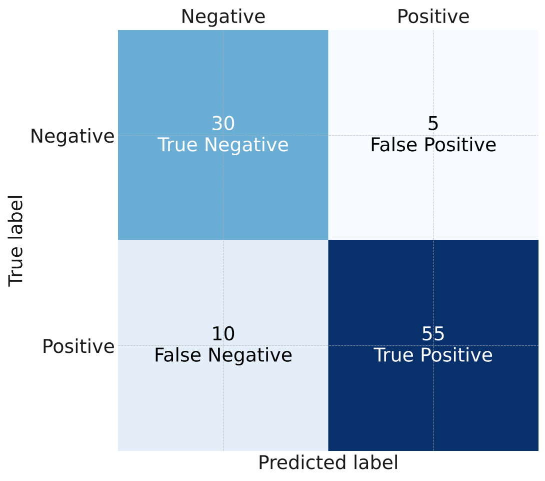


Figure 2.3: Example of a confusion matrix for binary classification.

Accuracy

Is one of the most common measures for classification. It reflects the overall effectiveness of a classifier. Given a problem with l classes, the accuracy Equation can be expressed as:

$$\text{Average Accuracy} = \frac{1}{l}\sum_{i=1}^{l}\frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i}. \tag{2.1}$$

Precision

Precision is a measure of agreement between the number of samples correctly classified as members of a given class and the total number of samples identified by the classifier as belonging to this group. For multi-class classification, it can be given as an average of the precisions in all classes:

$$\text{Precision} = \frac{1}{l}\sum_{i=1}^{l}\frac{TP_i}{TP_i + FP_i}. \tag{2.2}$$

Recall

Recall quantifies the effectiveness of the model in identifying the class label for a given class. The formula for the average recall of the model is given by Equation:

$$\text{Recall} = \frac{1}{l}\sum_{i=1}^{l}\frac{TP_i}{TP_i + FN_i}.$$

F-score

F-score calculates the harmonic mean between precision and recall and can be expressed as [25]:

$$\text{F-score} = \frac{(\beta^2 + 1)\cdot \text{ Precision } \cdot \text{Recall}}{\beta^2 \cdot \text{ Precision} + \text{Recall}}, \tag{2.3}$$

where β is a positive real factor used to adjust the weight of the importance of recall.

2.2 Regularization

One of the most challenging factors in machine learning is the algorithm's ability to perform well on previously unobserved input. To better understand its performance, a *test set* (which is independent of training data) is used to provide an unbiased evaluation of a final model fit. Another separate set, called *validation set* extracted as a subset of training data, is used to examine how the model handles previously unobserved input during the training before evaluating on test data. The main use of this dataset is to tune the so-called *hyperparameters* - model settings that are not adjusted during the learning process.

A term that describes the ability to give satisfactory results on previously unseen data is called *generalization*. While training a machine learning algorithm, a training error is calculated from the training data as well as a generalization (test) error - an error expected on new data. From these values, we can study the adequacy of the given algorithm by observing its ability to minimize:

- Training errors: The inability to adequately minimize the training error results in *underfitting*.

- The gap between test and training errors: If the gap becomes too large, *overfitting* occurs.

Examples of these unwanted behaviors of the models are depicted in Figure 2.4. The likelihood of over-fitting or under-fitting the model can be controlled by its capacity. This property represents the model's ability to fit different functions and can be controlled by adjusting the algorithm's hypothesis space. Although models with higher capacity are capable of solving more complex tasks, they are more prone to overfitting on simpler ones. One of the principles typically used in machine learning to help establish optimal model complexity is the *Occam's razor*, suggesting that among all competing hypotheses that fit the data equally well, the simplest should be preferred [22].

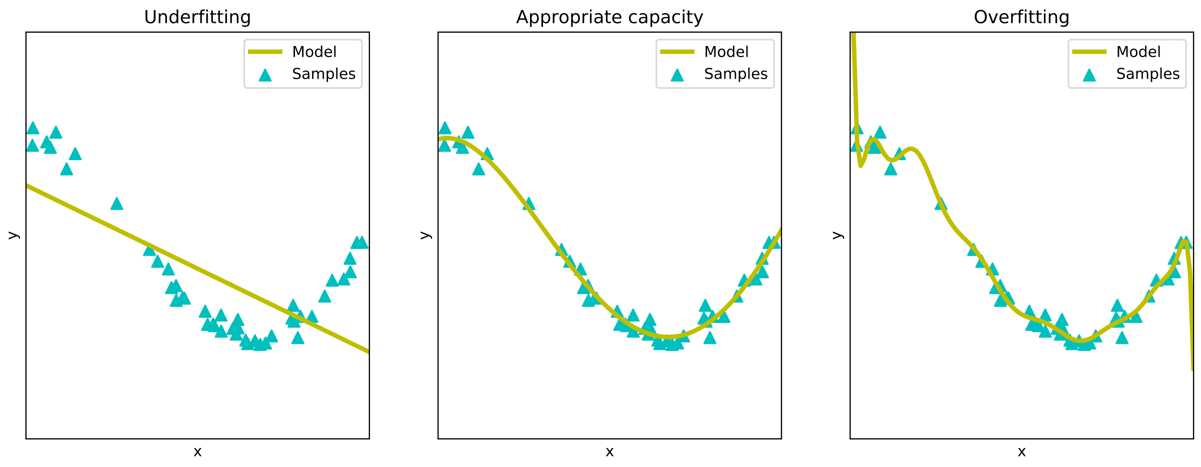


Figure 2.4: Visualisation of model behaviors if insufficient minimization of training errors (underfitting) or gap between training and test error (overfitting) occurs and a model with appropriate capacity.

Apart from this principle, several techniques have been developed to improve the machine learning model's performance on unseen data. They are collectively called *regularization* and some of the examples commonly used in deep learning are presented in the remainder of this chapter.

L^2 Parameter Regularization

This method, also known as *ridge regression*, is based on limiting the model's capacity to improve generalization. It penalizes the model's coefficient sizes by introducing a weight decay term into the objective function. Given the data $\mathbf{X}$, targets $\mathbf{y}$, coefficients (weights in neural networks) $\mathbf{w}$, and the objective function $J(\mathbf{w};\mathbf{X},\mathbf{y})$ that will be optimized, the L^2 regularized objective function $\tilde{J}(\mathbf{w};\mathbf{X},\mathbf{y})$ can be expressed as [22]:

$$\tilde{J}(\mathbf{w};\mathbf{X},\mathbf{y}) = J(\mathbf{w};\mathbf{X},\mathbf{y}) + \alpha\Omega(\mathbf{w}) = J(\mathbf{w};\mathbf{X},\mathbf{y}) + \frac{\alpha}{2}||w||_2^2.$$

This makes the smaller weights more preferable and helps mitigate overfitting [26]. Another similar but slightly less common regularization technique is L^1 regularization with a weight decay factor of $\Omega(\mathbf{w}) = ||w||_1 = \sum_i |w_i|$, which results in a more sparse solution.

Dataset Augmentation

Generalization of machine learning models can also be improved by using a larger training set. However, that is not always possible, so expanding the training set with artificially modified data may be a viable solution. This method has proven to be especially effective for classification tasks. The main goal of a classifier is to take a high-dimensional input and label it with a particular class. One of its main challenges is that it needs to become invariant to various transformations. Thus, creating new data for this task means using the applicable transformations on the existing data. For example, in the object classification task, the classifier needs to be robust to translation or rotation, which can be applied to the training set to create new samples. Another data augmentation technique that can also be used in some regression tasks is noise injection. Machine learning algorithms should be robust to a little noise, therefore including the original data with added noise should not make the problem unsolvable.

Early Stopping

Early stopping is typically used for the regularization of deep learning models. During the training process, the error calculated on training data should gradually decrease. In contrast, the validation error tends to start to rise at some point (see the example in Figure 2.5). This means that the model would be able to generalize better if the training did not continue beyond the point where the minimum validation error was reached. The early stopping algorithm terminates the training if the validation error has not decreased in a given number of training steps. The parameter that controls the maximum number of learning steps in which the validation error is allowed to increase is called *patience*. After reaching this point, the training is stopped and the model parameters are restored to the point where the validation error began to rise [22].

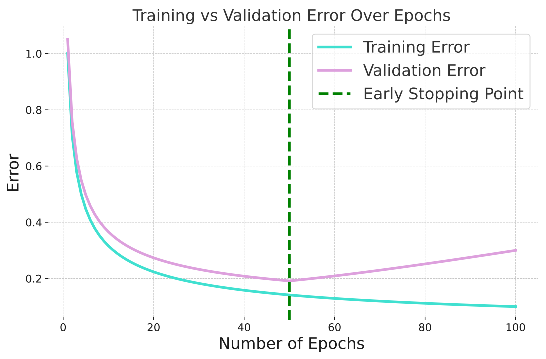


Figure 2.5: Example of the shape of training and validation error with the vertical line suggesting where early stopping would interrupt the training loop.

Dropout

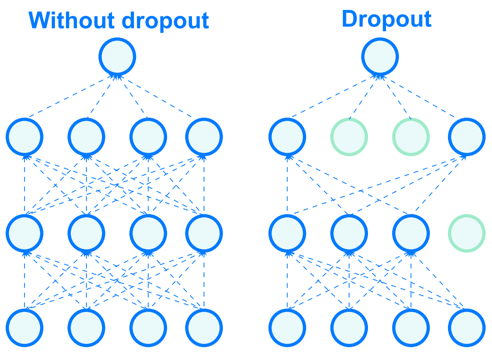


Figure 2.6: Illustration of dropout regularization technique [26].

Dropout is one of the regularization techniques used in deep learning. This method is relatively easy to implement and has many different variations, each better suited for different tasks. They are proven to be effective in reducing the model's sensitivity to input variations and noise. Primarily used in larger models with vast number of trainable parameters, the standard dropout method turns off randomly chosen neurons and their relevant connections in a given layer during training [26] which is illustrated in Figure 2.6. This simulates averaging the results of multiple reduced versions of the original network. As such, it enhances the network's robustness against overfitting by combining multiple predictions from substantial networks during the inference. In addition to that, by training the reduced versions of the network, the amount of computations is lower making the training faster [27].

Label Smoothing

This regularization technique is used to prevent both overfitting and overconfidence in neural networks. The output of classification tasks in neural network models is given in form of probability for each label:

$$\forall k \in \{1, \ldots, K\} : p(x) = \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_k)},$$

where x is the input sample and z_k are the unnormalized log-probabilities (logits). Label smoothing involves adjusting the label distribution for a given training example with the ground-truth label y by combining the original ground-truth distribution $q(k|x) = \delta_{k,y}$ (where $\delta_{k,y}$ is Dirac delta) with a fixed distribution over labels, based on a smoothing parameter ϵ. This leads to a mixed ground-truth distribution in the form of:

$$q'(k) = (1 - \epsilon)\delta_{k,y} + \frac{\epsilon}{K},$$

which prevents the largest logit from overshadowing all others. This technique has been shown to stabilize model performance by preventing overfitting and making it more adaptable through a more balanced confidence in predictions [28].

Chapter 3

Artificial Neural Networks

Most of the computer vision approaches to RSC detection mentioned in Chapter 1 rely on deep learning. This chapter aims to explain the basic concept of artificial neural networks, which are the foundation for these models. Next, convolutional neural networks and some of the state of the art architectures which were previously mentioned will be explored in more detail.

These machine learning algorithms are inspired by human biology. The main idea behind this approach is to construct a network of so-called neurons, which operate in parallel [29]. Unlike other machine learning techniques, artificial neural networks (ANNs) have the ability to learn the representation of given data, which becomes increasingly abstract in each layer. However, human input in the form of specifying the hyperparameters is still needed even for these learning algorithms.

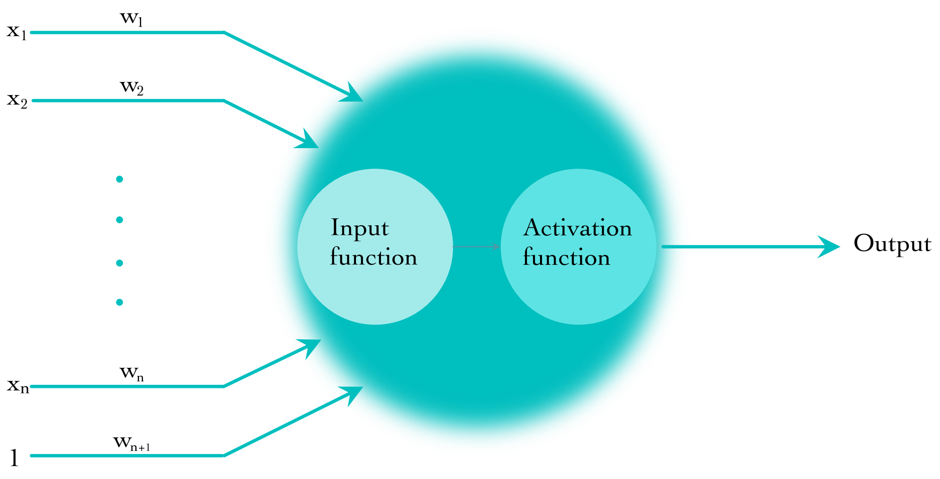


Figure 3.1: The composition of artificial neuron called perceptron.

3.1 Perceptron

One of the first artificial neuron models was the perceptron (see Figure 3.1). It is equipped to solve two-class classification problems where the data are linearly separable. The Equation for the separating hyperplane of the classes yields:

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{x} = 0, \tag{3.1}$$

where $\mathbf{x} = (x_1, x_2 \dots, x_n, 1)^T$ is a vector of inputs, $\mathbf{w} = (w_1, w_2, \dots, w_{n+1})^T$ is a vector of the corresponding weights, and f is called *input function*. When performing classification on linearly separable data with two classes C_1 and C_2, the aim is to find weights that satisfy the following condition:

$$\mathbf{w}^T\mathbf{x} \begin{cases} > 0 & \text{if } \mathbf{x} \in C_1 \\ < 0 & \text{if } \mathbf{x} \in C_2. \end{cases} \tag{3.2}$$

Let $\mathbf{w}(1)$ be the weight vector of arbitrary values, $\mathbf{x}(k)$ the pattern vector corresponding to step k and $\alpha > 0$ the *learning rate*. Then the training algorithm for step k of the perceptron is following [30]:

1. if $\mathbf{x}(k) \in C_1$ and $\mathbf{w}^T\mathbf{x}(k) \leq 0$, let $\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha\mathbf{x}(k)$
2. if $\mathbf{x}(k) \in C_2$ and $\mathbf{w}^T\mathbf{x}(k) \geq 0$, let $\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha\mathbf{x}(k)$
3. else, let $\mathbf{w}(k+1) = \mathbf{w}(k)$.

Afterwards, *activation function* g is used to determine the class membership of given data. In this case, the activation function is a thresholding function, which assigns the pattern to class C_1 if the thresholded output is 1 and to class C_2 for this value equal to -1.

Artificial neurons in multilayer networks work in the same way as perceptron, differing mainly in activation functions. Some of the most frequently used activations in artificial neurons are: sigmoid, rectifier linear unit (ReLU), hyperbolic tangent (see Figure 3.2) or in some instances softmax [30].

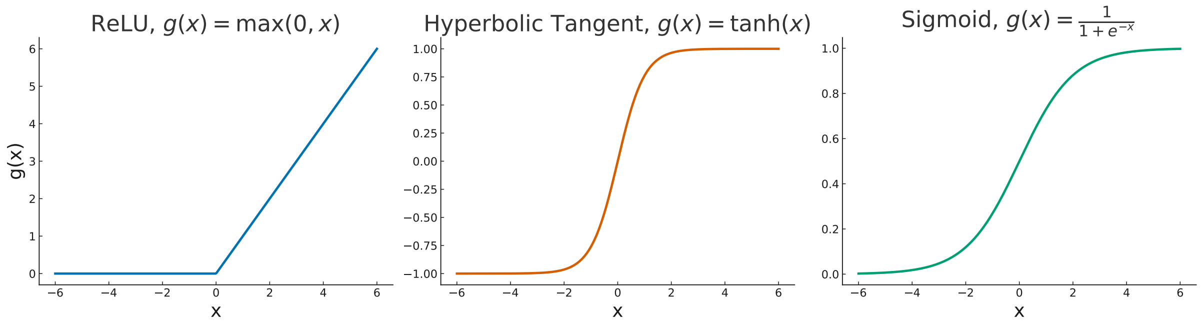


Figure 3.2: Activation functions used in artificial neuron structures.

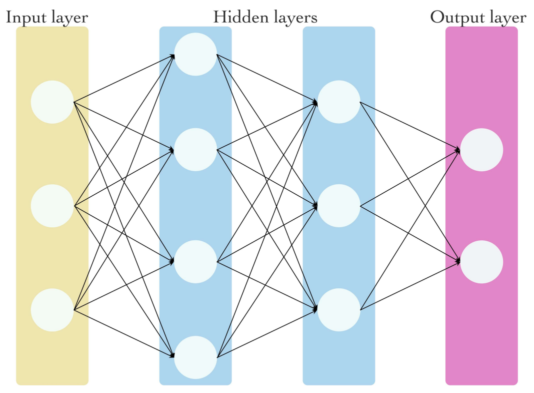
There are many different architectures of neural networks based on the way the network is connected. Some of the most well-known are, for example, feedforward networks, convolution, radial basis function networks, and many more [30].

3.2 Feedforward Neural Network

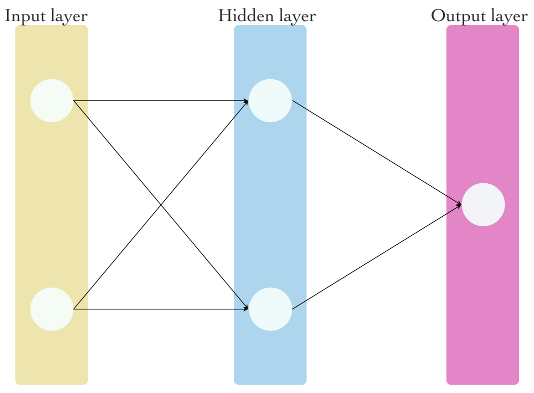
This network usually consists of several layers of neurons. The first one (*input layer*) is comprised of the same number of neurons as the dimensionality of the input vector. Data processed in the input layer are subsequently sent to the first *hidden layer*. Each layer receives the data, performs (previously described) operations, and sends the output to the next one [29]. The number of neurons can differ in each layer, but every neuron has only one output, which is connected to all neurons in the next layer to create a fully connected network. If there is only one hidden layer in the network, it is called *shallow*, while those with two or more are labeled *deep*. The number of layers of ANN is called its *depth* [30]. The last hidden layer is connected to the *output layer*, which provides the values used to determine the final decision [29]. Examples of deep and shallow feedforward neural networks are shown in Figure 3.3.

The training algorithm for these types of networks consists of four basic steps [30]:

1. input of the training data (also called forward pass),
2. classification of the data by network and the determination of the cost $J(\boldsymbol{\theta})$,
3. computation of required changes, that minimize its error, is carried out by passing it through the network (using the backpropagation algorithm),
4. the model weights are updated, and the process is repeated until the error reaches an acceptable level



(a)



(b)

Figure 3.3: Examples of feedforward (a) deep and (b) shallow neural network structures.

This neural network training process utilizes optimization algorithms to reduce the cost function $J(\theta)$ which can be expressed as an average of per-example loss functions L over training set samples [22]:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\hat{p}_{data}} L(f(\mathbf{x};\boldsymbol{\theta}), \mathbf{y}),$$

where $f(\mathbf{x};\boldsymbol{\theta})$ is the predicted output given input $\mathbf{x}$, $\hat{p}_{data}$ is the empirical distribution and $\mathbf{y}$ is the target output. Among some of the common optimization algorithms used for this purpose are:

- *Stochastic gradient descent (SGD)*: This optimizer is an extension of the basic gradient descent algorithm. The gradient of the loss in this case is computed from minibatches of m samples drawn i.i.d. from data generating distribution. Along with the values of the initial parameters $\boldsymbol{\theta}$, it requires a learning rate ϵ. First, a minibatch of inputs $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ and corresponding targets $\{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(m)}\}$ is sampled from the training set. A gradient is computed as follows: $\mathbf{g} = \frac{1}{m} \sum_i \nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), \mathbf{y}^{(i)})$ and based on this calculation, the weights and biases are updated $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\mathbf{g}$. The learning rate ϵ can be chosen by trial and error or by monitoring the leaning objective function curves during training.

- *SGD with momentum*: To accelerate the learning process of SGD, an algorithm using momentum is utilized. The momentum algorithm introduces a velocity $\mathbf{v}$ to the update rule, which acts as momentum of a particle in physics. The update rule changes from the previous SGD algorithm in the following manner:

$$\begin{aligned} \mathbf{v} &\leftarrow \alpha\mathbf{v} - \epsilon\mathbf{g} \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \mathbf{v}. \end{aligned}$$

  The hyperparameter of $\alpha \in [0, 1)$ determines the speed with which the effect of the past gradients decays.

- *Root mean square propagation (RMSProp)*: RMSProp belongs to the class of optimization algorithms with an adaptive learning rate. The gradient computations of the previous algorithms are followed by the update of variable $\mathbf{r}$ which accumulates the sum of squared gradients, is used during the parameter update in the following manner:

$$\begin{aligned} \mathbf{r} &\leftarrow \rho\mathbf{r} + (1-\rho)\mathbf{g} \odot \mathbf{g} \\ \Delta\boldsymbol{\theta} &= -\frac{\epsilon}{\sqrt{\delta + \mathbf{r}}} \odot \mathbf{g} \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}, \end{aligned}$$

  where δ is a small constant that prevents division by zero, ρ is a decay rate and $\cdot$ stands for element-wise multiplication.

- *Adam*: Deriving its name from the phrase "adaptive moments", this form of optimization not only utilizes the momentum algorithm but also involves an adaptive learning rate. In this case, the update of weights in step t includes updating the first order moment $\mathbf{s}$ (momentum) and the second order moment $\mathbf{r}$ in the form of [22]:

$$\begin{aligned} \mathbf{s} &\leftarrow \rho_1\mathbf{s} + (1-\rho_1)\mathbf{g} \\ \mathbf{r} &\leftarrow \rho_2\mathbf{r} + (1-\rho_2)\mathbf{g} \odot \mathbf{g}. \end{aligned}$$

  This is followed by applying corrections to these moments to account for their initialization at the origin yielding:

$$\begin{aligned} \hat{\mathbf{s}} &\leftarrow \frac{\mathbf{s}}{1-\rho_1^t} \\ \hat{\mathbf{r}} &\leftarrow \frac{\mathbf{r}}{1-\rho_2^t} \end{aligned}$$

  and finally the parameters are updated using these calculations as:

$$\begin{aligned} \Delta\boldsymbol{\theta} &= -\epsilon\frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta} \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}. \end{aligned}$$

3.3 Convolutional Neural Network

The main difference between convolutional neural networks (CNN) and feedforward networks is that they use convolution as an input function in certain layers. This allows the input of these networks to have an image format or similar grid-like form. Instead of previously extracted features, these ANNs learn the patterns directly from given images. Therefore, these types of networks are well suited for image processing applications [30]. Aside from the input and output layer, a CNN is commonly comprised of

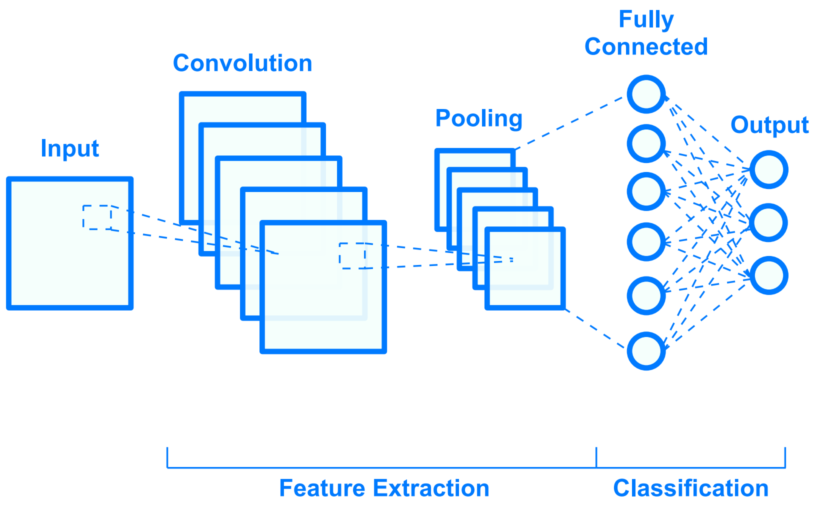


Figure 3.4: An example of CNN structure.

three types of layers - *convolutional*, *pooling* and *fully connected*. This structure is shown in Figure 3.4 which depicts:

- **Convolutional layers** use kernels, which are convolved with the input. The kernel is the same depth as the input, has low spatial dimensionality, and is applied to the whole image. Same as in previously described ANNs, an activation function is applied to the outcome of convolution. An example of how this layer performs the operation on an image is shown in Figure 3.5 [31]. The commonly used definition of convolution in many neural network libraries for two-dimensional image $\mathbf{F}$ with kernel $\hat{\mathbf{K}}$ can be represented by the following Equation:

$$\hat{\mathbf{G}}_{i,j} = (\mathbf{F} * \hat{\mathbf{K}})_{i,j} = \sum_{m,n} \mathbf{F}_{i+m,j+n} \hat{\mathbf{K}}_{m,n},$$

  which is also called cross-correlation. In computer vision tasks, the input to CNN architecture is generally an image which can also be thought of as a three-dimensional tensor with two indices in the spatial coordinates and one into the different color channels. Thus, the convolution Equation for convolving the input data $\mathbf{F}_{i,j,k}$ (i being the channel, j row and k column index) with a four-dimensional kernel tensor $\hat{\mathbf{K}}_{i,j,k,l}$ (where i represents the output channel index, j the input channel index and k, l spatial dimensions) is defined as:

$$\hat{\mathbf{G}}_{i,j,k} = \sum_{l,m,n} \mathbf{F}_{l,j+m-1,k+n-1} \hat{\mathbf{K}}_{i,l,m,n}.$$

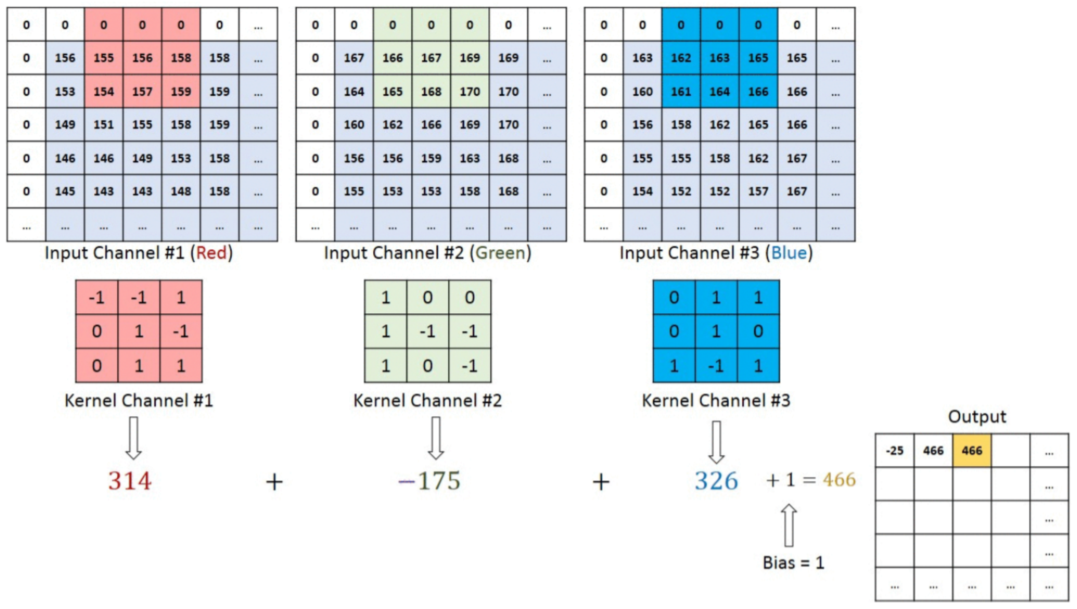


Figure 3.5: The convolution of input image with $3 \times 3 \times 3$ kernel [32].

To reduce computational costs, convolution can be computed in a way that skips over some of the kernel positions, yielding the following representation of this operation:

$$\hat{\mathbf{G}}_{i,j,k} = \sum_{l,m,n} \mathbf{F}_{l,(j-1)s+m,(k-1)s+n} \hat{\mathbf{K}}_{i,l,m,n},$$

where the scalar value s is called a *stride* [22].

- **Pooling layers** are used to reduce the spatial dimensionality of the input feature map. The most common are *max-pooling layers* [31]. This method of pooling is done by moving the kernel along the input by the given stride and returning the maximum value of the covered portion of the data.

- The **fully connected layer** functions in the same way as a typical ANN. The output of the previous layer is flattened and a feedforward neural network with a soft-max activation function is used for classification [32].

Having described the basic principles of CNNs, the remainder of this chapter will focus on some of the CNNs structures mentioned in Chapter 1. For the task of RSC detection, along with numerous other computer vision applications, it is essential that the model works in real time and on computationally limited platforms. Therefore, the models that tend to be used for this task prioritize these requirements.

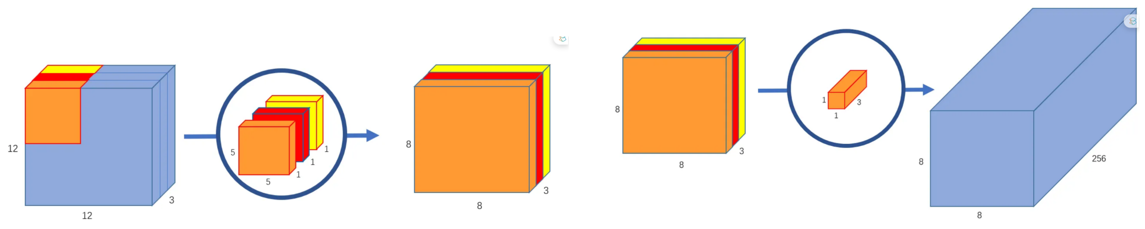
MobileNet

This architecture was created with the aim of minimizing the size and computational requirements making it well suited for mobile devices and embedded systems (such as cameras in autonomous vehicles or drones).

The main building blocks of this network architecture are *depthwise separable* convolutions. These layers consist of depthwise and pointwise convolutions (see example of both in Figure 3.6). The output feature map $\hat{\mathbf{G}}$ of the convolution of the depthwise convolutional kernel $\hat{\mathbf{K}}$ with the i-th channel of the feature map $\mathbf{F}$ is defined as:

$$\hat{\mathbf{G}}_{i,j,k} = \sum_{l,m} \mathbf{F}_{i,j+l-1,k+m-1} \hat{\mathbf{K}}_{i,l,m}.$$

However, this operation does not combine the input channels to create new features. For this reason, the output has to be combined with the 1×1 (pointwise) convolution creating a linear combination of the output of the depthwise layer. This operation is efficient compared to standard convolution, which has a computational cost of $D_{K_h} \cdot D_{K_w} \cdot M \cdot N \cdot D_{F_h} \cdot D_{F_w}$, with $D_{K_h} \times D_{K_w}$ representing the kernel size, $D_{F_h} \times D_{F_w}$ the size of the feature map, M the number of input channels and N output channels. For this operation, the cost is reduced to $D_{K_h} \cdot D_{K_w} \cdot M \cdot D_{F_h} \cdot D_{F_W} + M \cdot N \cdot D_{F_h} \cdot D_{F_w}$ which corresponds to the sum of depthwise and pointwise convolutions, resulting in a computational reduction of $\frac{D_{K_h} \cdot D_{K_w} \cdot M \cdot D_{F_h} \cdot D_{F_W} + M \cdot N \cdot D_{F_h} \cdot D_{F_w}}{D_{K_h} \cdot D_{K_w} \cdot M \cdot N \cdot D_{F_h} \cdot D_{F_w}} = 1/N + 1/D_K^2$ [33].



(a) (b)

Figure 3.6: Example of depthwise (a) and pointwise (b) convolution [34].

The idea of depthwise separable convolutions was further explored by other versions of the first MobileNetV1 architecture. MobileNetV2 introduces a linear bottleneck and inverted residual structure (see Figure 3.7) to decrease the number of operations and memory required. The bottleneck residual block consists of 1×1 convolution with the activation function ReLU6 (ReLU with an upper limit of six) that takes as an input feature map of size $D_{F_h} \times D_{F_w} \times M$, depthwise convolution with stride s again followed by ReLU6 which transforms the output of the previous layer with size $D_{F_h} \times D_{F_w} \times (tM)$ (t is an expansion factor), finally a linear 1×1 convolution layer is utilized transforming the $\frac{D_{F_h}}{s} \times \frac{D_{F_w}}{s} \times (tM)$ sized output of depthwise convolution to $\frac{D_{F_h}}{s} \times \frac{D_{F_w}}{s} \times (N)$. It also includes residual (or skip) connections that connect the input and the output of this block by addition but only if they match in the number of channels. The computational cost of this block is $D_{F_h} \cdot D_{F_w} \cdot M \cdot t \cdot (M + D_K^2 + N)$ which, in comparison to the cost of depthwise separable convolution alone, seems higher; however, this structure allows for significantly smaller input and output dimensions [35].

This idea was also expanded in [36] into an even faster architecture named MobileNetV3. In contrast to the second version, the building block is modified using squeeze and excitation layers. The squeeze operation uses global average pooling to shrink the input. Subsequently, excitation is applied, which consists of a fully connected layer followed by ReLU and another fully connected layer with sigmoid activation. The final output of the squeeze and excite module is obtained by rescaling the original feature maps with the activation values generated by the excitation operation [37]. MobileNetV3 uses this inside the MobilenetV2 module, as can be seen in Figure 3.8. Another change from the previous version is the replacement of sigmoid activations with piecewise linear hard analog in the form of: hard-$\sigma = \frac{ReLU6(x+3)}{6}$, which also changes another activation used primarily in the deeper layers of the model, called swish (swish$(x) = x\sigma(x)$) to h-swish$(x) = x\frac{ReLU6(x+3)}{6}$. Modifications to the structure were also made in terms of redesigning the computationally expensive layers at the beginning and end of the network [36].

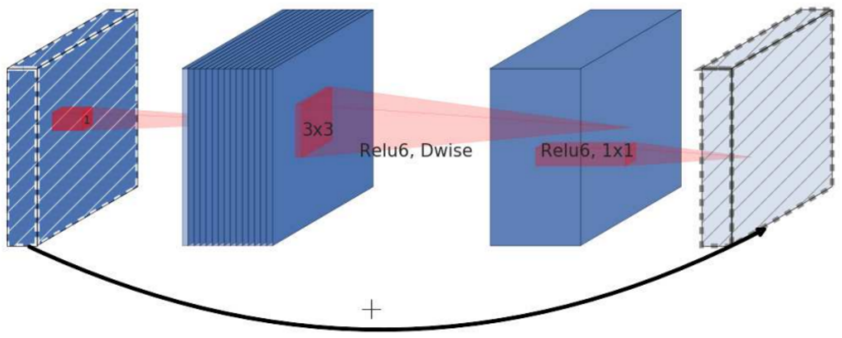


Figure 3.7: Inverted residual block used in MobilenetV2 [35].

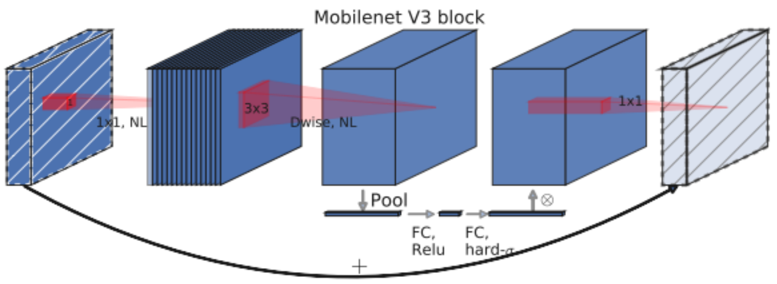


Figure 3.8: Modifications made to inverted residual block in MobieleNetV3 [36].

EfficientNet

A new network scaling method that uses the so-called *compound coefficient* ϕ to uniformly scale width, depth, and resolution (see Figure 3.9) established in [38] aims to improve the accuracy and efficiency of these architectures. The intuition behind this is that for higher resolution images, increasing network depth allows larger receptive fields to capture similar features, and widening the network becomes essential for discerning finer patterns across more pixels, and so the scaling dimensions of the network should be balanced.

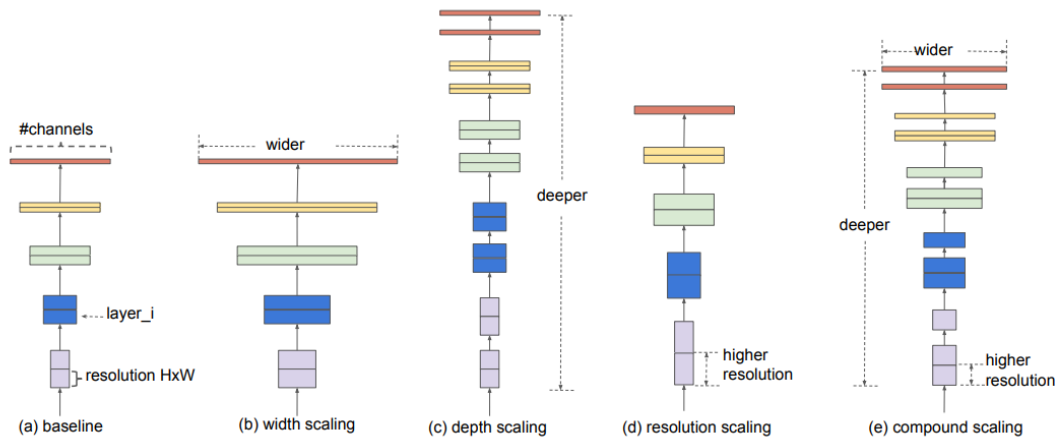


Figure 3.9: The comparison of compound scaling (e) with conventional scaling (b)-(d) that increases one dimension of network width, depth, or resolution on baseline model (a) [38].

The compound coefficient is user defined and controls the availability of resources for model scaling which are assigned to width, depth or resolution based on constants α, β and γ in the following manner:

$$\begin{aligned} \text{depth:} \quad & d = \alpha^{\phi} \\ \text{width:} \quad & w = \beta^{\phi} \\ \text{resolution:} \quad & r = \gamma^{\phi}, \end{aligned}$$

such that $\alpha\beta^2\gamma^2 \approx 2$ and $\alpha,\beta,\gamma \geq 1$. A mobile-sized architecture created in [38] called EfficientNet-B0 uses an inverted bottleneck with squeeze and excitation as its basic building block. Other architectures are derived from this using the compound scaling method (EfficientNet-B1 to EfficientNet-B7). These models have been shown to improve both accuracy and efficiency over previously established CNN models while also proving to be computationally cheaper and faster [38].

Chapter 4

Experimental Setup and Dataset Creation

This Chapter aims to describe in detail the setup of the experiments while giving a reasoning behind the choices made in terms of models, as well as the chosen datasets and the processing of images from the captured recordings. As already mentioned in Chapter 1 computer vision approaches to RSC detection are highly cost-effective and precise, yet somewhat dependent on the choice of dataset. Therefore, comparing previous research on this topic is not as straightforward. Contributing to this uncertainty is also the fact that there is a lot of variability in the conditions that each of the approaches aims to recognize. The choice made for the experiments with RSC sensing in this regard is driven mainly by the fact that the goal of this thesis is to recognize the conditions in terms of the surface state rather than the surface type. This, in combination with the provided data used for creating the dataset, led to the decision to split the target RSC into three classes *wet*, *dry* and *snow* disregarding the surface type, which should allow the models to focus on the main task the aim is to explore.

The choice of CNN model was driven by the fact that for this use case the inference speed is of essence along with the size of the model. Both chosen network structures, *MobileNetV3* Large and *EfficientNetB0* (described in Chapter 3), are created for mobile devices making them a suitable choice for the target task. In addition to this, both CNN models have proven to be very effective in various classification tasks, with EfficientNet generally offering better accuracy in classification tasks, however, at the cost of a slightly slower inference speed. Another favorable feature of using EfficientNetB0 is the straightforward way to scale the baseline model if necessary. Both of these CNN architectures are implemented and available for training in the Pytorch library, which was utilized in this work.

From the existing public datasets described in Chapter 1, RoadSaW was chosen for initial experiments with training these networks. The main motivation behind it was its balanced nature along with the very precise labeling provided. In addition to this, RSCD (which is to the best of our knowledge the only other dataset created exclusively for this task) only provides the images in form of small patches cut from the position in front of the vehicle, where the tires would pass, which in our opinion would provide less information about the road condition in front of the vehicle than the trapezoidal region of interest patch used in RoadSaW covering the whole area that the vehicle will pass through. Furthermore, another advantage of RoadSaW is the fact that the cut-out images are transformed into the bird's-eye view, which should make the model more robust to changes in camera angles and positions. For this thesis, the large size of the road patch was used at a distance of 15 meters from the vehicle, which was the version of the data compatible with the snow coverage images that the authors also provide. The snow coverage set provides three snow class labels *fresh fallen*, *fully packed* and *partially covered* which were all merged into one snow class. In addition to this, the snow data had to be subsampled from existing images to maintain the balance of the dataset. This was done so that images from the train, test and validation subsets were also placed in the same subset in the final RoadSaW set, and each of the original snow classes remained balanced within the merged class of the dataset.

4.1 Dataset Creation and Preprocessing

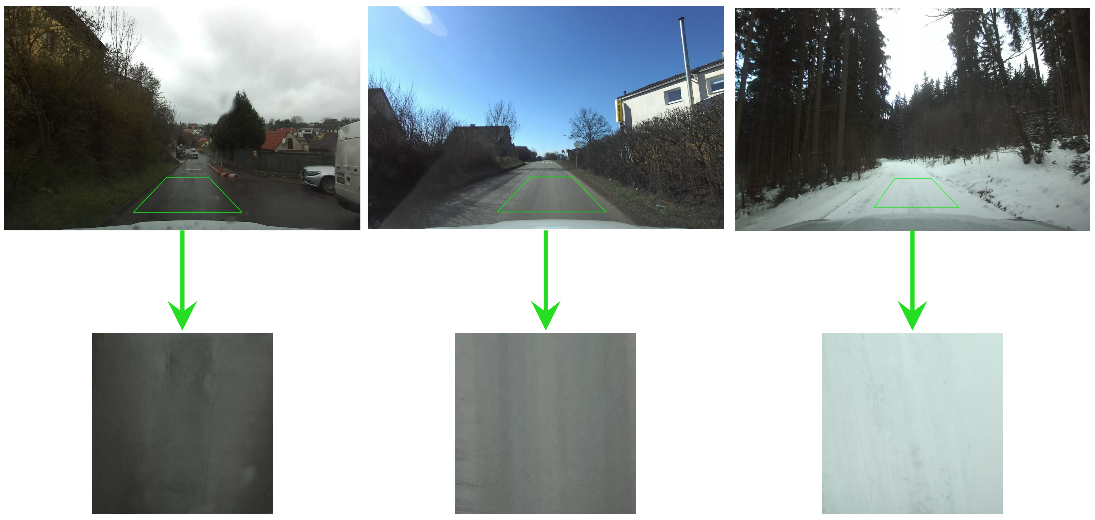


Figure 4.1: Example of preprocessing of the recorded data.

The created dataset consists of 66 recordings captured by the ZED 2 stereo camera mounted behind the rear-view mirror inside the test vehicle (see Figure 4.5) with a resolution of 1920 × 1080 pixels. These were acquired during multiple seasons of the year, providing variability in lighting and weather conditions. Figure 4.3 shows the recording dates and the number of images captured throughout the various drives within them, while the color of the bars indicates the class to which they belong in terms of RSC. The featured locations span across multiple smaller towns in Czechia, including some connecting roads, as well as various locations within the city of Prague and its suburban areas. This introduces a lot of variability in the vehicle speed, the type of road surface, and the conditions of the road in terms of damage or other imperfections. Since this dataset was created in real driving conditions (unlike RoadSaW), it also contains scenarios such as obstacles on the road, parts of vehicles, or people blocking a section of the image, shade, or traces of the sidewalk (see these examples in Figure 4.2).

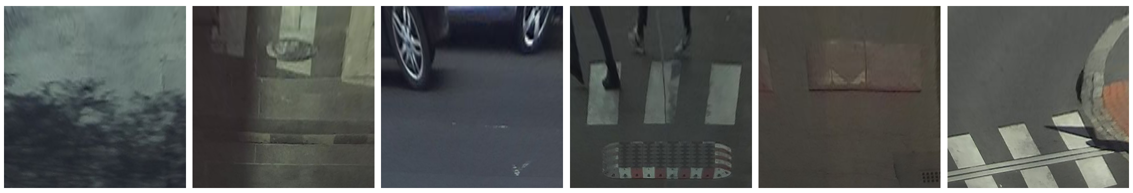


Figure 4.2: Examples of real-world driving conditions included in the created dataset.

From the recordings, which were provided at a rate of 15 FPS, the individual frames were extracted and hand-labeled according to the RSC, disregarding the instances where the condition could not be

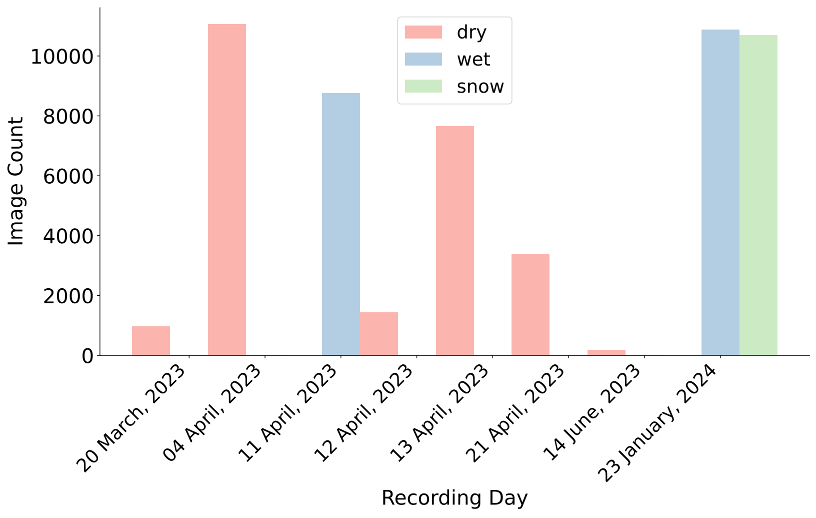


Figure 4.3: Count of images extracted from the drives on the recorded dates on the x-axis, labeled with the assigned RSC classes.

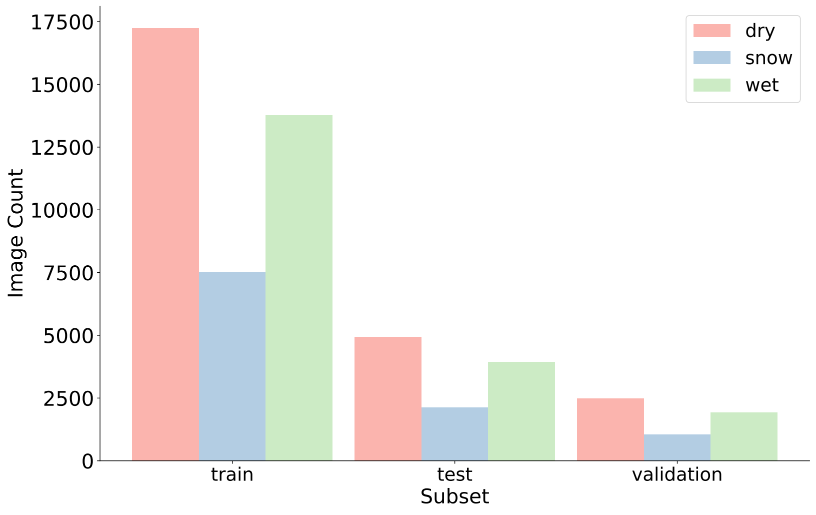


Figure 4.4: Count of images belonging to the different classes in each of the subsets of the dataset.

determined with high confidence. Since some of the locations were within cities and towns, frames where the car was still or barely moving were also eliminated, preventing the final dataset from containing multiple images from one spot. After this process, a trapezoidal region of interest was extracted on the labeled images of the road scenes. The trapezoid dimensions and location were determined by visual inspection of the images. It was positioned to approximately match the width of the vehicle's ego-lane, with its height covering a significant portion of the road ahead, while keeping in mind that too large of a size might often include unwanted noise such as vehicles ahead, people passing through, or, based on the direction of driving, the roadside. In addition to cropping the images, homography mapping was used to obtain a bird's eye view of the road surface (see Figure 4.1).

After preprocessing the recordings, the resulting images were split into train, test and validation set

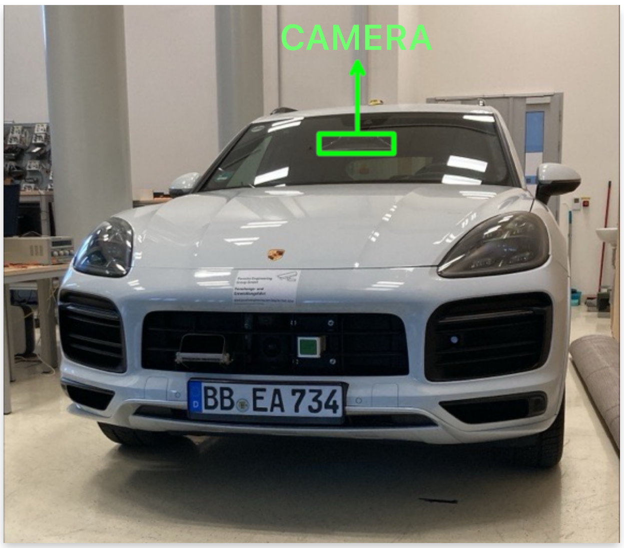


Figure 4.5: Test vehicle with location of the camera.

where the training set accounted for 70%, test 20% and validation 10% of the data. However, this task was not as straightforward since the recordings were of different lengths and the different classes were not evenly distributed within them, making stratified splitting impossible. Therefore, all recordings were further split into shorter sequences of approximately 30 frames with a 20-image gap (corresponding to almost 20 meters when driving at 50 km/h) between each pair. Since the images show only the region of the road surface in front of the vehicle disregarding the surroundings, this gap is assumed to be wide enough for the road to display different characteristics in terms of water and snow distribution on the surface, as well as different properties of the road surface itself and slightly changed lighting conditions. This allowed us to divide the data into a train set of 38 655, validation with 5 358 and a test set consisting of 10 939 images. The class counts in these subsets are visualized in Figure 4.4.

Chapter 5

Evaluation

This Chapter presents the results of experiments with the created dataset as well as the RoadSaW dataset described in Chapter 1. During the research for this thesis there were no models that would have trained weights available for this purpose found. Therefore, a decision was made to initialize the weights as the default pre-trained classification weights for both models in Pytorch which are trained on the ImageNet-1K dataset. Subsequently, both architectures have the classifier adjusted for the three-class classification task, and the output features of the CNN backbone to the classifier were set to use dropout with probability of $p = 0.2$. In addition to this regularization technique, data augmentation is also utilized with random rotation in the range of $[-5°, +5°]$, horizontal flipping, brightness alterations, and also a subtle added Gaussian blur. Different combinations of these alterations and their magnitudes were also examined, providing less satisfactory results. Both networks were trained using SGD with momentum of 0.9 and a decrease in the learning rate was implemented to be triggered on the validation loss plateau. The loss function utilized for both of these networks was the corss-entropy loss defined as:

$$L = -\sum_{m \in B} \sum_{i=1}^{C} y_i^{(m)} \log(\hat{y}_i^{(m)}),$$

where y is the target, $\hat{y}$ output of classifier, C the number of classes and B the input batch. Monitoring the validation loss, early stopping was also used to prevent the architectures from overfitting. This combination of regularization techniques showed the best results not only on the training but also on the validation set based on multiple trainings and evaluations.

5.1 Experiments with RoadSaW

First, the training of both architectures was carried out on the RoadSaW dataset. The reasoning behind the choice to use this data was presented in Chapter 4. To determine the initial leaning rates and batch sizes for both of these models, a grid search was performed, and based on the trend of the train and the validation curves, both of these hyperparameters were selected.

MobileNet

The initial learning rate for this network was set to 10^{-4} with a batch size of 128 images. After further testing of regularization techniques, label smoothing allowed for longer training of the model without overfitting and provided better validation accuracy over previous tests. Training and validation accuracy throughout the training process is shown in Figure 5.1. This plot suggests that both training and validation accuracy seem to have less significant improvements after around 30 epochs. The gap between this metric for train and the validation sets also appears quite small, which implies that the model is able to generalize well. The resulting architecture has 98.86% train and 95.88% validation accuracy (see Table 5.1) with the corresponding confusion matrices visualized in Figures 5.3b and 5.3c.

Table 5.1: Metrics of the final trained MobileNet on the train, and validation sets of RoadSaW.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Validation	95.88	95.89	95.88	95.88
Train	98.08	98.08	98.08	98.08

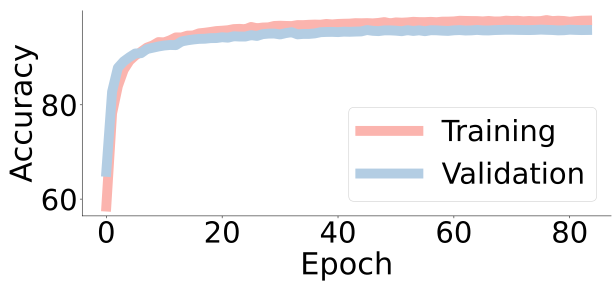


Figure 5.1: Train and validation accuracy with epochs for MobileNet training on RoadSaW dataset.

The results on the test dataset are shown in Table 5.2 and provide the accuracy, precision, recall, and F1 score for each class as well as the mean of these metrics for the overall evaluation of performance. As can be seen from these, the model is highly effective in identifying the snow class, which it is capable of predicting correctly in almost every instance. In the dry class, it also shows high accuracy and recall metrics; however, lower precision suggests that the model is over-predicting this class. The lower recall of identifying wet conditions implies that the model has a tendency to miss some of the wet images. The corresponding confusion matrix in Figure 5.3a further demonstrates that the majority of misclassified samples are between the wet and dry classes, which would be expected. What seems more unusual is the fact that a snow sample was mislabeled as dry. Upon closer inspection of these inaccuracies (see the examples in Figure 5.2), the incorrectly predicted instances between wet and dry classes were mainly due to what appeared to be the higher speed of the vehicle during the capturing of the mentioned images, which possibly distorted some of the important texture in the image. The sample misclassified as dry by the model, despite belonging to the snow class, also depicted in the Figure, show some resemblance to certain instances of dry images.

Table 5.2: Metrics of the final trained MobileNet for prediction on RoadSaW.

Class	Accuracy	Precision	Recall	F1 Score
Overall	96.92	97.10	96.92	97.00
dry	99.32	92.16	99.32	95.60
snow	99.77	100.00	99.77	99.89
wet	91.67	99.14	91.67	95.26

Moving Window Prediction Smoothing

During video data analysis, it was observed that class transitions tend to be less abrupt and more continuous. Thus, moving-window aggregation of predictions in time seems like a straightforward method of improving the model's performance. This approach involves synthesizing the classification results

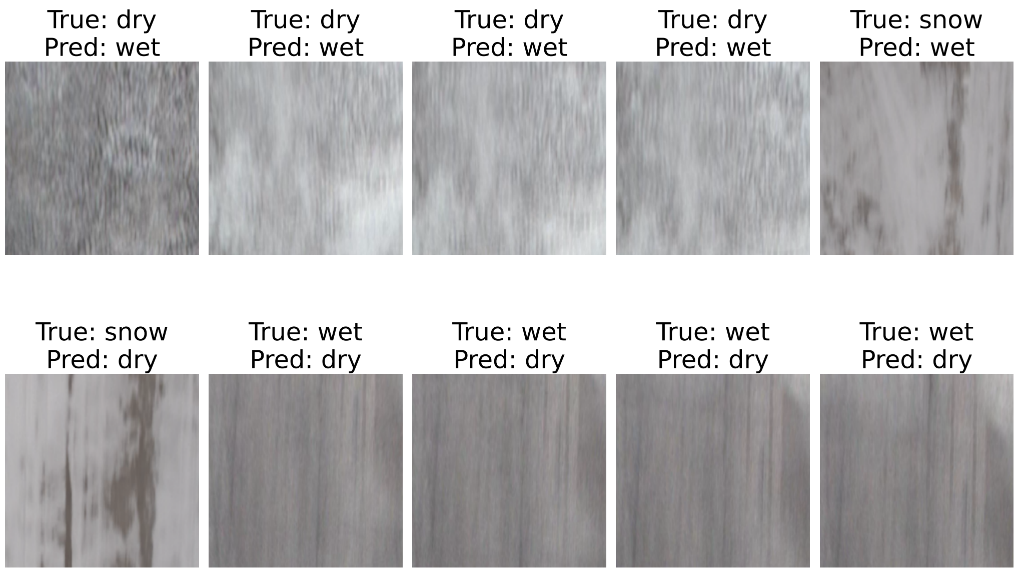


Figure 5.2: Examples of misclassified images from test set of RoadSaW by trained MobileNet architecture.

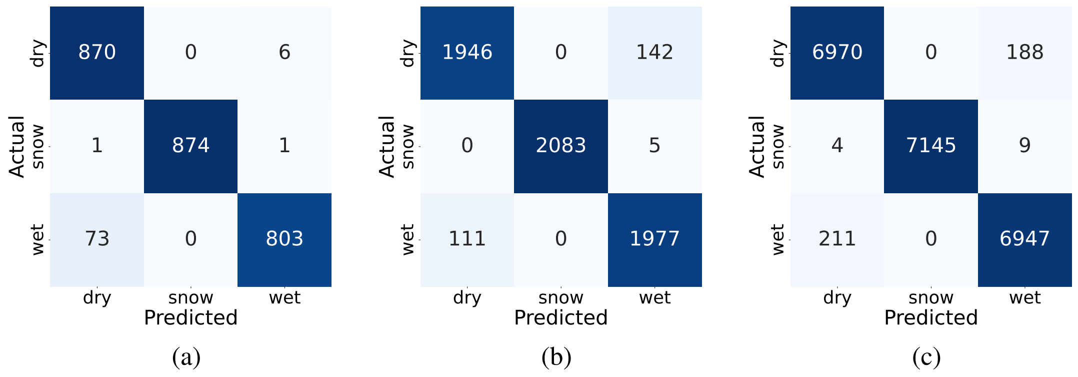


Figure 5.3: Confusion matrices for MobileNet on the RoadSaW dataset. Matrix (a) shows the classification results for test dataset, (b) for validation and (c) for the train set

from a set of neighboring frames and smoothing the final results. For this experiment, every new prediction is given as a majority class in the current window, which in this case was used of size five. This size of window would examine a segment of video that spans approximately one sixth of a second, assuming that the camera is providing videos at 30 frames per second (which the test car camera is capable of producing). It was set small enough to preserve some temporal resolution of the data and to ensure that any rapid transitions in weather conditions can be detected without significant delay. Given that each class in this dataset contains the same number of images, the batch size of the said size was used to avoid overlapping the predictions in subsequent classes. This choice was made in order to make the results more general and not create errors that would depend on the order in which the classes were input into the classifier, thus the results do not include the misclassifications that would have been caused by the delay due to window aggregation.

Table 5.3: Metrics of the final trained MobileNet on the test, train, and validation sets of RoadSaW dataset for prediction using the aggregation technique.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Test	97.72	97.85	97.72	97.78
Validation	96.41	96.41	96.41	96.41
Train	98.43	98.43	98.43	98.43

Table 5.3 presents the results of the aggregation on all subsets of RoadSaW. These indicate a nearly 1% improvement in the overall metrics on the test and validation set, and some minor improvements on the training set. Thus, the aggregation process was shown to be effective in improving the performance of the model even with a window of size as small as five frames.

EfficientNet

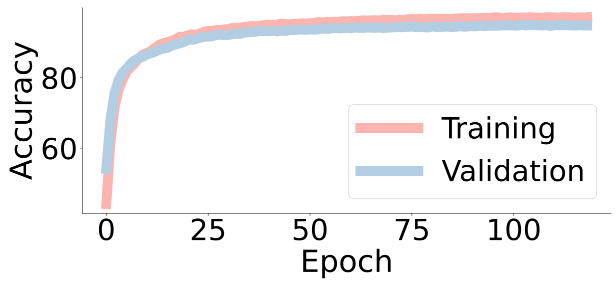


Figure 5.4: Train and validation accuracy with epochs for EfficientNet training on RoadSaW dataset.

This model showed the best characteristics when training with a batch size of 256 and the initial learning rate 10^{-4}. In case of this architecture, label smoothing has not proven to be as effective, and thus it was left out. The training curves were plotted in terms of accuracy the same as for the previous architecture and are displayed in Figure 5.4. These curves again exhibit no signs of overfitting with a minimal gap between them. Also visible from the plot is that this model was able to train for a longer period than the previous one without showing signs of overfitting and triggering the early stopping mechanism. The final trained EfficientNet model achieves almost 98% in all metrics on the training set and 95% on the validation set (see Table 5.4). The confusion matrices in Figure 5.5 again imply that while the model is still very accurate, it may occasionally mistake wet conditions for specifically dry ones. However, this occurs a bit more frequently for EfficientNet than with the trained MobileNet architecture. Upon examination of the misclassified frames, high car speed seems to be again the major cause of the wrong classifications.

Table 5.4: Metrics of the final trained EfficientNet on the train and validation sets of RoadSaW.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Validation	94.86	94.88	94.86	94.87
Train	97.64	97.64	97.64	97.64

Table 5.5 presents the results for this model on the RoadSaW test set. The overall accuracy of 96.42% and F1 score of 96% indicate that the model is highly effective at classifying RSCs in most conditions. It was able to learn the patterns in the snow class perfectly, not even once producing a wrong

prediction. The lower precision for the dry class and the reduced recall for the wet class indicate that further improvements could be made in distinguishing between the two. However, the overall test metrics demonstrate good performance in RSC detection.

Table 5.5: Metrics of the final EfficientNet for prediction on the test set of RoadSaW.

Class	Accuracy	Precision	Recall	F1 Score
Overall	96.42	96.54	96.42	96.48
Dry	97.72	92.04	97.72	94.80
Snow	100.00	100.00	100.00	100.00
Wet	91.55	97.57	91.55	94.46

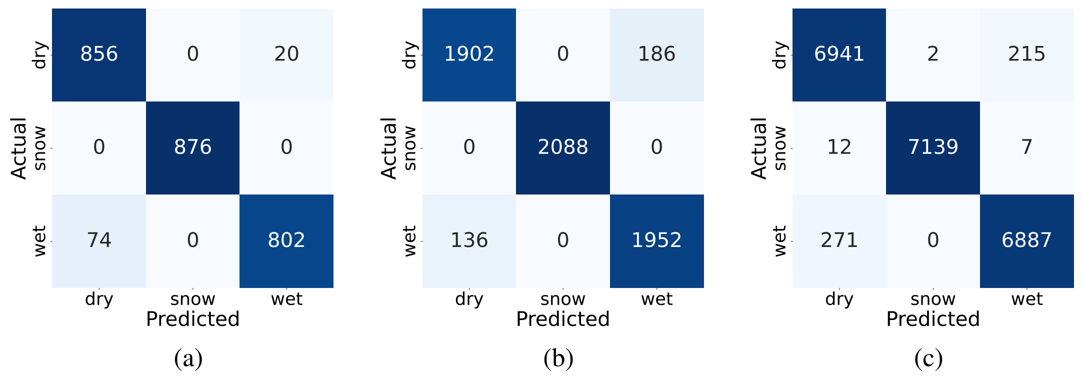


Figure 5.5: Confusion matrices for EfficientNet on the RoadSaW dataset. Matrix (a) shows the classification results for test dataset, (b) for validation and (c) for the train set.

Moving Window Prediction Smoothing

Incorporating a moving window in time aggregation of predictions increases all of the performance metrics of the network (see Table 5.6) on the different subsets of RoadSaW, indicating better generalization and consistency in model predictions. Although there is a clear improvement of the overall metrics, it is not very substantial suggesting that the predictions were already quite smooth and the temporal aggregation is less valuable for this model.

Table 5.6: Metrics of the final trained EfficientNet for prediction on RoadSaW using window aggregation approach.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Test	97.49	97.65	97.49	97.57
Validation	95.21	95.23	95.21	95.22
Train	98.27	98.27	98.27	98.26

5.2 Experiments with Created Dataset

Even though the models performed well on the RoadSaW test set, the performance of the trained architectures on the created dataset was quite low. This could presumably have been caused by changes in the size, shape, and location of the extracted region, by the different hardware and calibration used for data collection or even the different location with varying roads. Furthermore, the created dataset consists of real-world driving scenarios, unlike RoadSaW, which was captured in a controlled environment. Therefore, the contrast in recording conditions might have caused challenges in translating the knowledge of the trained models on RoadSaW to real-world driving situations. Due to this poor performance, the weights had to be trained again, initializing with the default Pytorch classification weights from ImageNet-1K, on the train set of the collected data. The hyperparameters and regularization techniques used were initially kept in line with the previous experiment, as they had proven to be effective in the RSC classification task. One difference from previous training was that both networks were trained using weighted cross entropy loss, where weights are calculated as follows:

$$\text{weight}_{class} = \frac{\text{total number of samples}}{\text{number of classes} \cdot \text{number of samples in class}},$$

to account for the imbalance in the classes.

MobileNet

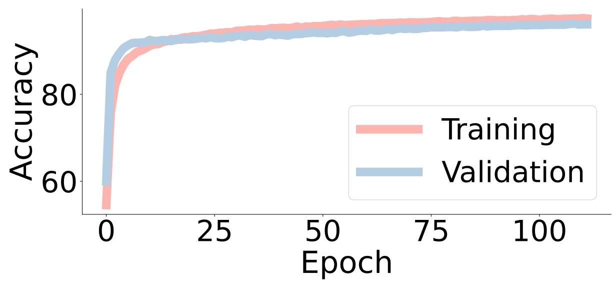


Figure 5.6: Train and validation accuracy with epochs for MobileNet training on the created dataset.

The initial training with the unchanged hyperparameters and regularization techniques displayed signs of overfitting early in the training. Tuning some of the hyperparameters showed that the architecture could benefit from larger batch size of 256 and restraining from using the blurring augmentation. The final training curves in Figure 5.6 again demonstrate good generalization abilities and the effectiveness of the regularization techniques used. The final training and validation metrics are provided in Table 5.7 and show quite high accuracy on the validation set of almost 97% and on training data over 98%. This again implies good generalization abilities of the trained network on unseen data. The results on the test set displayed in Table 5.8 provide the classification metrics for individual classes. Here, misclassifications tend to become more frequent in the snow class than on the previous dataset, while the dry class provides the best results of almost 99% in all metrics. More insight into mislabeled instances is provided by the confusion matrices in Figure 5.7. The model seems to be less successful in distinguishing the wet and the snowy conditions, which is possibly caused by the snow classes containing more instances of melting snow. The overall test results show quite good performace on unseen data with F1 score over 95% and the accuracy a little more than 96%.

Table 5.7: Metrics of the final trained MobileNet on the train and validation sets of the created dataset.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Validation	96.60	96.06	96.02	96.04
Train	98.24	98.20	98.19	98.19

Table 5.8: Metrics of the final trained MobileNet for prediction on the created dataset.

Class	Accuracy	Precision	Recall	F1 Score
Overall	96.15	95.65	95.02	95.32
Dry	99.01	98.61	99.01	98.81
Snow	90.57	94.35	90.57	92.42
Wet	95.48	94.00	95.48	94.73

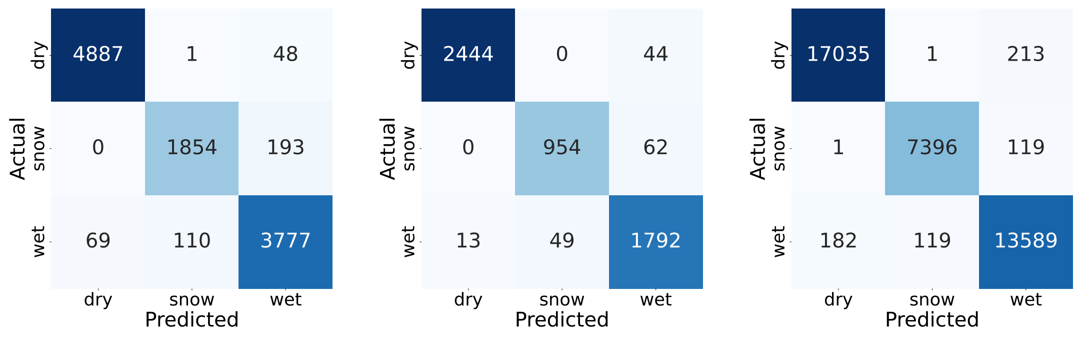


Figure 5.7: Confusion matrices for MobileNet on the created dataset. Matrix (a) shows the classification results for test dataset, (b) for validation and (c) for the train set.

Moving Window Prediction Smoothing

Aggregation of predictions with the moving window is evaluated including errors caused by delay. Even with these errors, the metrics improved, although not very significantly (see Table 5.6). The highest test accuracy achieved with this network on the real-world data is 96.5% and the F1 score of 95.6%.

Table 5.9: Metrics of the final trained MobileNet on the test, train, and validation sets of the created dataset for prediction using the aggregation technique.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Test	96.54	95.98	95.29	95.62
Validation	97.31	96.84	96.60	96.70
Train	98.81	98.81	98.75	98.78

EfficientNet

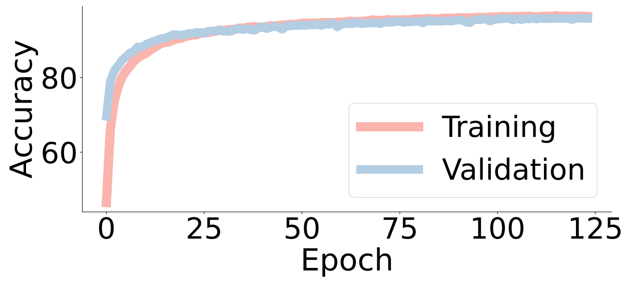


Figure 5.8: Train and validation accuracy with epochs for EfficientNet training on the created dataset.

The EfficientNet architecture, when trained on the created dataset, again showed good generalization abilities, as can be seen from the training and validation curves in Figure 5.8. However, the performance on this dataset seems to be slightly lower compared to RoadSaW, with the final validation and training metrics provided in Table 5.10. Although these are not as high, they still provide quite good performance in the more complex situations, both exceeding 96%. Further examination of the confusion matrices in Figure 5.9 shows that most misclassifications occur between the wet and dry classes or between the snow and wet classes. The reason the latter is occurring with these data in contrast to RoadSaW is that the created dataset contains samples in the snow class that are melting unplough snow which might often resemble wet roads. Another observation made is that misclassifications between wet and dry classes are in majority due to damage and imperfections in the road surface which could cause the model to mistakenly interpret these as water. Figure 5.10 shows a few of the misclassified samples,

Table 5.10: Metrics of the final trained Efficientnet on the train, and validation sets of creted dataset.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Validation	95.88	95.61	95.48	95.54
Train	97.41	97.51	97.33	97.42

where the most notable last one shows a case with traces of visible slushy snow on the road surface. Under the said conditions, it was difficult even for the human eye to determine whether the surface conditions are wet or snowy. There are additional samples featuring unploughed melting snow, which tend to result in higher error rates from the model.

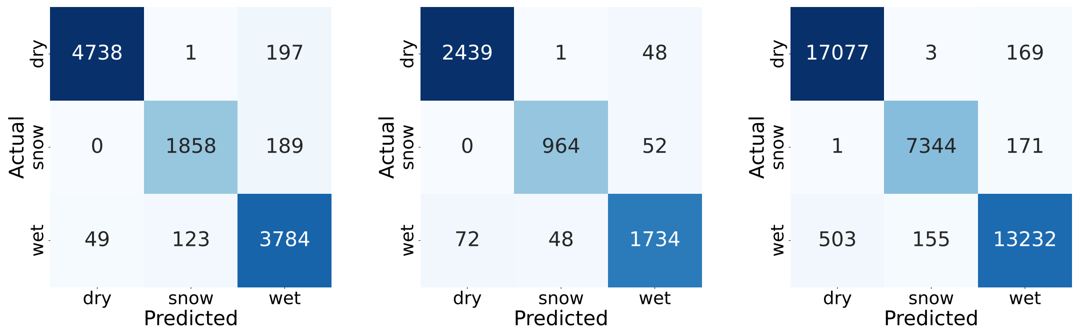


Figure 5.9: Confusion matrices for MobileNet on the RoadSaW dataset. Matrix (a) shows the classification results for test dataset, (b) for validation and (c) for the train set.

The results on the test set are presented in Table 5.11 and also indicate that the model is having difficulty identifying the snow class with the recall of almost 91% suggesting it has tendency to miss some of the snow predictions, which, as can be seen in Figure 5.9, are most frequently mislabeled as wet. Despite this imperfection the accuracy of almost 95% proves that the model is well suited for the RSC detection task.

Table 5.11: Metrics of the final trained EfficientNet for prediction on the created dataset.

Class	Accuracy	Precision	Recall	F1 Score
Overall	94.89	94.49	94.14	94.31
Dry	95.99	98.98	95.99	97.46
Snow	90.77	93.74	90.77	92.23
Wet	95.65	90.74	95.65	93.13

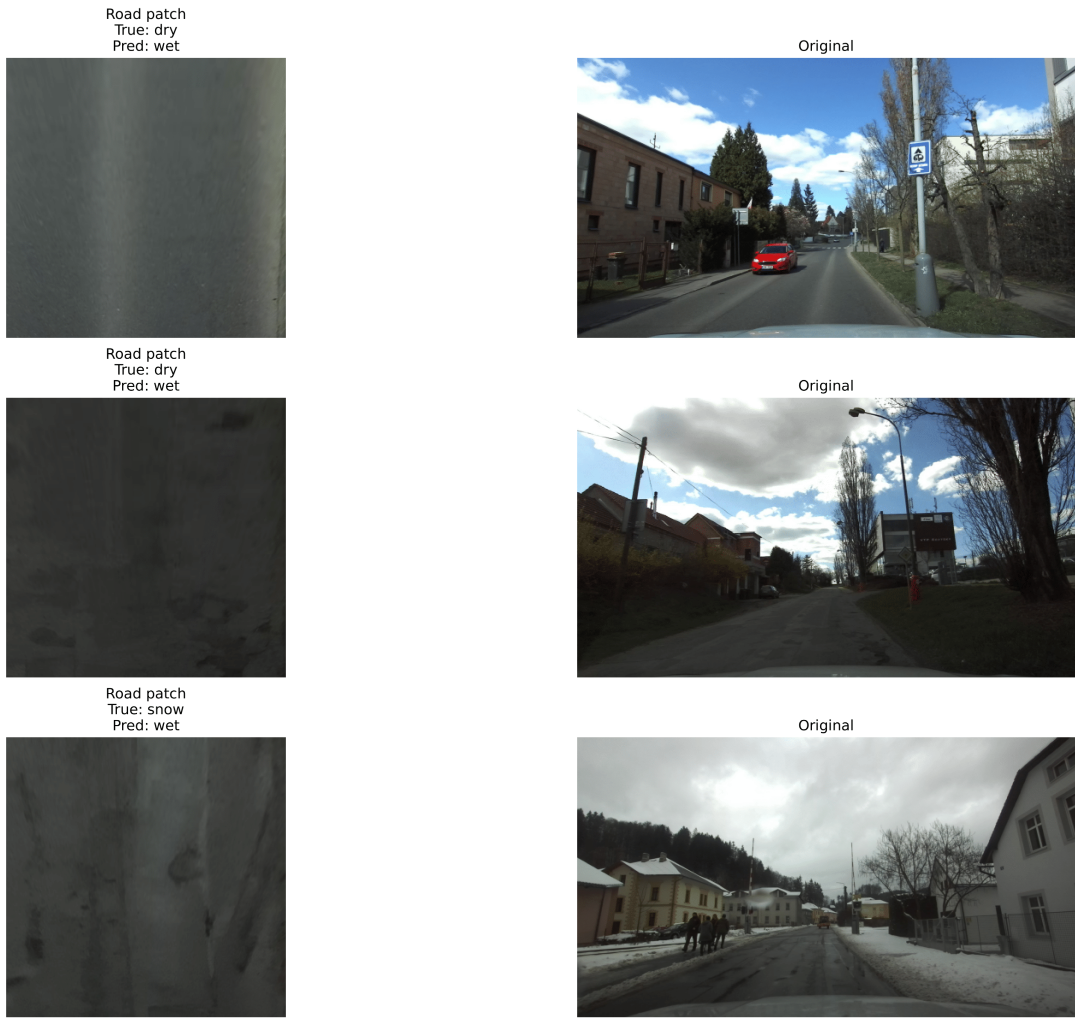


Figure 5.10: Misclassified road patches by EfficientNet architecture on the test set of created dataset along with the original road scene to which the cutout corresponds to.

Moving Window Prediction Smoothing

Window aggregation of results was again performed in a manner that there are two expected delayed predictions in the test and train set and three in the validation set, lowering the metrics. However, as can be seen from Table 5.12 even with these expected errors, the performance of this model improved by around 1% on the test set and less rapidly on the training and validation set, providing the highest test results of almost 96%.

Table 5.12: Metrics of the final trained EfficientNet on the train and validation sets of created dataset for prediction using the aggregation technique.

Subset	Accuracy [%]	Precision [%]	Recall [%]	F1 Score [%]
Test	95.93	95.44	94.92	95.18
Validation	96.55	96.28	96.28	96.28
Train	98.05	98.08	98.10	98.09

5.3 Prediction Speed Evaluation

As previously mentioned, the prediction speed of the chosen models is of essence in real-world scenarios. The two previous sections compared these networks in terms of classification performance on the two datasets. However, for the model to be able to perform in real-time, these networks need to be able to produce predictions at very high speed to keep up with the incoming samples. To evaluate how these models perform in this regard, the `timeit` library in Python is utilized. The inference speeds reported in Table 5.13 are given as a mean of 200 predictions on Laptop GPU NVIDIA GeForce RTX 3050 Ti and a CPU AMD Ryzen 7 5800U. MobileNet shows a faster inference time, with the ability to process one image in 23 ms on GPU, which suggests that in this setting, it has potential to provide results even with videos of a 30 frames per second (fps) rate. With slower inference even on GPU EfficientNet is still very effective and has the capacity to provide results in real-time on both CPU and GPU with 15 fps.

Table 5.13: Inference speed on GPU and CPU for both networks given as a mean of 200 prediction times.

Network	Inference Time GPU [ms]	Inference Time CPU [ms]
EfficientNet	35.53	55.54
MobileNet	23.20	40.00

Although both architectures provide very fast inference, only MobileNet has the potential to be used at a 30 fps rate, which could also possibly allow for bigger window for prediction aggregation and possibly smoother, more precise results.

5.4 Discussion

This Chapter presented the results of chosen CNN architectures, MobileNet and EfficientNet, for the RSC detection task. These were evaluated on two datasets: a public computer vision dataset for road surface condition classification (RoadSaW) and a dataset created for the purposes of this thesis from test vehicle recordings. Both of the architectures showed good performance on RoadSaW that exceeded the accuracy of 96% on all subsets, with MobileNet resulting in slightly better classification metrics.

The models with trained weights were further tested on the collected data, showing poor performance under real-world conditions. Both architectures required further training to adapt to the different driving scenarios and preprocessing techniques introduced in the new dataset.

This dataset was created from the recordings of 66 drives during 8 recording dates in different seasons and weather conditions. The images were labeled, split into short 30-frame drives, and reduced to only road surface cut-outs located in front of the vehicle transformed into a top view. The splitting of drives was necessary for the dataset creation because of uneven class distribution within the class which would have prevented us from creating a stratified split into training, test and validation sets. One shortcoming of this dataset is what appears to be a slightly uneven distribution of images with different snow conditions within the snow class, where images with various levels of melted snow seem to be not as evenly distributed, causing the models to have worse performance in this category.

Training both architectures on the training set showed promising results, which are presented together with the results on RoadSaW in Table 5.14. As this Table suggests, the MobileNet architectures was able to outperform EfficientNet on both of the datasets, proving to be very efficient in the RSC detection task. Additional evaluation was provided for both networks, employing window aggregation of predictions to smooth out the final results. This approach is in line with the gradual nature of changes in road surface conditions in real-world driving scenatios and results in a slight delay of predictions. A small window of 5 predictions was used and the improved accuracies of both networks are presented in Table 5.15. The improvement of said metric was more visible on the validation set of both models. Even in this setting, trained MobileNet architecture has proven to have better performance in training and validation set. The final test metrics of both models on the created dataset presented showed good generalization abilities, with lower performance metrics for the less optimally balanced snow class presumably caused mainly by this shortcoming.

Table 5.14: Results of both models on the train and validation sets of the two datasets.

Trained Architecture	Dataset	Subset	Accuracy [%]
MobileNet	RoadSaW	Train	98.08
		Validation	95.88
	Recorded	Train	98.24
		Validation	96.60
EfficientNet	RoadSaW	Train	97.64
		Validation	94.86
	Recorded	Train	97.41
		Validation	95.88

Finally, the architectures were evaluated in terms of inference speed and their ability to produce predictions at different frame rates of the recording device. Both models provided a fast prediction speed capable of processing data in time for a 15-fps recording rate. However, MobileNet outperforms EfficientNet while proving to be capable of processing images even at 30 fps rate on the GPU.

Numerous computer vision approaches to RSC detection were presented in Chapter 1. These vary in the conditions they aim to recognize, the datasets used, and the extraction of the region of interest for prediction. Therefore, the comparison with the approach taken in this thesis is not as straightforward. The evaluation of trained MobileNetV2 provided in [6] classified three types of road surface and two road surface conditions, resulting in an F1 score of almost 93% on the test set with the same region of interest size and shorter distance to the vehicle than the one used in this thesis. In comparison, the better performing trained MobileNetV3 architecture was able to produce the mean F1 score for dry and wet conditions of 95.43% without and 96.56% with window aggregation on the test set of road patches

Table 5.15: Results of both models on the train and validation sets of the two datasets with prediction aggregation.

Trained Architecture	Dataset	Subset	Accuracy [%]
MobileNet	RoadSaW	Train	98.43
		Validation	96.41
	Recorded	Train	98.81
		Validation	97.31
EfficientNet	RoadSaW	Train	98.27
		Validation	95.21
	Recorded	Train	98.05
		Validation	96.55

placed further in the distance (creating more challenging conditions). With a similar goal in terms of RSC estimation classes, the authors in [16] trained multiple models for the detection of dry, wet and snow conditions, as well as for snow tracks in all driving lanes, using multiple autonomous driving datasets. They were able to achieve an F1 score maximum of 90.9% for the ego-lane RSC classification, compared to this result, both models in this thesis achieved an F1 score of more than 94% on the test set with more resource efficient architectures.

Based on the comparisons with the previous approaches, both of the models used for RSC detection in combination with the chosen data preprocessing techniques and surface condition classes seem to provide comparable results. Although EfficientNet has proven to be effective in RSC estimation, MobileNet was able to outperform this architecture on both datasets and inference speed abilities, making it the better suited approach for this task.

Conclusion

The aim of this thesis was to find suitable computer vision methods applicable to real-time RSC detection. Two architectures were chosen on the basis of their high inference speed and promising performance in various classification tasks. Also explored was the option of using architectures with pre-trained weights for RSC classification, which to the best of our knowledge were not available. Therefore, we resorted to training the models from scratch on the chosen data.

First, a public dataset called RoadSaW was used to train chosen models mainly for its very precise class labeling (using Mobile Advanced Road Weather Information Sensor) and the preprocessing in form of a transformed cutout of the road surface that was also used in the created dataset. This set was further altered to fit the classification task that this work was aiming to explore, by disregarding the surface type labels and merging the dataset with the compatible snow coverage data.

For training of these convolutional networks, different batch sizes, learning rates, and regularization techniques were tested. The performance of both trained networks showed good generalization abilities on unseen data and high classification measures on the train and the validation set. A limitation of both models was the occasional misclassification between dry and wet classes, which upon closer inspection appeared to be primarily caused by the higher speeds at which the images were captured. Overall, the trained MobileNet architecture performed better on train and test set, providing higher accuracy by almost 1% on both of these sets.

In sensing RSC, it generally appears that class transitions are more gradual and abrupt changes are less common. Therefore, smoothing model predictions over even a small moving window could enhance overall results. How this aggregation of classification outputs contributes to better results of the model was tested. Even with a window as short as five frames, the classification metrics of the MobileNet architecture increased around 1%, while for EfficientNet it was slightly lower. The highest classification metrics achieved by the better-performing MobileNet all reached almost 98% on the RoadSaW test set.

For the purposes of testing the trained models on real-world data, a dataset using recordings from the test vehicle was created. Provided data include various driving scenarios recorded within cities, smaller towns and even some connecting roads between them all located within Czechia. The collection of images was recorded at a rate of 15 fps and the recording dates span between March and January capturing variety of seasons and weather conditions. The uneven distirbution of classes between the individual drives did not allow for stratify splitting of the data into train, test, and validation subsets. Therefore, these had to be further split into shorter segments of 30 images while discarding 20 sample segments (corresponding to almost 20 meters when driving at 50 km/h) between each of them. For the model to focus only on the road in front of it and not the surrounding weather conditions, a trapezoidal cutout of the road surface was extracted from each of the frames and transformed into a bird's-eye view using homography. The shape of these was chosen to be wide enough to fit into the vehicle ego lane and capture a wide range of the surface, while keeping in mind that too large of a height of the trapezoid will capture a lot of redundant information as well. The final dataset created in this thesis consists of almost 55 000 images from 1 854 short 30-frame drives. One limitation of our dataset was the uneven distribution of images of the road surface with melting snow slush across the snow classes in the split dataset.

Although both trained models were very effective at identifying RSC on RoadSaW, their abilities did not translate very well into real world conditions. After examining the performance of the trained networks on the created dataset, it was observed that the models had to be trained again in order to adapt to the new recording conditions. Since the previous training setup was shown to be very effective for the RSC detection task, all regularization techniques and hyperparameters were initially kept unchanged. While EfficientNet showed acceptable performance and good regularization abilities in this set-up, MobileNet required small adjustments to be able to trian well for this task. The results from this model again showed to be more favorable with the MobileNet architecture achieving test accuracy of a little over 96%.

Finally, the networks were compared in terms of inference speed. Both models demonstrated high processing speed with the potential to be used at the 15 fps recording rate. However, MobileNet has demonstrated greater effectiveness in this regard, indicating capability of generating predictions at a 30 fps rate on the GPU. While both networks demonstrated high processing speeds suitable for real-time RSC detection, MobileNet consistently outperformed EfficientNet not only in terms of inference speed but also in classification performance metrics on both of the datasets in training and validation subsets. Thus, proving its potential for deployment in practical applications at a frame rate of 30 fps on a GPU and very good RSC detection abilities, MobileNet has shown to be the more effective approach to this task.

One of the challenges encountered in this thesis was the difficulty in comparing the classification results directly with previous studies due to differences in datasets and RSC classes. Despite these challenges, the proposed models, especially MobileNet, gave satisfactory results, even outperforming some of the previous similar researches in this field with more extensive data and more computationally efficient models but with fewer classes. Nevertheless, limitations such as the uneven distribution of different snow types within the snow class in the created dataset or the occasional misclassification of wet and dry images at higher vehicle speeds on RoadSaW indicate areas for improvement. To solve this issue, future studies might also investigate the integration of additional vehicle speed information in the data to improve the accuracy and robustness of the classification. Achieving generalization across different cameras or locations has also shown to pose a significant challenge, and it is not guaranteed that a model trained on this dataset would be as effective in other vehicles with different settings. However, the focus of this thesis was to develop a model that could be effectively deployed on the test vehicle, and in this context, the results have been proven to be both robust and applicable.

Bibliography

[1] Amr Abdelraouf, Mohamed Abdel-Aty, and Yina Wu. "Using Vision Transformers for Spatial-Context-Aware Rain and Road Surface Condition Detection on Freeways." In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 18546–18556. DOI: 10.1109/TITS.2022.3150715.

[2] Yao Ma et al. "Current Non-Contact Road Surface Condition Detection Schemes and Technical Challenges." In: *Sensors* 22.24 (2022). ISSN: 1424-8220. DOI: 10.3390/s22249583.

[3] Sohini Roychowdhury et al. "Machine Learning Models for Road Surface and Friction Estimation using Front-Camera Images." In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489188.

[4] Chien Pin Sherman Hsu et al. "Infrared spectroscopy." In: *Handbook of instrumental techniques for analytical chemistry* 249 (1997).

[5] You Li and Javier Ibanez-Guzman. "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems." In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61. DOI: 10.1109/MSP.2020.2973615.

[6] Kai Cordes et al. "RoadSaW: A Large-Scale Dataset for Camera-Based Road Surface and Wetness Estimation." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2022, pp. 4439–4448. DOI: 10.1109/CVPRW56347.2022.00490.

[7] Daniel Fink et al. "Resource Efficient Classification of Road Conditions through CNN Pruning." In: *IFAC-PapersOnLine* 53.2 (2020), pp. 13958–13963. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2020.12.913.

[8] Fisher Yu et al. *BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning*. 2020. arXiv: 1805.04687 [cs.CV].

[9] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset." In: *International Journal of Robotics Research (IJRR)* (2013).

[10] Dan Barnes et al. "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset." In: *arXiv preprint arXiv: 1909.01300* (2019).

[11] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[12] Carlos A. Diaz-Ruiz et al. *Ithaca365: Dataset and Driving Perception under Repeated and Challenging Weather Conditions*. 2022. arXiv: 2208.01166 [cs.CV].

[13] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021.

[14] Tong Zhao et al. “A Comprehensive Implementation of Road Surface Classification for Vehicle Driving Assistance: Dataset, Models, and Deployment.” In: *IEEE Transactions on Intelligent Transportation Systems* 24.8 (2023), pp. 8361–8370. DOI: `10.1109/TITS.2023.3264588`.

[15] Daniel Fink et al. “Resource Efficient Classification of Road Conditions through CNN Pruning.” In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 13958–13963. ISSN: 2405-8963. DOI: `10.1016/j.ifacol.2020.12.913`.

[16] Hasith Karunasekera et al. “Varying Road Surface Condition Estimation in Ego and Adjacent Lanes.” In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–6. DOI: `10.1109/IV55152.2023.10186540`.

[17] David Vosahlik et al. “Self-Supervised Learning of Camera-based Drivable Surface Friction.” In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 2773–2780. DOI: `10.1109/ITSC48978.2021.9564894`.

[18] Sheela Ramanna et al. “Near real-time map building with multi-class image set labeling and classification of road conditions using convolutional neural networks.” In: *Applied Artificial Intelligence* 35.11 (2021), pp. 803–833. DOI: `10.1080/08839514.2021.1935590`.

[19] Md Nasim Khan and Mohamed M Ahmed. “Weather and surface condition detection based on road-side webcams: Application of pre-trained convolutional neural network.” In: *International journal of transportation science and technology* 11.3 (2022), pp. 468–483. DOI: `10.1016/j.ijtst.2021.06.003`.

[20] Amr Abdelraouf, Mohamed Abdel-Aty, and Yina Wu. “Using vision transformers for spatial-context-aware rain and road surface condition detection on freeways.” In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 18546–18556. DOI: `10.1109/TITS.2022.3150715`.

[21] Fei Hu and Qi Hao. *Intelligent sensor networks: the integration of sensor networks, signal processing and machine learning*. Taylor & Francis, 2012.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[23] Javaid Nabi. *Machine Learning —Fundamentals*. en. May 2019. URL: `https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916` (visited on 07/01/2022).

[24] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey.” In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285. DOI: `10.48550/arXiv.cs/9605103`.

[25] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks.” In: *Information Processing & Management* 45.4 (2009), pp. 427–437. ISSN: 0306-4573. DOI: `doi.org/10.1016/j.ipm.2009.03.002`.

[26] Imrus Salehin and Dae-Ki Kang. “A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain.” In: *Electronics* 12.14 (2023). ISSN: 2079-9292. DOI: `10.3390/electronics12143106`.

[27] Xue Ying. “An overview of overfitting and its solutions.” In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022. DOI: `10.1088/1742-6596/1168/2/022022`.

[28] Cui Xu Xiao-Ling Xia and Bing Nan. “Facial expression recognition based on tensorflow platform.” In: *ITM Web of Conferences* 12 (2017), p. 01005. DOI: `10.1051/itmconf/20171201005`.

[29] Jan Flusser, Tomáš Suk, and Barbara Zitová. *2D and 3D Image Analysis by Moments*. English. 1st edition. Chichester, West Sussex, United Kingdom; Hoboken, NJ: Wiley, Dec. 2016. ISBN: 978-1-119-03935-8.

[30] Rafael Gonzalez and Richard Woods. *Digital Image Processing*. English. 4th edition. New York, NY: Pearson, Mar. 2017. ISBN: 978-0-13-335672-4.

[31] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks." In: *arXiv preprint arXiv:1511.08458* (2015). DOI: doi.org/10.48550/arXiv.1511.08458.

[32] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. en. Dec. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 05/03/2022).

[33] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[34] Chi-Feng Wang. *A Basic Introduction to Separable Convolutions*. https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728. (Accessed on 04/05/2024). Aug. 2018.

[35] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520. DOI: 10.48550/arXiv.1801.04381.

[36] Andrew Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV].

[37] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[38] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].