Università degli Studi di Salerno

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA
E MATEMATICA APPLICATA

Corso di Laurea in Ingegneria Informatica

Multi-task techniques for road weather classification and surface anomaly detection

Supervisor:
Prof. Diego Gragnaniello

Candidate:
Mattia Marseglia
Mat. 0622701697

Supervisor:
Prof. Alessia Saggese

ANNO ACCADEMICO 2022/2023

Abstract

Description of the problem faced

The automotive industry is undergoing a technological revolution thanks to the use of machine learning applied to Advanced Driver Assistance Systems (ADAS). Neural network-based systems have the potential to significantly enhance vehicle safety and control by accurately detecting various road conditions.

In this sector, the quest for innovation is relentless, with the automotive industry increasingly collaborating with the technology sector to develop such advanced solutions. The ultimate goal is to create increasingly autonomous and safe vehicles, harnessing the full potential of artificial intelligence and machine learning systems. One particularly promising field, which is also the problem addressed in this thesis, is the classification of road surface conditions, divided into three distinct tasks: estimating vehicle-road friction related to weather conditions, recognizing road surface material, and identifying road irregularities. Solving these challenges is essential to enable ADAS to dynamically adapt driving behavior, promoting both safety and efficiency.

This research will explore the necessary solutions and approaches to fully leverage the potential of artificial intelligence in addressing these tasks.

Framing of the paper in the contemporary technical scenario

To address the problem outlined in the previous section, we adopted a resolution approach based on multi-task classification. Initially, we conducted a thorough analysis to identify the most promising datasets available in the state of the art. This initial research phase allowed us to fully understand the breadth and complexity of the resources at our disposal.

Subsequently, we evaluated the solutions proposed in the field of machine learning and delved into the exploration of multi-task and Multi-label models, seeking to identify new ideas and innovative insights to address our specific problem. This process of evaluation and research led us to the evolution of methodologies and the discovery of promising approaches.

At this point, our thesis work focused on finding a solution that reconciled the need for advanced models with the limited availability of adequate resources. For this purpose, we used the RSCD Dataset as a fundamental element of our investigation, providing a promising data foundation for our studies. This dataset, among those considered in this phase, emerged as the most relevant to our objectives.

Finally, we analyzed in detail the main model used to address the problem in question, based on this dataset, identifying it as the baseline to start from.

Personal contribution of the candidate to the solution of the problem described

The candidate's personal contributions aim to enhance the current state of the art by addressing and mitigating various challenges highlighted by the state-of-the-art analysis, with a particular emphasis on data-related issues. These issues encompass both the significant class imbalance in the available datasets and the likely shortage of information necessary to successfully tackle this intricate problem.

To effectively address these issues, several steps were taken. Firstly, after a thorough analysis, the RSCD dataset was restructured. This step was pivotal in making the data more suitable for our research needs and alleviating issues related to correlations among the different sets of training, validation, and test data.

Secondly, an innovative model was developed that combines a Convolutional Neural Network (CNN) based on Efficient Net B0 with a Recurrent Neural Network (RNN), specifically a Long Short-Term Memory (LSTM). This architecture was designed with the goal of efficiently capturing spatial and temporal correlations in the data. The integration of CNN and RNN allows the model to make the most of the data, further refining our ability to address the problem at hand.

Description of the application/experimental content of the paper

The experimental journey was a gradual process that involved the analysis and progressive implementation of various solutions in comparison to a baseline established by the reference paper. Initially, we initiated an experimental investigation to assess the importance of data reorganization concerning the RSCD dataset. This initial step was crucial in highlighting the implications of proper data management. It became evident that the same model trained with the initial data organization produced unreal performance, entirely different from what was achieved by simply reorganizing the data in a less correlated manner.

Subsequently, starting from a single-task model, several modifications were introduced during the course of the experiments, leading to the evaluation of the performance of the complex model introduced earlier. This sequence of steps allowed us to examine the evolution of the model's performance throughout the experimental process.

The results obtained clearly demonstrated a significant improvement in the final model, which managed to achieve a more robust balance on both the classical metrics introduced and the balanced ones. This highlights the success of the solutions proposed during the experimentation and the significant progress made compared to the reference baseline.

Contents

Chapter 1

Introduction

The automotive industry stands at the forefront of technological innovation, thanks to the seamless integration of machine learning technologies into advanced driver assistance systems (ADAS). These systems, powered by neural networks, have heralded a transformative era, significantly enhancing vehicle safety and driver control. In their current commercialized form, ADAS systems adeptly detect and respond to diverse road conditions, ranging from lane markings to potential hazards, redefining the driving experience.

Nevertheless, the quest for innovation is unceasing. Ongoing research in this domain is driven by a vision of expanding the horizons of ADAS capabilities. Beyond the familiar challenges of lane keeping and obstacle avoidance, researchers and engineers are now striving to equip ADAS with the intelligence to classify road surface conditions, interpret weather variables, and discern signs of road degradation. These enhancements are pivotal in enabling ADAS to adapt driving behavior dynamically, fostering both safety and efficiency.

As we embark on this journey of exploration, we delve into a realm where neural networks take center stage, particularly multi-task neural networks. These networks introduce a compelling dimension to AI systems, enabling them to tackle not just one, but multiple related tasks simultaneously. This innovation holds immense potential for ADAS and a myriad of other AI applications.

Multi-task learning, represent a paradigm shift in AI. These networks have the remarkable ability to handle a bouquet of interconnected tasks while drawing upon shared knowledge and representations. Unlike their single-task counterparts, which specialize in a singular function, multi-task networks thrive on diversity and

adaptability.

In the context of ADAS, where the road environment presents multifaceted challenges, the utility of multi-task learning becomes evident. Traditionally, individual models were designed to address specific aspects, such as road surface conditions or material recognition. However, these segregated models often lack the holistic perspective required for robust decision-making.

Multi-task networks change this landscape by offering a unified approach. They synthesize insights from multiple tasks, providing a comprehensive understanding of the road environment. This not only enhances the quality of decision-making but also minimizes computational overhead.

Efficiency is a hallmark of multi-task learning. These networks are adept at optimizing resource utilization, making them ideal for deployment in resource-constrained environments, such as onboard vehicles with limited computational power. By sharing computations and representations across tasks, multi-task networks maximize real-time inference capabilities. Furthermore, multi-task learning excels in data efficiency. In domains like ADAS, where acquiring labeled data for each specific scenario is arduous, multi-task networks can be trained on shared datasets. This reduces the demand for vast quantities of annotated data and eases the data collection burden.

The primary dataset for this research is the recently introduced RSCD dataset, comprising one million labeled images for three separate tasks. The first task involves estimating road surface friction related to weather conditions, classified into six categories: dry, wet, water, fresh snow, melted snow, and ice. The second task focuses on recognizing the material of the road surface, with images representing asphalt, concrete, mud, and gravel. The third task is the identification of road surface irregularities due to material degradation, including smooth surfaces and those with mild or severe undulations.

In this thesis, we embark on a quest to harness the potential of multi-task neural networks in the realm of ADAS, particularly in the context of road surface classification. By leveraging state-of-the-art neural network architectures and training methodologies, we aim to demonstrate how multi-task learning can amplify the capabilities of ADAS systems, fostering safer and more efficient driving experiences.

The subsequent chapters will delve into the technical intricacies of multi-task network design, training strategies, and comprehensive performance evaluations. Through empirical studies and real-world scenarios, we will uncover the practical benefits and challenges of integrating multi-task learning into the fabric of ADAS.

In conclusion, the fusion of AI and multi-task neural networks not only augments the capabilities of ADAS but also opens new frontiers in the broader landscape of artificial intelligence. As we embrace the power of multi-task learning, we anticipate breakthroughs that will redefine the boundaries of efficiency, resource optimization, and data utilization across diverse AI applications.

Examining the chapters that will comprise this thesis, Chapter 2 focuses on the state of the art related to the problem under study, delving into both the available datasets and the most commonly used methodologies. In Chapter 3, innovative contributions in this research field will be scrutinized, both concerning the analysis of data used to train the employed models and the development of the final model. Lastly, in Chapter 4, the conducted experiments will be presented along with all the observations derived from them.

Chapter 2

State of the art

From classical advanced driver assistance systems (ADAS), the seamless integration of machine learning technologies has ushered in a new era of innovation, steering research towards autonomous driving. As highlighted in the introduction, the relentless pursuit of safety and efficiency by the automotive industry has been significantly advanced by these developments. However, the quest for further improvements remains unceasing. This chapter embarks on a journey to explore existing datasets, labeling standards, and the most commonly used and promising machine learning models for road surface classification, an increasingly crucial issue for autonomous driving and addressed in this thesis.

2.1 Evolution of Machine Learning in Road Surface Classification

The understanding of information pertaining to road surfaces, particularly the classification of pavement types, the assessment of their condition, and the detection of any anomalies, is a relatively recent field of study. This field holds significant potential for improvement but is concurrently characterized by a lack of robust solutions, primarily due to the limited availability of suitable datasets. In the following sections of this chapter, we will begin by thoroughly examining the status of existing datasets in the literature and subsequently delve into the techniques employed to address this complex challenge.

2.1.1 Datasets

In the context of such a heterogeneous problem as road surface classification, there are primarily two types of datasets with very different applications: those acquired from fixed cameras and those obtained from onboard vehicle cameras. For the purposes of this thesis, we focus on evaluating state-of-the-art datasets collected from onboard vehicle cameras. In this field, the availability of a suitable, extensive, and robust dataset is of paramount importance. This is because this problem encompasses a wide range of weather and landscape conditions, and the quality of training data plays a crucial role in the effectiveness of machine learning algorithms.

A critical aspect in training deep neural networks is the availability of annotated training data that fairly represents the various classes of road surfaces. Road surface classification presents a particular challenge due to the class imbalance problem, where some classes may be overrepresented (majority classes) while others are underrepresented (minority classes) in the dataset. This imbalance can lead to a significant degradation in classification performance. This problem is especially relevant when applying convolutional neural networks to road surface classification.

Although there are numerous datasets available for general image classification or autonomous driving in general, such as BDD100K [1] or Waymo [2], specific datasets for road surface classification are scarce. Using generic datasets for autonomous driving can lead to highly imbalanced datasets, as most of the images are recorded under sunny or cloudy conditions to ensure better lighting and reduce optical occlusions due to rain on the windshield. Additionally, most of the road surfaces represented in these datasets consist of smooth asphalt, while surfaces like dirt roads or sand are not represented, as they do not fall within typical urban road scenarios.

In the following section, we will analyze various state-of-the-art datasets in this field, such as the Assessment of Deep Convolutional Neural Networks for Road Surface Classification Dataset, GAPs v2, and the Road Surface and Wetness Dataset. These datasets were among the first to be published but have several shortcomings in terms of robustness. Finally, our focus will be on the RSCD Dataset, which is undoubtedly the most promising and the most recent among those listed.

Assessment of Deep Convolutional Neural Networks for Road Surface Classification Dataset

This dataset is described in [3], who represents one of the initial attempts to systematically address this challenge (2018). To create a balanced and representative dataset, images from various sources were utilized. This approach offers the advantage of covering various camera configurations, thus avoiding the risk of training models specific to a particular setup. Furthermore, care was taken to ensure that the images maintained a perspective similar to that of a windshield-mounted camera, avoiding artificial distortions.

The main sources of images included the NREC Human Detection & Tracking in Agriculture dataset [4], the KITTI [5] Vision Benchmark Suite, the Oxford Robocar Dataset [6], the New College Vision and Laser Data Set [7], a dataset of images published by other authors [8], and image sequences captured in the Stadtpilot project by the research vehicle Leonie [9]. However, a preliminary analysis of these datasets revealed a significant class imbalance between the majority class, asphalt, and the minority class, cobblestone, with a ratio of 10:1.

To address this class imbalance challenge, instead of resorting to over-sampling or under-sampling techniques, additional images were added through a Google search, following a similar approach as used for fine-grained image classification.

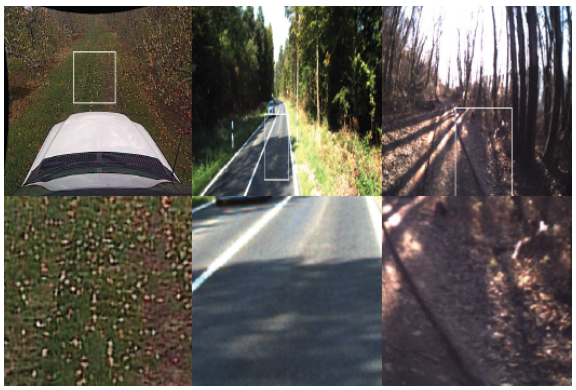

Figure 2.1: Some example images from the dataset. From [3]

Furthermore, the selection of training and test data was done considering that the datasets provided sequences of frames rather than a random collection of independently recorded frames. This allowed for maintaining consistency in road conditions across frames within the same sequence. The final test set consisted of

300 images per class for all datasets, while the remaining images were used to create three different training sets, each with specific considerations regarding class balance and variations in road conditions.

GAPs v2 dataset

In 2019, the paper titled "Improving Visual Road Condition Assessment by Extensive Experiments on the Extended GAPs Dataset" [10] introduced the GAPs v2 dataset, representing a significant advancement in the field of road surface classification. This dataset emerged as one of the most extensive and comprehensive ones in the domain of road pavement imperfections, offering high-quality and standardized images.

The creation of GAPs v2 involved expanding a previous version of the dataset by adding 500 images from an additional federal highway. This integration increased the total number of high-definition road surface images to 2468.

A significant improvement was made in data annotations, achieving greater precision through the use of smaller bounding boxes tightly encircling imperfections and a rigorous verification of annotations by experts. This attention to detail contributed to making the GAPs v2 dataset an important resource for training and evaluating machine learning models.

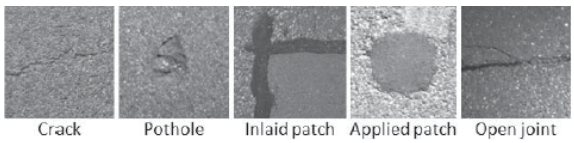

Figure 2.2: Surface defect classes in GAPs dataset. From [10]

Additionally, a subset of the dataset containing 50,000 images was created, designed for rapid experiments. This approach was inspired by the well-known MNIST and CIFAR datasets and was conceived to quickly assess the performance of classification models. The effectiveness of this subset was demonstrated in transferring knowledge gained from training models to identical model configurations trained on the full dataset.

The creation of GAPs v2 was motivated by the need to address challenges related to class imbalance and improve annotation accuracy, making this dataset a

valuable resource for research in the field of road surface classification.

Road Surface and Wetness Dataset

The "Road Surface and Wetness (RoadSaW)" dataset [11] represents a significant contribution in the field of road surface classification and humidity evaluation. This extensive dataset was created through recordings made on a test track, using a camera mounted on a truck and Lufft's MARWIS device, specialized in measuring the height of the water film on the road. To simulate various humidity conditions, controlled sprinklers were used on three types of surfaces: asphalt, basalt (pebbles), and concrete. The data was collected over 10 different days, generating a comprehensive dataset designed for road surface classification, humidity analysis, and uncertainty estimation.

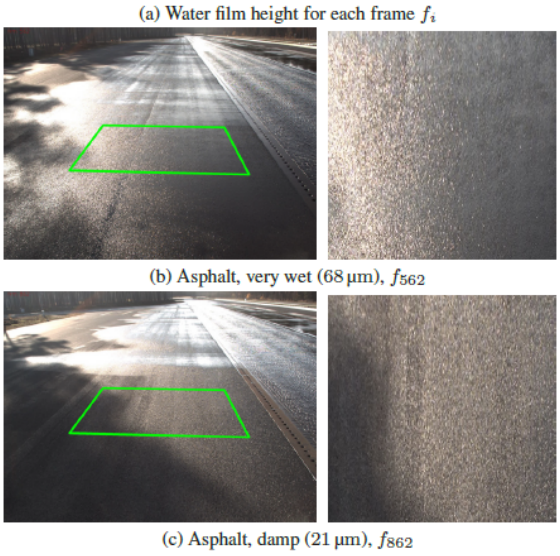

Figure 2.3: The ground truth measurement for the water film height (a) for the selected ROI in the camera images (b), (c). The image patches on the right are derived from the input images (left) using camera calibration. Horizontal gray lines in (a) indicate the proposed split in four levels of wetness dry, damp, wet, very wet. From [11]

During the recordings, a series of driving maneuvers were performed on all surfaces, covering a range of humidity levels. In addition to measurements taken at constant speeds, accelerations and decelerations were included, with speeds reaching up to 80 km/h. Examples illustrating dry and wet conditions are provided in

Figure. Furthermore, "Close-to-Distribution (CtD)" datasets were created to assess the generalization capability of the analyzed approaches.

Regarding image acquisition, the cameras (FLIR Blackfly 3.2MP, 30 fps) were positioned behind the truck's windshield at a height of 2.66 meters and on the car at 1.26 meters. The base dataset was built using data recorded on the truck, while the "Close-to-Distribution" dataset used data recorded on the car. To ensure accurate results, the cameras were meticulously calibrated using 3D measurements and manually annotated 2D positions, minimizing mapping errors. This calibration allowed for the association of the camera image with the 3D world, enabling the generation of a Bird's Eye View (BEV) of the road surface in every visible position. Temporal synchronization with the vehicle-recorded data (MARVIS and speed) was achieved through timestamps. The RoadSaW dataset includes three different patch sizes (2.56 m^2, 7.84 m^2, 12.96 m^2) extracted at four different distances from the vehicle (7.5 m, 15 m, 22.5 m, 30 m). This variety of sizes and distances was considered to evaluate how the amount of texture influences the estimates.

However, it is important to note that, since the recordings were made on the same road surfaces under controlled irrigation, the generalization capability of any models to real-world conditions on different surfaces and under various conditions may be limited.

RSCD Dataset

To achieve the previously discussed objectives, it is crucial to have access to accurate information regarding the level of friction and the condition of the road surface, including factors such as whether it is wet or dry, as well as any potential irregularities in the terrain. This data helps define the safety limits of the vehicle and enhances its real-time dynamic control capabilities. Furthermore, for autonomous vehicles, increasing accuracy becomes even more important as they require advanced perception of road conditions for trajectory planning and path control.

To address these needs, the RCSD (Road Surface Classification Dataset) was created specifically to overcome the limitations of existing datasets. This dataset provides an extensive collection of road images captured in real driving conditions, covering a wide range of road materials, weather conditions, and levels of brightness.

The RCSD consists of one million patches, each with detailed labels related to friction level, irregularities, and road material type.

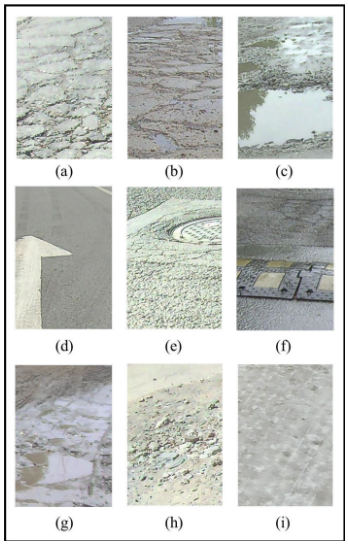

Figure 2.4: Example images from the RSCD dataset. From [12]

Notably, this dataset, released in 2023, used vehicle-mounted cameras for image acquisition, ensuring consistency with the data available in real-world scenarios. Additionally, it is the first dataset to provide simultaneous annotations for friction level, irregularities, and the properties of road images, demonstrating a precise and comprehensive definition for each road property.

The dataset encompasses a wide range of operating conditions, ensuring that algorithms developed using RCSD are robust and applicable to practical scenarios. Deep learning experts can use this dataset as a reference for comparing the performance of various image classification algorithms.

The data description of RCSD reveals its comprehensive nature. Road images are annotated under various weather conditions (dry, wet, water, fresh snow, melted snow, and ice) for the friction level property, and road materials (asphalt, concrete, mud, and gravel), as well as road irregularities categorized as smooth, slightly uneven, and severely uneven based on road undulation amplitude. The dataset defines 27 classes, including various combinations of these properties.

It is important to note that when friction levels are represented by fresh snow, melted snow, or ice, road material and irregularities are not annotated. Similarly, irregularities are not labeled for muddy or gravel roads.

The dataset was described in the article [13] and consists of approximately one million images, extending the initial version of 370,151 described in [12]. The training images are in .jpg format, each sized at 240 × 360 pixels with a resolution of 96 dpi. To evaluate the developed algorithms, the initial version included a test set comprising 12 continuous sequences of 30 images, each with dimensions of 1920 × 1080 and eight patches within each labeled image. In the final version, the dataset is divided into train, validation and test with a static split, in a randomized way.

The data acquisition process involved a LI-USB30-AR023ZWDRB USB camera mounted on the vehicle's bonnet, capturing images from real roads. The camera's parameters are detailed below:

Camera		**Lens**	
Resolution	1920 x 1080	Aperture	1:1.4
ISP	AP0202	Focal lenght	6mm
Sensor size	1/2.7"	Mount	CS
Frame size	30 fps	Distortion	0.35%
Max dynamic range	105 db		

Table 2.1: Camera and lens parameters used for acquisition in RSCD Dataset [13]

Data collection spanned from October 2021 to May 2022 in Beijing, covering various weather conditions, sunlight brightness levels, road service ages, aggregate characteristics, and driving operations. The dataset intentionally includes corner cases such as debris on the road, dirt on the lens, and camera motion blur to encompass as many realistic situations as possible, making it a solid foundation for practical driving assistance applications.

Considering that the tire's path significantly influences vehicle response, the original images were cropped into patches measuring 240 × 360 pixels to facilitate accurate road classification. This cropping and selection process was executed through a Python script and followed by manual image classification.

The strengths of the RCSD dataset lie in its comprehensive annotation of road properties, wide range of operating conditions, and potential for benchmarking image classification algorithms. However, it is essential to consider the limitations, including the limited choice of road routes (all near Beijing) and concerns about

data quality. Since each patch must add information not provided by the previous ones, some patches may be genuinely important in smaller numbers.

In conclusion, RCSD represents an important resource for research and development of driving assistance systems and autonomous vehicles. However, it is crucial to account for the challenges associated with training and validating models, as well as the wide variability of real-world road conditions.

2.1.2 Techniques

Throughout our journey of exploring the challenges related to road surface analysis, we have outlined the complex requirements associated with classifying road surface types, assessing their condition, and detecting any anomalies, while also evaluating the major existing datasets. With a deeper understanding of the data possibilities, we now examine the techniques and methods currently in use to address these challenges. This section will provide a comprehensive overview of these approaches, categorizing them into 'effect-based' and 'cause-based' strategies. We will delve into their characteristics, advantages, and limitations, aiming to offer a complete picture of the available methodologies and the most relevant research directions in this context.

Effect-based

Methods based on effects were the first to be experimented with, leveraging signals from onboard sensors already present on vehicles as their main input. These sensors, including accelerometers, gyroscopes, wheel speed sensors, and more, provide valuable data about the vehicle's dynamics and interactions with the road surface during driving.

The key element of this approach is the reverse estimation of road parameters through the use of state observers. In practice, these observers, relying on real-time acquired sensor signals, work intelligently to extract information about the road surface, such as terrain type, friction level, and the presence of any irregularities.

This means that, instead of directly measuring road surface characteristics, such as the coefficient of friction or irregularities, effect-based methods exploit the relationship between the vehicle's dynamic behavior and road conditions. This

approach offers several advantages, including the ability to utilize existing sensors without the need for expensive additional equipment.

Two notably relevant approaches employ state observers based on the Kalman filter and its advanced variants as a fundamental tool.

In the first work [14], titled "Algorithms for Real-Time Estimation of Individual Wheel Tire-Road Friction Coefficients," the authors propose a sophisticated approach for real-time estimation of the friction coefficients of individual wheels of the vehicle. Their method is structured into three main phases:

1. Estimation of the longitudinal tire force on the wheel: This step aims to calculate the force acting in the longitudinal direction on the tire of each wheel of the vehicle.

2. Measurement or estimation of the longitudinal slip ratio, which represents a critical measure of the relationship between the wheel speed and the vehicle speed. Three different algorithms are employed based on the type of available sensors: the first one utilizes engine torque and brake torque measurements available through the vehicle's CAN network and the inertial vehicle speed obtained from a carrier-phase-based GPS system; the second one uses engine torque, brake torque measurements, and a longitudinal accelerometer; the third one omits engine torque measurements, relying solely on a longitudinal accelerometer and the vehicle speed derived from GPS. In all three cases, it is assumed that wheel speed measurements are available.

3. Utilization of a recursive least-squares parameter identification algorithm: The final step employs a parameter identification algorithm to calculate the road friction coefficient for each tire. This process is based on data obtained in the first two steps and allows for precise estimates of the friction coefficients.

The second one [15], is titled "Estimation of Tire-Road Friction Coefficient based on Combined APF-IEKF and Iteration Algorithm." In this work, the authors propose an innovative method that combines APF-IEKF with an iteration algorithm based on SAT to estimate the tire-road friction coefficient using existing sensors. This study evaluates how the use of APF (Adaptive Particle Filtering) for state estimation addresses the issue of particle degeneracy, as it takes into account the

most recent observation and removes non-Gaussian noise, thus providing a preliminary value for the slip angle. Subsequently, the IEKF (Iterated Extended Kalman Filter) optimizes the estimation obtained from APF. This process allows for a more precise estimation of the tire slip angle. Furthermore, the iteration method is employed to estimate the tire-road friction coefficient based on SAT (Self-Aligning Torque) and a modified brush tire model. This approach is straightforward and can identify the tire-road friction coefficient before tire force reaches saturation.

In the article, the vehicle model and tire model are first presented to introduce the combined estimation method. Subsequently, the combined APF-IEKF and iteration method are detailed. Finally, the proposed method is validated through simulation tests conducted in Carsim/Simulink and experimental vehicle tests on snow-packed terrain.

The previously mentioned solutions require a complex dynamic modeling of vehicles and tires to achieve highly accurate estimates. This approach can pose significant challenges when the vehicle is not subjected to sufficient stimuli. Alternatively, machine learning-based approaches have been developed, such as the use of artificial neural networks (ANNs), which enable the automatic extraction of vehicle response characteristics without the need for complex modeling. These approaches also offer greater robustness against noise.

A notable example of this approach has been described in a specific scientific study. In this study, various onboard sensors, including the IMU, were used to feed a neural network based on LSTM. This neural network was employed to extract and select road characteristics. The results obtained through this method demonstrated a certain level of robustness, with a classification accuracy of 94.6% on four different types of roads. This evaluation was conducted through experiments carried out on real vehicles [16].

Furthermore, ensemble learning techniques were applied, involving the stacking of multiple multi-layer LSTM networks to further enhance the precision and stability of the model. The output of the ensemble network was calculated as the average of the outputs of individual neural networks, contributing to more reliable and generalizable results.

Cause-based

Although methods based on the mentioned effects do not require additional sensors, they inherently exhibit latency as they rely on passive responses measured by onboard sensors. In recent years, attention has shifted towards cause-based methods, which typically utilize actively collected signals from various types of sensors.

For example, in [17], the challenge of road surface classification using reflected ultrasonic waves is addressed. To achieve this goal, a Deep Neural Network (DNN) with 1D convolution layers for feature extraction is employed, followed by a multi-layer perceptron (MLP) for classification. The preference for 1D Convolutional Neural Networks (CNNs) over 2D CNNs is due to the narrow temporal dimension of the data to mitigate the risk of overfitting. The D-CNN architecture is designed with a few lightweight common layers, ensuring simplicity and efficiency. Training is based on a softmax cross-entropy loss function, and the Adam optimizer with dropout is used for regularization.

In the context of [18], an evaluation of road pothole detection using cameras was conducted. This research introduced and tested various models, including model LM1, capable of accurately detecting potholes even in adverse weather conditions, and model LM2, specialized in real-time detection. Furthermore, the SV2 approach was introduced, which proved to be highly accurate in detecting potholes and road imperfections when combined with stereoscopic vision cameras. A notable feature of SV2 is its ability to track a pothole from one frame to another, along with its relatively simple implementation. Model LM1 is based on Mask R-CNN and leverages transfer learning using pre-trained weights from a comprehensive dataset like COCO to accurately identify pothole shapes. On the other hand, model LM2 specializes in real-time detection using YOLOv2, making it suitable for advanced driver assistance systems. A significant aspect highlighted by this research is the challenge of field truth annotation for potholes, given their irregular shapes. It is noted that there is currently no standardized platform or benchmark for pothole identification. To address this challenge, a set of six specifically designed datasets for pothole identification was created as part of this work. Since these methods are independent of the specific vehicle dynamics, performance improvement is expected

in anticipation of the future roads for smart vehicles.

With the rapid development of smart vehicles and sensor technology, cameras have gradually become a standard for perceiving the driving environment. Many studies have demonstrated that road surface perception based on vision to power deep neural networks (DNNs) is a promising method and one of the best among cause-based methods, as demonstrated in this research [19] on road surface classification using convolutional neural networks, with a focus on two main architectures: InceptionV3 and ResNet50. The research examines training and classification performance on various datasets, including baseline data, data enriched with images from Google searches, and extended data for all classes. The results indicate that InceptionV3 achieved good training performance but exhibited overfitting when classes were extended. ResNet50 required more training time but produced better results, especially with class extensions. The analysis emphasizes the importance of data balancing to achieve good classification performance. Furthermore, confusion matrices are examined, revealing some challenges in road surface classification, such as confusion between "wet asphalt" and "asphalt." The research concludes that the second dataset, balanced with Google images, produced the best results. Additionally, models for pothole detection are presented, underscoring the importance of accurate data annotation to improve the performance of smart vehicles on future roads.

In a different context outlined in the article [20], research was conducted with the aim of considering not only the characteristics of the road surface but also those of the sky and the surrounding environment in order to obtain a complete picture of the situation. In this study, the model analyzes information from various components of the road scene and has demonstrated high accuracy in both training and testing. However, it should be noted that the models developed in this context were trained on a limited set of images from specific road conditions. Therefore, further testing is necessary to assess their robustness and performance in more complex and varied scenarios to ensure the validity of the results obtained.

A more efficient approach was undertaken in the following study [21], where an innovative approach based on the analysis of road images is proposed to improve both efficiency and accuracy in the tire-road friction coefficient estimation algorithm. The authors meticulously extract visual features, such as road color and texture, using

a "hand-crafted" approach from acquired images. These features are leveraged to classify different road surface conditions using a Support Vector Machine (SVM) algorithm. Another significant step in this study is the integration of an interconnected disturbance observer, which plays a crucial role in the process of estimating the maximum friction coefficient between the tire and the road, based on the vehicle's dynamics.

Deep learning requires large-scale data, and the generalization ability of developed models heavily depends on the diversity of the training dataset. Most existing research tests performance in a few specific road conditions and classes, which is not convincing for practical applications. Furthermore, due to the non-convex nature of neural networks, trained models generally reach a local minimum, leading to inherent uncertainty and randomness in the models. Additional strategies are needed to enhance the robustness of developed deep learning algorithms.

Finally, the most promising work is [13], which serves as a starting point for this thesis project, is based on a significantly extensive dataset produced and published not long before [22] by the same authors, on which they proposed an architecture aimed at solving a problem that is remarkably similar to the one addressed in this thesis project.

This scientific work focuses on creating an image classification model, particularly for road surface classification. The model takes images as input and returns probability scores corresponding to each class. The work chooses to use a CNN (Convolutional Neural Network) based model, rejecting the possibility of using transformers due to their difficulty in training and lack of distribution-oriented optimizations, although in recent years, transformers have shown better feature extraction capabilities, achieving state-of-the-art performance in many tasks.

The paper provides a detailed description of the model architecture. It uses EfficientNet-B0 as the backbone for feature extraction from input images and employs a fully connected layer to derive a C-dimensional vector representing the probability for each road class. Here, C represents the number of classes in the dataset, which is 27 for this task.

The work explains the chosen loss function, which is the normalized cross-entropy (CE) loss with softmax operation to obtain the normalized probability vector. However,

it acknowledges that this loss alone may not be sufficient to address the problem of classes with a long-tailed distribution and more severe intraclass tail than interclass. To tackle this issue, a "center loss" is introduced that maintains a proxy for each class, and the loss is represented as the Euclidean distance between the image embedding and the proxy of the actual class. "Label smoothing" is also introduced to handle probability outputs so as not to overly emphasize cases on the boundary between different classes. Image augmentation is proposed to improve the model's generalization ability by manipulating brightness, hue, and contrast to simulate real variations in driving conditions.

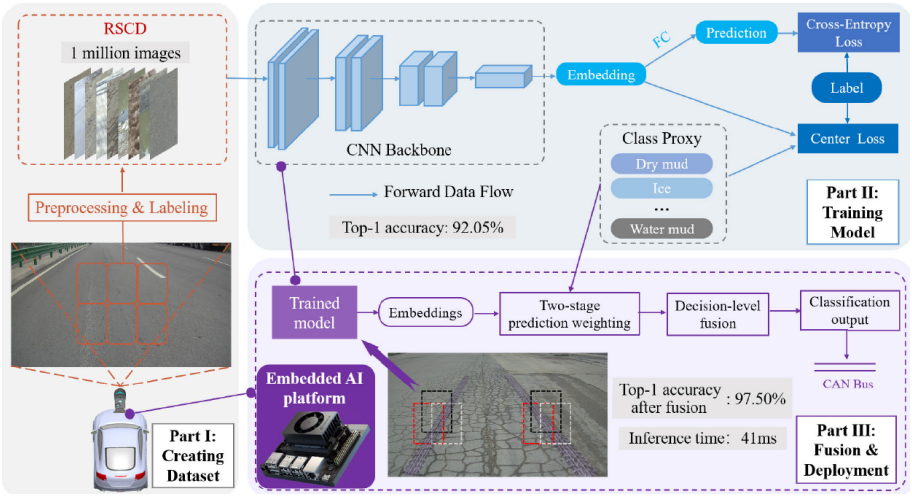

Figure 2.5: The technical framework of this paper. From [13]

Lastly, this work seeks to improve network choice by combining various decisions. Starting from a base image, it produces a series of road "crops," which are subjected to the network, and an algorithm for fusion is proposed to combine the classification results of the various sub-images. A method based on Dempster-Shafer (DS) evidence theory is suggested to combine information from different surrounding areas. This approach takes into account potential conflicts between decisions from different regions and attempts to mitigate them through two phases of classification probability correction. The first phase is based on the distance between the image embedding and class proxies, while the second phase considers the mutual support between different pieces of evidence for each class.

The main challenge of this work lies in attempting to solve the problem as a single task, with a classification into 27 classes, which combines the three initially

listed tasks.

2.2 Multi-Label Learning

In the context of multi-label classification (MLC), the goal is to predict a set of labels associated with a specific input, as opposed to the classic classification where a single label is predicted for the input. The scenarios of multi-label classification are very broad and are found in a wide variety of fields, ranging from protein function classification to document categorization, and even automatic image categorization. For example, an image may have labels like Cloud, Tree, and Sky; the output for a document may cover a range of topics such as News, Finance, and Sports; a gene can belong to functions like Protein Synthesis, Metabolism, and Transcription. Many challenging applications, such as image or video annotation, web page categorization, gene function prediction, and language modeling, can benefit from being formulated as multi-label classification tasks with millions or even billions of labels (in such cases, it is referred to as extreme multi-label classification or XMLC, which is becoming a rapidly growing research area).

The consequence of the above has led to the emergence of diverse needs due to big data and the necessity for new multi-label learning paradigms, resulting in new trends.

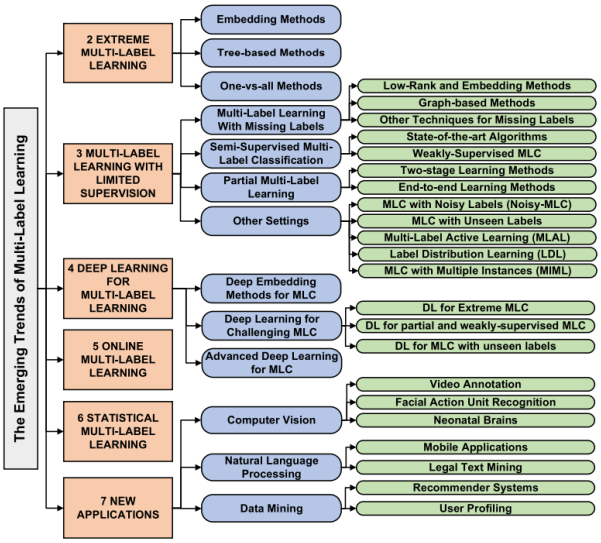

Figure 2.6: Overview of the major developments in multi-label learning. From [23]

Many Deep Learning methods for multi-label learning, thanks to their powerful learning capability, have achieved state-of-the-art performance in many real-world

multi-label applications. For instance, in multi-label image classification, where harnessing the power of deep learning is crucial to better capture dependencies among labels.

In the following, we will provide an overview, first introducing some representative deep embedding methods for MLC, then presenting challenging deep learning for MLC methods, and finally giving an overview of advanced deep learning for MLC.

Deep Embedding Methods for MLC

Unlike conventional methods for multi-label learning, deep neural networks (Deep NN) often explore new feature spaces and employ a multi-label classifier on top. The BP-MLL method [24] was a precursor in using neural network architecture to address the multi-label learning problem. To explicitly capture dependencies among labels within the neural network F, BP-MLL introduces a pairwise loss function for each instance x_i:

$$E_i = \frac{1}{|y_i^1|\,|y_i^0|} \sum_{(p,q)\in y_i^1 \times y_i^0} \exp\left(-\left(F(x_i)^p - F(x_i)^q\right)\right) \tag{1}$$

Where: y_i^1 and y_i^0 represent, respectively, the sets of positive and negative labels for the i-th example x_i, $(F(x_i))^p$ denotes the p-th element of $F(x_i)$, and in the summation, the difference between $(F(x_i))^p$ and $(F(x_i))^q$ measures the discrepancy between the network's outputs and the positive and negative labels. Finally, the exponential function is used to heavily penalize this quantity. Therefore, minimizing (1) leads to higher values for positive labels and lower values for negative labels.

However, it's important to note that BP-MLL has been shown not to perform as expected on text datasets [25]. To address this issue, a neural network (NN) based approach inspired by BP-MLL but adopting a more conventional cross-entropy loss function was proposed. Additionally, various NN techniques, such as rectified linear units (ReLUs), dropout, and AdaGrad, have been introduced, which can be effectively used in this setup.

Despite the effectiveness of embedding methods in capturing label dependencies and reducing computational costs, it's important to note that such methods are shallow models that may not be powerful enough to discover higher-order dependencies among labels. To bridge this gap, [26] introduced the Canonical Correlated AutoEncoder

(C2AE), the first deep neural network-based embedding method for large-scale multi-label classification. C2AE simultaneously seeks a deep latent space to embed instances and labels using two modules: deep canonical correlation analysis (DCCA) and an autoencoder, and predicts label correlations through a suitable loss function placed at the decoder output. C2AE seeks three mapping functions: F_x for features, F_e for encoding, and F_d for decoding. During training, C2AE associates instances X and labels Y in the latent space L and reconstructs them using the autoencoder, with the goal of minimizing the following objective function:

$$\min_{F_x,F_e,F_d} \left(\Phi(F_x, F_e) + \alpha\Gamma(F_e, F_d)\right) \tag{2}$$

Where $\Phi(F_x, F_e)$ and $\Gamma(F_e, F_d)$ represent the losses in the latent space and the output space, respectively, while α is used to balance the two terms. $\Phi(F_x, F_e)$, inspired by CCA, can be defined as follows:

$$\begin{aligned} &\min_{F_x,F_e} \|F_x(X) - F_e(Y)\|_F^2 \\ &\text{s.t.} \quad F_x(X)F_x(X)^T = F_e(Y)F_e(Y)^T = I \end{aligned} \tag{3}$$

On the other hand, $\Gamma(F_e, F_d)$, inspired by [24], is defined as follows:

$$\begin{aligned} \Gamma(F_e, F_d) &= \sum_{i=1}^{N} E_i \\ E_i &= \frac{1}{|y_i^1|\,|y_i^0|} \sum_{(p,q)\in y_i^1 \times y_i^0} \exp\left(-\left(F_d\left(F_e(x_i)\right)^p - F_d\left(F_e(x_i)\right)^q\right)\right) \end{aligned} \tag{4}$$

Where N represents the number of instances, $F_d(F_e(x_i))$ is the label reconstructed from x_i using the autoencoder. During testing, given an input instance $\hat{x}$, C2AE makes predictions as $\hat{y} = F_d(F_x(\hat{x}))$.

A subsequent study [27] proposes another deep embedding method, namely Deep Correlation Structure Preserved Label Space Embedding (DCSPE). In addition to DCCA, DCSPE further develops deep multidimensional scaling (DMDS) to preserve the intrinsic structure of the latent space. Finally, DCSPE transforms the test instance into the latent space, searches for its nearest neighbor, and ultimately considers the label of this neighbor as the prediction, using k-nearest neighbors

(kNN). However, since this search becomes computationally expensive as k increases, the need arises to strike a proper trade-off.

To avoid the aforementioned problem, [28] introduces an innovative method called Deep Binary Prototype Compression (DBPC) for rapid multi-label prediction. DBPC compresses the database into a small set of concise binary prototypes and uses these for prediction.

In the realm of multi-label image classification, [29] proposes a unified deep neural network that leverages both semantic and spatial relationships among labels with image-level supervision only. Specifically, the authors introduce the Spatial Regularization Network (SRN), which generates attention maps for all labels and captures underlying relationships among them through learnable convolutions.

Subsequently, [30] presents Adjacency-based Similarity Graph Embedding (ASGE) and Crossmodality Attention (CMA) to capture label dependencies and discover the positions of discriminative features.

Finally, [31], instead of requiring labor-intensive object-level annotations, suggests extracting knowledge from a weakly supervised detection (WSD) task to enhance multi-label classification (MLC) performance. The authors construct an end-to-end MLC framework enhanced by a knowledge distillation module that guides the classification model using the WSD model for object regions (RoIs). WSD and MLC act as the teacher and student models, respectively.

Among the described works, C2AE is a pioneering effort in deep embedding for MLC and has been used in many real-world applications, including multi-label emotion classification. Label correlation is fundamental for MLC, and some objectives, such as (4), have been employed to model label correlations in deep embedding methods for MLC. However, existing research shows that (4) may not be effective in the textual domain. In the future, effectively capturing label correlations will be an important research theme for these methods. Advanced techniques like graph convolutional networks (GCN) and recurrent neural networks (RNN) pave the way for better capturing label correlations and can drive further research in deep embedding methods for MLC.

Deep Learning for Challenging MLC

In real-world applications where multi-label learning involves a very high number of labels (XMLC), achieving good results often proves to be quite challenging. There are various reasons for this, including labels that may be partial, weakly provided, or entirely novel. Furthermore, as the number of labels increases, effectively leveraging the correlations between them becomes increasingly difficult. Below, recent advancements in deep learning to address these complex MLC problems are discussed.

[32] represents the first attempt to apply deep learning to XMLC. It employs a Convolutional Neural Network (CNN) and dynamic pooling to learn text representations. It also utilizes a much smaller hidden layer compared to the output layer for computational efficiency. However, despite being one of the early attempts to mitigate the aforementioned issues, it still struggles with effectively capturing the important subtext for each label.

Subsequently, AttentionXML [33] was proposed, which has two main features: a multi-label attention mechanism with raw text as input, enabling the capture of the most relevant text portions for each label, and a wide and shallow Probabilistic Label Tree (PLT) to handle millions of labels, especially for "tail labels."

Simultaneously, based on C2AE, a new deep embedding method called Ranking-based Auto-Encoder (Rank-AE) [34] for XMLC was proposed. Rank-AE uses an efficient attention mechanism to learn comprehensive representations from various input types, creating a latent space for instances and labels. It then develops a margin-based classification loss that is more effective for XMLC, especially with noisy labels.

[35] empirically demonstrates that DNN-based embedding methods for XMLC are prone to overfitting and exhibit poor performance when tested on different data. Building on this finding, [144] further proposes a new regularizer called GLaS for embedding-based neural network approaches.

[36] fine-tunes a pre-trained deep transformer for better feature representation. It also introduces a new label clustering model for XMLC using the transformer as a neural matcher. With these techniques, top-notch performance is achieved on several widely used extensive datasets.

Recently, [37] develops the DeepXML framework, which can generate a family of algorithms, including four subtasks: intermediate representation, negative sampling,

transfer learning, and classifier learning. This leads to the Accelerated Short Text Extreme Classifier (Astec), which is more accurate and faster than state-of-the-art deepXML on public short text datasets.

Simultaneously, [38], to enhance the performance of these models, attempts to leverage label metadata, such as informative textual descriptions, which are usually ignored by existing methods. This model jointly exploits the metadata-enriched label model and feature representation and accurately predicts millions of labels.

Different from the aforementioned work, this research [39] incorporates label text and correlations between labels. It develops frugal architectures and scalable techniques to train the model with a label correlation graph containing millions of elements.

Inspired by this work, [40] emerges as one of the most intriguing approaches. It utilizes joint document-label graphs that can incorporate various sources, including label metadata. This work also introduces label-specific attention to achieve high-capacity extreme classifiers. The final model proves to be up to 18% more accurate than the state-of-the-art and is trained 2 to 50 times faster on benchmark datasets.

An important observation is that the above-mentioned efforts primarily focus on challenges in the label space in the MLC problem. However, in the real world, there are significant challenges in the feature space as well. For example, some features may disappear or change over time, or the distribution of labels may continuously vary. Addressing challenges simultaneously in both the label and feature spaces is more demanding and can be considered a future research area for the challenging problem of MLC.

Advanced Deep Learning for MLC

Recently, several advanced deep learning architectures have been developed for multi-label classification (MLC) problems.

The first one to be discussed is presented in the following work [41], one of the early endeavors in this direction. The article aims to leverage relationships between labels by learning to split them into a Markov Blanket Chain and then applying a new deep architecture that exploits this subdivision. Several experiments were conducted on various extensive and popular multi-label datasets, demonstrating

significant improvements of this model over the state of the art.

As an alternative to deep neural networks (DNN), [42] proposes deep forest, a learning framework based on an ensemble of tree models, which do not rely on backpropagation. The pioneer in introducing this architecture for MLC is [23], which presents Multi-Label Deep Forest (MLDF), a model designed to address two challenging MLC problems: optimizing various performance measures and reducing overfitting. Extensive experiments show that MLDF achieves the best performance in terms of hamming loss, one-error, coverage, ranking loss, average precision, and macro-AUC on the datasets used for experimentation.

Another highly important development branch involves graph-based convolutional networks (GCN), where various studies have been carried out. For instance, [43] analyzes the importance of these Graph Neural Networks (GNNs) in the field of machine learning. This work addresses the fact that many learning tasks involve structured data in the form of graphs, containing rich relational information among elements. These tasks may include modeling physical systems, learning molecular fingerprints, predicting protein interfaces, and disease classification, all requiring a model capable of learning from graph data.

A work that utilizes these architectures is [44], related to the multi-label classification of images. It initially constructs a directed graph over object labels, uses GCN to model correlations between labels, and maps their representation into interdependent object classifiers.

Similarly, Semantic-Specific Graph Representation Learning (SSGRL) [45] incorporates semantic decomposition and interaction modules to learn and correlate label-specific representations. Correlation is achieved through GCN on a graph constructed based on label co-occurrence.

Lastly, [46], to better inject label information into the base CNN, introduces lateral connections between GCN and CNN at shallow, intermediate, and deep levels.

After discussing the graph theory-based development branch, we will explore another highly promising line of development, which extensively employs recurrent networks.

A noteworthy study is [47], which first utilizes RNNs to transform multi-label classification into a sequential prediction problem. Labels are initially arbitrarily

ordered, and the network learns how to predict the sequence by comparing each of them to a sentence. The main advantage of this approach is its focus on predicting positive labels, which represent a much smaller set compared to the entire set of possible labels.

Concerning multi-label image classification, [48] employs recurrent neural networks (RNNs) through a CNN-RNN-based architecture to more effectively leverage higher-order dependencies among labels in an image. This approach allows the model to learn a joint embedding of the image and labels, characterizing both the semantic interdependence of labels and their relevance to the image. Furthermore, an advantage of using these models is the ability to train the CNN-RNN from scratch, integrating both types of information into a single framework.

Similarly, [49] uses a Long-Short Term Memory (LSTM) sub-network to sequentially predict semantic labeling scores on individual regions. The goal is to capture global dependencies among these regions, and promising results are obtained in large-scale multi-label image classification.

Conversely, [50], instead of using a fixed and static order for labels, assumes a dynamic context-based label order, not requiring a predefined order. It integrates visual attention with LSTM layers for multi-label image classification.

Another development comes from [51], which proposes an approach to multi-label classification that allows for dynamically choosing the label order based on context. The proposed approach consists of two main components: a simple EM-like algorithm that initializes the learned model and a more elaborate reinforcement learning-based approach. Experiments on three public multi-label classification datasets demonstrate that the dynamic reinforcement learning-based label ordering approach outperforms recurrent neural networks with a fixed label ordering, both in terms of bipartition measures and ranking measures on all three datasets.

These studies [47] [49] [50] [51] have shown that MLC can be transformed into a sequential prediction problem using an RNN-based decoder to model label dependencies. However, training an RNN decoder requires a predefined or dynamic label order, which is not directly available in MLC specifications. Moreover, RNNs trained in this manner tend to overfit to label combinations in the training set and struggle to generate label sequences never seen before.

In this article [52], a new framework for MLC is proposed that does not rely on a predefined label order, thereby alleviating exposure bias. Experimental results on three benchmark datasets for multi-label classification show that this method significantly outperforms competitive baselines. Furthermore, it has been found that the proposed approach is more likely to generate unseen label combinations during training compared to baseline models, demonstrating better generalization capacity.

Finally, the most recent and promising study is [53]. In contrast to previous approaches, it introduces two new methods called "predicted label alignment" (PLA) and "minimal loss alignment" (MLA) to address label assignment without imposing a predefined order. These methods dynamically align the order of actual labels based on loss minimization during training. This leads to faster training and eliminates issues like repeated label assignment in the predicted sequence. The proposed method uses a CNN-RNN architecture, where a Convolutional Neural Network (CNN) extracts a visual representation of the image, and a Long Short-Term Memory (LSTM) neural network generates a label sequence based on the extracted representation. The use of LSTM is advantageous because it can handle long-term dependencies in sequential data.

Advanced deep learning architectures have greater capabilities and can be more effective for MLC problems. However, these methods usually contain a large number of parameters and require high complexity in terms of training and prediction costs.

Chapter 3
Original contribution to problem's solution

The current section aims to delineate the thesis's contribution in proposing an innovative solution to the previously defined problem.

3.1 Description of the problem

The main objective of this thesis is the development of an innovative algorithm based on multitask neural networks to tackle the problem of "Road Surface Classification." This includes: estimating the friction related to weather conditions, divided into six categories: dry, wet, water, fresh snow, melted snow, and ice; recognizing the material of the road surface, such as asphalt, concrete, mud, and gravel; and identifying irregularities caused by material degradation, including smooth surfaces with light or severe cracks. Addressing a problem of this nature presents a series of complex challenges, and certainly, the generic nature of the problem must be evaluated. Indeed, roads, weather conditions, and potential landscapes can be incredibly heterogeneous worldwide. Each geographic region exhibits significant variations in road surfaces, featuring different types of materials, including asphalt, concrete, gravel, and even less common surfaces like mud. Furthermore, weather conditions can vary widely, with regions facing tropical, desert, mountainous, or arctic climates.

The combination of these variations results in a vast and diversified range of possible road conditions. For example, in colder regions, roads may frequently be

covered with snow or ice during the winter, while in tropical areas, they may be exposed to heavy rainfall. Moreover, even within the same location, road surfaces undergo degradation over time, with variations that can lead to irregularities, changes in appearance, cracks, or worn-out asphalt, and all these conditions can also appear with different nuances, manifesting profoundly distinct characteristics.

In light of these considerations, the first possible approach to alleviate this problem is to introduce constraints on the system's usage conditions that are appropriate. With this approach, it is possible to start with robust data related to the choices made. Unfortunately, a significant challenge is the scarcity of data suitable for the state of the art in training and evaluating multitask neural networks in this context. Collecting labeled data for each specific scenario, road condition, and material type is a laborious and expensive process. This scarcity certainly represents an obstacle to the development and optimization of artificial intelligence models.

Examples of usage conditions for the proposed system could include the type of camera configuration (onboard or fixed), the maximum vehicle speed, environmental factors to which the system will be subjected, etc. These conditions play a fundamental role in defining the operational scope and limits of the system itself. The choice between an onboard or fixed camera configuration can significantly impact the quality and type of data available for analysis. Onboard cameras offer a dynamic perspective but may be subject to vibrations and obstacles, while fixed cameras provide a stable view but lack the flexibility of onboard configurations; system performance may vary depending on the speed at which the vehicle is traveling. Higher speeds can introduce challenges related to image quality, processing speed, and real-time decision-making. Additionally, environmental conditions such as lighting, weather, and road conditions can influence the quality of the tools used, affecting the accuracy and reliability of predictions. Each of these features, if not assessed and addressed, can create a disconnect between the data used to prepare these systems and the data they will encounter in reality, leading to a profound inadequacy of the system itself.

Finally, from a practical perspective, another challenge is the design of multitask architectures that can fully exploit the correlations between each individual task. This would allow for efficiency improvements compared to creating dedicated architectures

and likely lead to improved performance in a problem where this is essential to increase passenger safety.

In the following chapters of this thesis, we will delve into the technical aspects of addressing these challenges, with a particular focus on the analysis of the prominent dataset for tackling this problem, the RSCD Dataset [13], and the development of a robust architecture aimed at capturing the correlations between various tasks as effectively as possible.

3.2 Data Analysis

For this thesis, the dataset used is the RSCD Dataset [13]. In the next section, a description of the data analysis process will be provided, along with a comprehensive overview to justify the decisions made, taking into consideration their relevance and compatibility with the research objectives. Furthermore, a detailed analysis of the selected datasets will be presented, including a meticulous examination of the classes, as well as a thorough assessment of the quantity and quality of the available samples. These evaluations aim to establish a solid foundation for the subsequent stages of the research, seeking to obtain representative and sufficiently robust data to effectively contribute to the proposed objectives.

3.2.1 Qualitative analysis of data

In this section, we will examine the process of static data splitting used for the dataset presented in the reference paper [13]. Our main objective is to evaluate the presence of any issues among the different data subsets, identifying significant relationships or problematic trends.

Before proceeding, it is important to reflect on the assumptions underlying the data splitting described in the paper. The document emphasizes that this was done randomly, with a percentage of 93%, 2%, and 5% for training, validation, and test sets, respectively. However, there are several significant considerations to be made, especially regarding the nature of the images contained in the dataset.

The first issue arises due to the presence of large groups of images presumably acquired on the same day. From the data, it can be observed that there are 84 days of acquisition, while the total number of patches in the dataset exceeds one million. On a single day, weather conditions tend to be similar, and the various roads traveled are limited, resulting in less diverse labels. This introduces a high correlation between data from the same day, which could compromise the ability to effectively assess the generalization of algorithms.

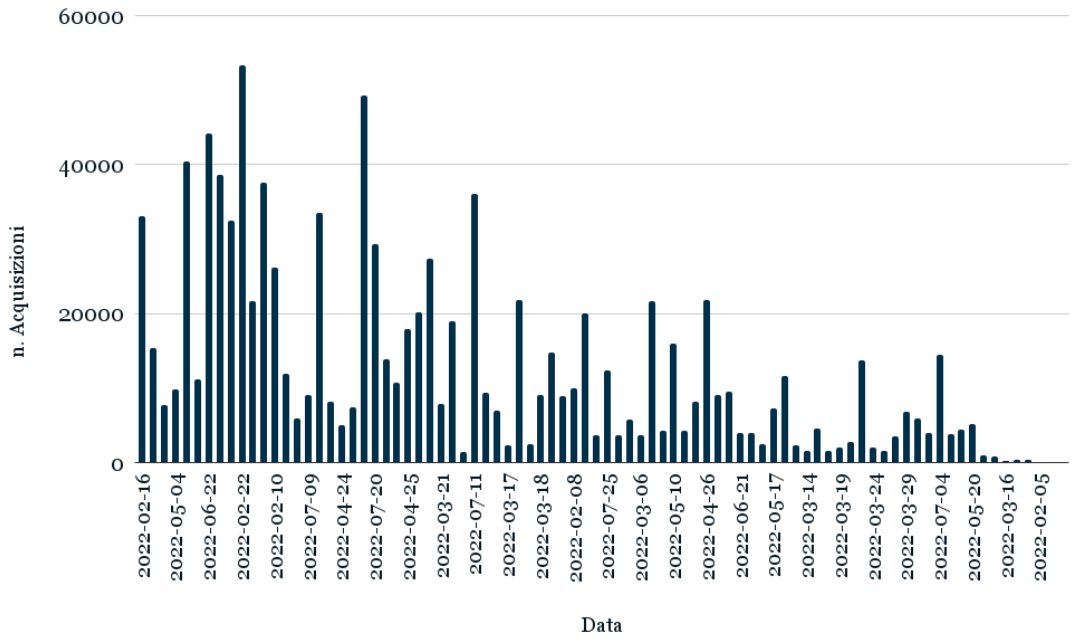

Figure 3.1: Image relating to the number of samples acquired on different days.

Secondly, the images were captured at a sampling rate of 30 fps, resulting in a very high similarity between consecutively captured images. Randomly distributing these images across the training, validation, and test sets can lead to issues in model selection. One might rely on less reliable performance metrics that tend to closely resemble those of the training data, compromising the effectiveness of the validation phase.

Furthermore, it should be noted that the dataset consists of a million patches, not individual images. This means that patches from the same image exhibit an extremely high correlation with each other. This requires particular attention to avoid introducing bias when distributing them among the different sets.

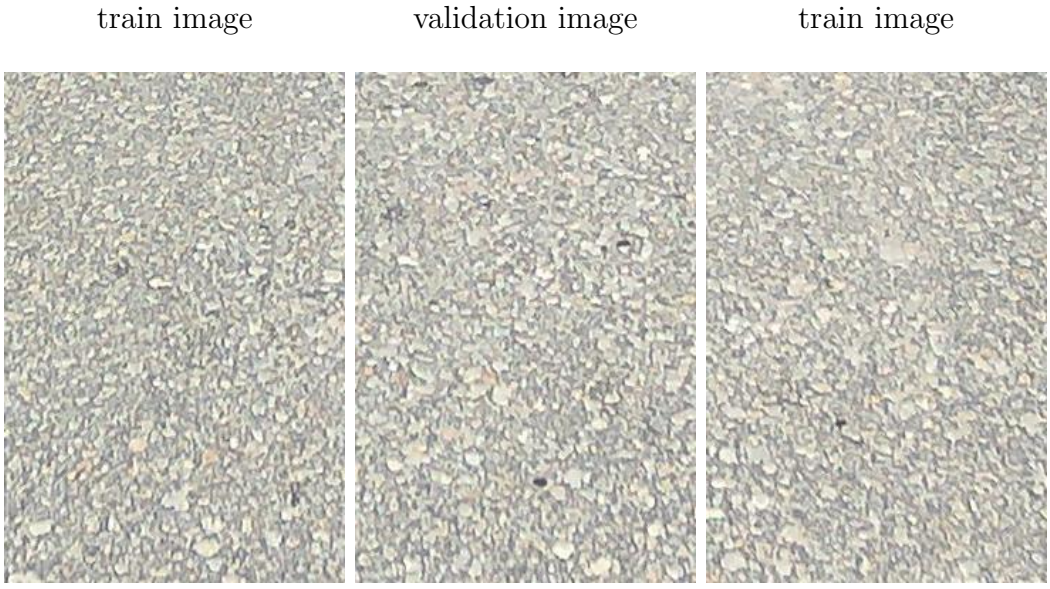

Figure 3.2: Example of successive patches acquired on the same day.

All these challenges highlight the limitations of the data splitting approach adopted in the reference paper. It is essential to explore more suitable solutions to effectively address these issues. Furthermore, considering alternative data splitting approaches that take into account the specific characteristics of the dataset could be beneficial, thereby improving the validity of the results obtained.

Continuing our analysis, we now focus on the data distributions within the various subsets. It is important to note that, although there are three tasks to be solved, these graphs treat the problem as a single task, as was also done in the reference paper [13]. In fact, each class is viewed as a combination of classes from the various tasks. This unified perspective allows us to uniquely analyze the problem, already linking classes from different tasks that will share the same labels during training. Below are the distributions of the three subsets provided by the paper [13]:

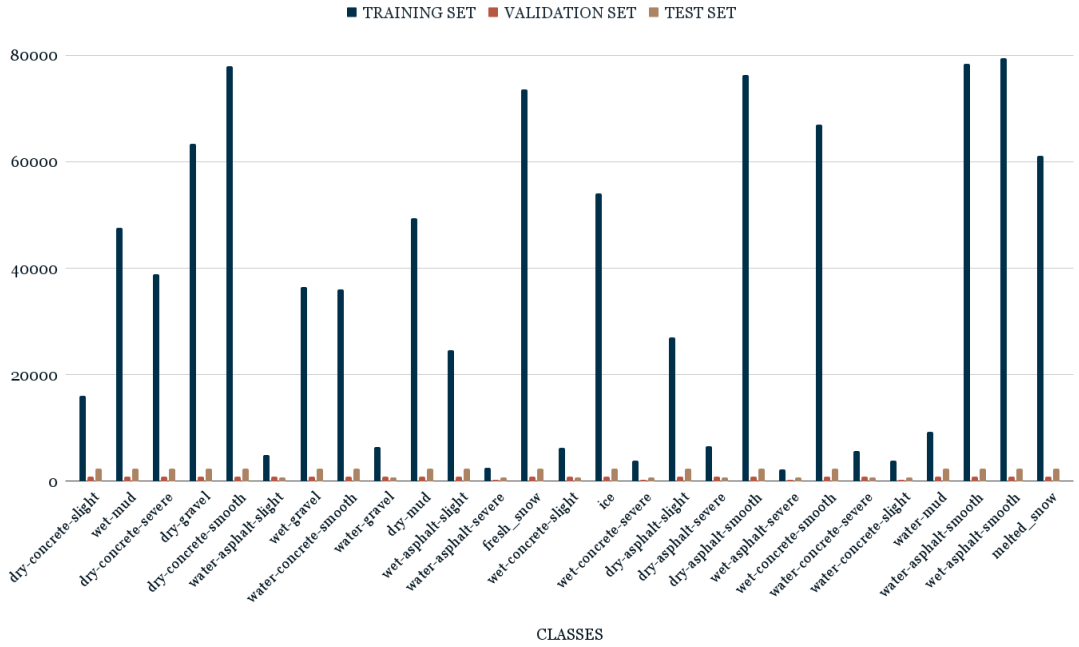

Figure 3.3: Distribution of the training, validation, and test sets using the splitting proposed in the paper [13].

In the beginning, there is a noticeable disparity in the number of images per class in the training, validation, and test sets. This imbalance could pose a significant challenge in making reliable performance estimates for the models, as both the validation and test sets are characterized by a limited amount of data.

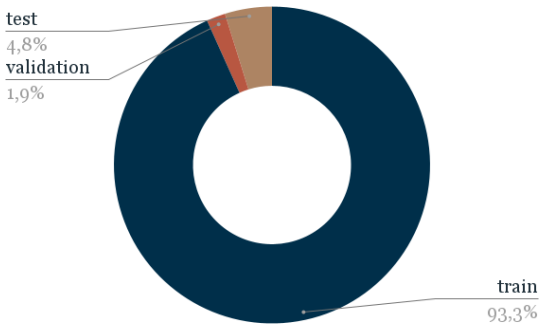

Figure 3.4: Number of samples in the training, validation, and test sets with the split proposed in the paper. [13].

Subsequently, one can notice a positive aspect, the attention paid to the class distribution in the validation and test sets. These sets show a certain balance among the different classes, despite the limited number of samples. This balance is

important because no specific constraints have been established for the conditions in which the system should perform well in the addressed problem. Therefore, having a balanced set for performance evaluation in terms of samples allows for a fair assessment of performance, considering all possible classes equally.

Following all the considerations made earlier, the data split has been modified to limit any issues as much as possible. The solution will be presented below.

3.2.2 Proposal for dataset reorganizzation

In this section, we will describe the choices made to split the data into various subsets to ensure an accurate and balanced representation of the data. In the initial reorganization, the goal was to minimize correlations between the various sets, addressing the issues listed earlier.

Specifically, starting with a single dataset containing all the data, the steps that were taken included:

- Reorganize the data into days: Data acquired on the same day are highly correlated, so it was chosen to organize the data in this way. In particular, patches belonging to the same image will be placed in the same folder, and the same applies to patches from sequential images and, of course, those acquired on the same day.

- Allocate each day, and thus all the patches it contains, into the various subsets: For each day, the entire group of associated images was directed to a specific set. This division of "acquisition days" was carried out while maintaining the percentages established by the reference paper, namely 93%, 2% and 5%, to maintain a certain continuity in the analyses conducted.

Certainly, it should be noted that by dividing these groups of images into various days, although they will tend to respect the percentages defined in parentheses, there is a risk that this may not hold true when considering the percentages of individual images in the various sets. This is because different days, as evident from Figure 3.1, do not have the same number of acquisitions.

Below is the distribution of classes in the new split, obtained after the steps described earlier:

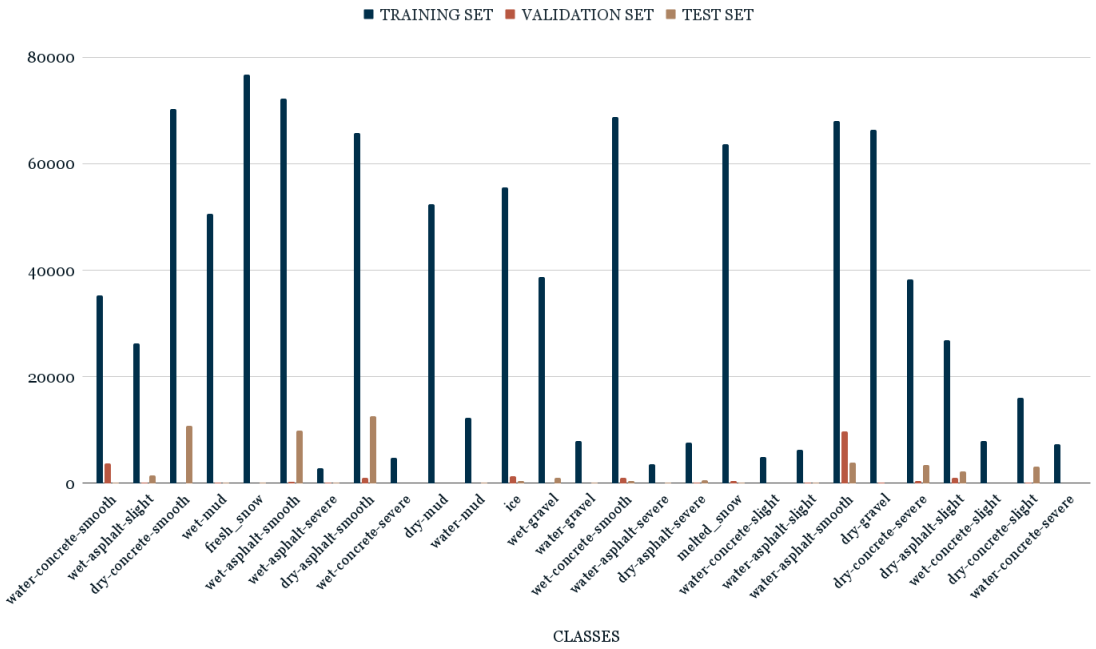

Figure 3.5: Distribution of the training, validation, and test sets using the new split proposed following the analyses done so far.

What is evident from the previous graph is certainly the reduced number of samples in the validation and test sets compared to the training set, as in the split made by the reference paper.

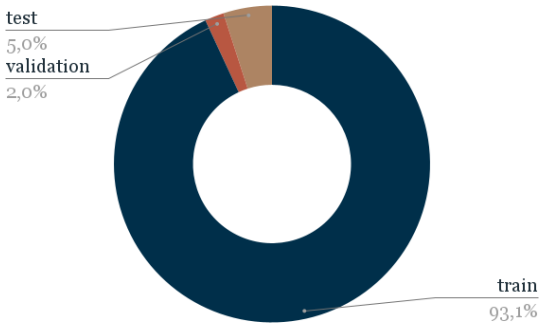

Figure 3.6: Number of samples in the training, validation, and test sets with the currently proposed split.

Additionally, it can be seen that by now dividing into groups of images rather than individual images, the balance in the number of samples per class has been lost compared to what was previously achieved, and some classes even have no images in the validation and test sets. This is a serious issue as it leads to model selection not

being based on all classes, which is dangerous for the goal of uniform performance sought in this thesis.

As a result, we proceeded to modify the split percentages to ensure at least some representation of all classes in all sets. The established split is now 60%, 30%, 10% for the training, validation, and test sets, respectively.

Below is the new and final graph showing the distributions of various classes within the final sub-sets:

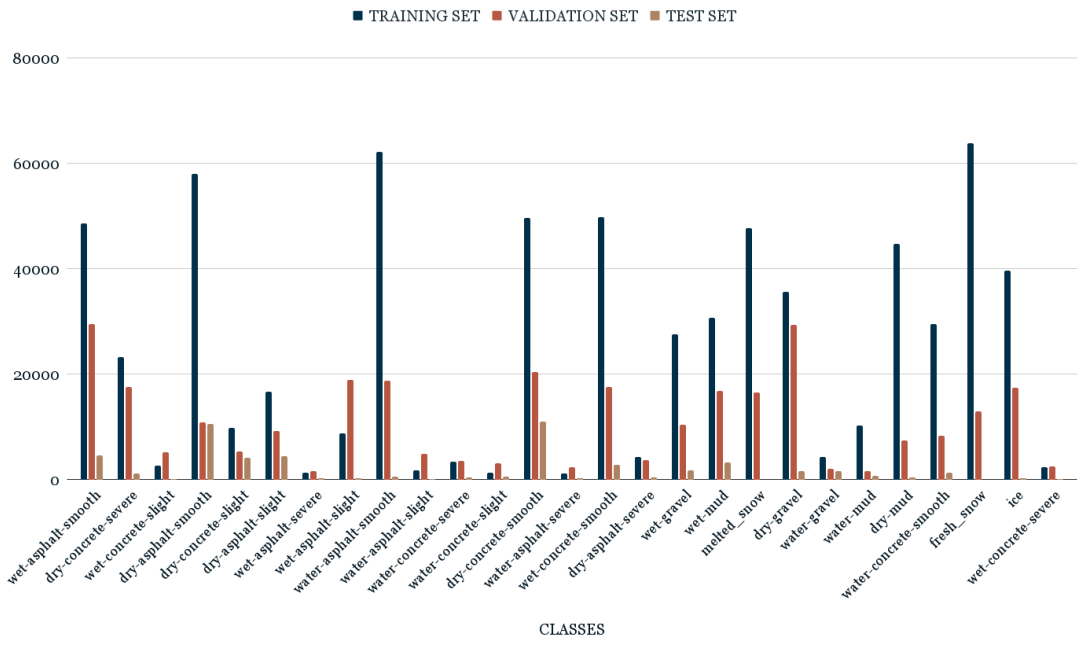

Figure 3.7: Distribution of the training, validation, and test sets using the final proposed split is as follows:

What can be seen from the graph is the smaller gap in the number of samples in the training set compared to the other two, ensuring more reliable performance measurements for various models. Additionally, it can be noted that all classes are represented in the sets, and there is better alignment between the distribution in the training set and the other sets. Maintaining the distribution of the training set may not necessarily be a positive factor given the widely discussed objectives. However, it is possible to use metrics that weigh and balance the imbalance of the various classes, such as balanced accuracy, which will be a key metric considering the new distribution obtained.

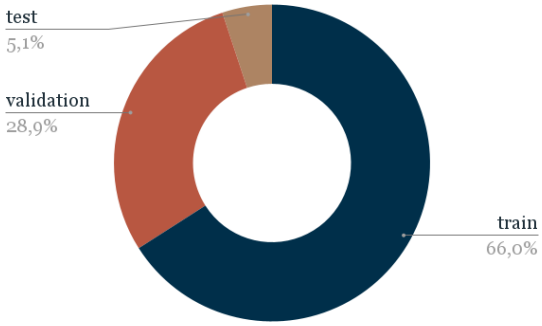

Figure 3.8: Number of samples in the training, validation, and test sets using the final proposed split.

3.3 Proposed Approach Clarification

In the next section, we delve into a detailed exploration of the proposed approach, aiming to provide a comprehensive understanding of its underlying methodologies. There are three sections: the Application Framework, System Design Overview, and a look at Methodological Breakthroughs to provide a complete framework for the proposed methodology.

3.3.1 Application Framework

The Application Framework represents an essential pillar for the proper functioning of the driving assistance system, defining the conditions and requirements that guide its implementation and application. In the design of the Application Framework, it is crucial to consider the following constraints and parameters:

Onboard Vehicle System Placement: The system is designed to be placed on a moving vehicle. This is a necessary condition for solving the problem itself. All the data used to create the system were collected aboard a moving vehicle. This constraint actually complicates the problem, as the system may encounter various complications during operation.

Vehicle Speed: During data collection, the vehicle moved at speeds ranging from 20 to 80 km/h. This constraint is of primary importance, as it affects the dynamics of image acquisition and the system's responsiveness to speed variations. The system must be designed to operate reliably within this speed range.

Camera and Lens Parameters: The specific parameters of the camera and lens used during data collection have been listed in Table 2.1. It is essential that the parameters of the final acquisition system are consistent with those of the camera used, as these directly influence the performance of the system, which otherwise may struggle to make predictions based on images that are not robust.

Onboard Vehicle Device Placement: The dataset was collected using a USB camera mounted on the vehicle's hood. This implies that the system requires a data acquisition device placed in a similar position. This placement can also affect the system's performance.

3.3.2 System Design Overview

Given the complex requirements described earlier and the quest for practical solutions, this thesis has chosen to construct its solution based on certain commonly employed approaches for addressing the aforementioned challenges. The selection of these approaches was made based on their documented effectiveness in solving these problems and their versatility in handling correlations in the fields of computer vision and multi-label tasks.

In the following section, we will initially introduce the tools used to understand the proposed solution, followed by the reference architecture, and then all the modifications and innovations made in our solution, outlining the key features and innovative contributions.

3.3.3 Preliminary Instruments

In this section, we will examine key components that are crucial for understanding the neural network architecture used in this thesis work. This architecture combines elements of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), drawing inspiration from previous work in this field, notably from the paper [53], which presents an architecture that will be detailed later. This combination of CNN and RNN has been chosen to address specific challenges and tasks related to processing visual and sequential data.

We will delve into both the CNN component, responsible for extracting informative features from input images, and the RNN, which plays a fundamental role in generating relevant label sequences. Their main characteristics will be highlighted.

CNN

Convolutional Neural Networks (CNN), commonly known as ConvNets or CNNs, represent a milestone in the field of machine learning, especially in image and visual data processing. They were introduced in the 1990s, inspired by the neocognitron. Yann LeCun and others, in their article "Gradient-Based Learning Applied to Document Recognition" [54], demonstrated that a CNN model, which aggregates simpler features into progressively more complex ones, can be successfully

used for recognizing handwritten characters.

CNNs are a regularized form of feed-forward neural networks that learn feature engineering through autonomous filter (or kernel) optimization. This architecture was designed to address the issues of vanishing or exploding gradients during backpropagation, which were common in previous neural networks.

Their regularized structure is based on the concept of convolution, which is one of the key features that make them so effective in processing images and data with spatial structure. It allows them to analyze small local regions of the image at a time, rather than considering the entire image in a single pass. This is crucial because many visual features, such as lines, curves, edges, and textures, are locally distributed within an image. This approach makes CNNs more memory and computationally efficient and less prone to overfitting. During the convolution process, the filter (or kernel) weights remain fixed and are applied to different positions of the image.

These networks also tend to be translation-invariant, meaning they can detect the same features regardless of their position in the image. For example, if a CNN learns a filter to detect a vertical edge in one part of the image, it will be able to detect the same type of edge in other parts of the image.

Finally, they are capable of hierarchical learning: CNNs use a series of convolutional layers, each with different filters. This allows them to learn increasingly complex features hierarchically. For example, the first layers can learn to detect edges and corners, while subsequent layers can combine these features to recognize more complex shapes like faces or objects.

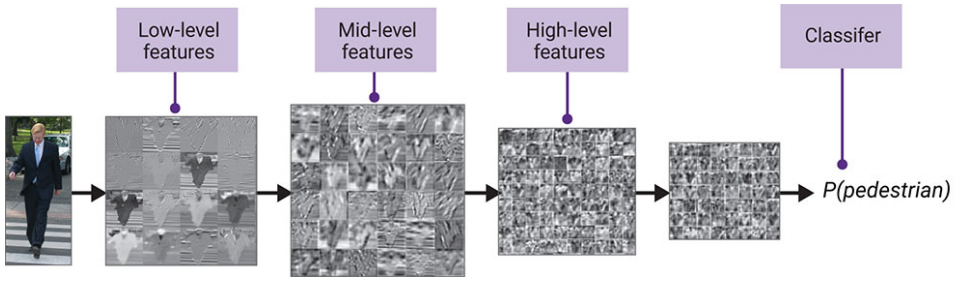

Figure 3.9: Example of Feature Extraction Process Conducted by a CNN

The applications of CNNs today are truly endless, ranging from image and video recognition to recommendation systems, from segmentation to medical image

analysis, from natural language processing to financial time series analysis, and so on.

A typical architecture of CNNs includes an input layer, hidden layers, and an output layer. The hidden layers typically consist of one or more layers that perform convolutions. When the input passes through these layers, the image is abstracted into a series of feature maps with different shapes. Additionally, other types of layers, such as pooling layers (max and average pooling being the most famous), can be used, which can be either local or global. These layers reduce the data dimensions by combining clusters of neurons into a single neuron in the next layer, significantly reducing the risk of overfitting. Fully connected layers and normalization layers are also frequently employed.

The goal of CNNs is to automatically learn filter optimization during training, in contrast to traditional algorithms that require manual feature engineering.

RNN

Recurrent Neural Networks (RNN), abbreviate come RNN, are artificial neural networks that distinguish themselves by the direction of information flow between their layers. Unlike unidirectional feed-forward neural networks, RNNs are bidirectional artificial neural networks, which means they allow the output of certain nodes to influence the subsequent input in the same nodes. The key feature of RNNs is their ability to use an internal state (memory) to process arbitrary sequences of input, making them suitable for tasks such as recognizing unsegmented connected handwriting or speech recognition.

RNNs have a rich history of development. The Ising model (1925) by Wilhelm Lenz and Ernst Ising was one of the early architectures of RNNs, although it did not have learning capabilities. Shun'ichi Amari improved its adaptation in 1972. This architecture is also known as the Hopfield network (1982). In 1993, a neural history compression system successfully solved a "Very Deep Learning" task that required over 1000 successive layers in a time-unrolled RNN.

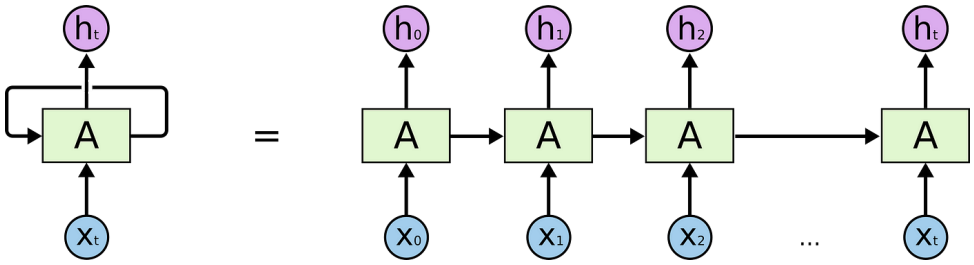

Figure 3.10: Idea behind the architectures of recurrent neural networks (RNN)

Among the most famous types today are:

- Long Short-Term Memory (LSTM): LSTM networks are a type of Recurrent Neural Network (RNN) that address the "vanishing gradient" problem in traditional neural networks. The vanishing gradient is a problem in which gradients during training become too small, making it challenging for the network to learn long-term relationships in sequential data.

  LSTMs overcome this issue by introducing three fundamental gates:

  Input Gate: This gate decides which new information should be added to the cell state and also uses a sigmoid function to determine which input values are relevant.

  Forget Gate: This gate determines which information from the previous cell state should be forgotten or retained. Here, a sigmoid function decides which values should be forgotten.

  Output Gate: This gate decides which information in the cell state should be used as the output of the current step. The output is filtered through a sigmoid function and passed through a hyperbolic tangent (tanh) function to limit values to the range between -1 and 1.

  These gates enable LSTMs to learn how to handle incoming information, store it long-term, and select which information to send as output. These networks are particularly suitable for processing complex sequential data, such as natural language understanding, speech recognition, and automatic translation.

- Gated Recurrent Unit (GRU): Gated Recurrent Units (GRUs) are a variant of RNNs that, like LSTMs, address the vanishing gradient problem. GRUs

simplify the architecture compared to LSTMs by reducing the number of gates from three to two:

Update Gate: This gate determines how much of the information from the past should be retained for the current time step. It functions similarly to the input gate in LSTMs, deciding which new information to add to the current state.

Reset Gate: This gate decides which information from the previous state should be forgotten or reset for the current time step.

GRUs are less complex than LSTMs and require fewer parameters, making them easier to train and less susceptible to overfitting. However, LSTMs tend to have a higher learning capacity in scenarios where it's necessary to store information for the long term.

3.3.4 Reference algorithms

The reference architecture under consideration has been proposed in this work [53]. It consists of two main parts: a CNN (encoder) that extracts a compact visual representation from the image and an RNN (decoder) that uses the encoding to generate a sequence of labels, modeling the dependencies between the labels themselves.

A diagram of the model's architecture is provided below:

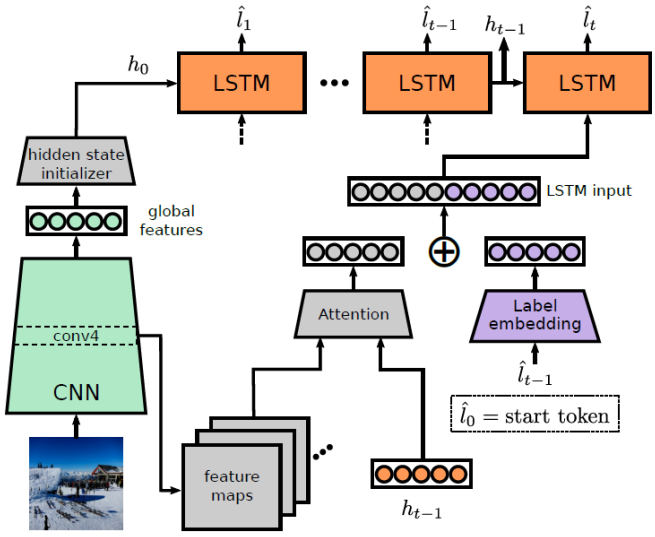

Figure 3.11: The CNN-RNN architecture used in the reference paper [53] includes a CNN encoder for images, an LSTM decoder for text, and an attention mechanism.

As a CNN, the pretrained ResNet-101 [55] on ImageNet [56] is used. As an RNN, a Long Short-Term Memory (LSTM) is employed. To initialize the hidden state, the final fully connected layer of the CNN is utilized. Subsequently, the RNN model predicts a new label at each time step until an end signal is generated.

An attention module is also included, which takes linearized activations from the fourth convolutional layer, along with the hidden state of the LSTM at each time step as input. The attention module focuses on different parts of the image, and the weighted features extracted from the module are concatenated with the predicted class word embedding from the previous time step. This is then provided as input to the LSTM for the current time step.

In their case, this approach effectively addresses the problem of representing small objects with global features because the attention module allows focusing on specific objects in the image during the prediction step.

Before discussing the training procedures, it's important to highlight an innovative approach introduced in this work. They introduce two new methods called "Predicted Label Alignment" (PLA) and "Minimum Loss Alignment" (MLA) to handle label assignment without imposing a predefined order. These methods dynamically align the order of actual labels based on loss minimization during training. This leads

to faster training and eliminates issues such as repeated assignment of labels in the predicted sequence. In particular, the PLA algorithm used in the final model aims to order labels initially, starting with those where the model is most confident, and subsequently, those where it is uncertain. This way, the network learns to predict labels it is more confident about first and leverages this information (encoded in the state) to improve the prediction of subsequent labels.

The steps taken for training this architecture are provided below:

- A preliminary training was conducted for the CNN with a simple classifier as the head. The optimizer used was Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and a momentum of 0.9. The model was trained for 40 epochs, and early stopping was applied in case no improvement was observed after 3 epochs.

- The final model was subsequently trained, incorporating the LSTM and attention modules, using the encoder obtained in the previous step. The ADAM optimizer was employed, along with a cyclic learning rate scheduler that decreased from 10^{-3} a 10^{-6} over three iterations. The learning rate was reduced by 10% compared to the CNN, and a fine-tuning phase was performed on the feature extraction part, further training for 30 epochs.

In all the previous steps for data augmentation, random affine transformations and contrast changes were applied.

In the inference process, the beam search algorithm was not used; instead, a greedy selection strategy was adopted, simply choosing the predicted output with the highest probability.

3.3.5 Methodological Breakthroughs

In the following section, we will explore the fundamental methodological advancements that have been achieved in our research. We will present both the technological innovations incorporated into the final architecture of our model and the substantial changes made to the initial architecture from which we drew inspiration.

GradNorm

In the field of multitask learning in computer vision, the use of individual models for specific tasks has achieved significant success, often leading to human-comparable or superior performance in a wide range of tasks. However, to attain a complete visual system for understanding complex scenes, the ability to efficiently and simultaneously perform multiple different perceptual tasks is essential, especially in resource-constrained environments like embedded systems such as smartphones, wearables, and robots/drones. Multitask learning, where a model shares weights across multiple tasks and makes multiple inferences in a single forward pass, can make such a system feasible. Not only are such networks scalable, but the shared features within them can introduce more robust regularization and enhance performance. Multitask networks, in general, are challenging to train; the various tasks must be adequately balanced for network parameters to converge to robust shared features useful for all of them. Methods in multitask learning have so far primarily attempted to find this balance by manipulating the network's forward pass (e.g., by constructing explicit statistical relationships between features or optimizing multitask network architectures). However, such methods overlook a key aspect: task imbalances hinder proper training because they manifest as imbalances in the gradients propagated backward. A task that is too dominant during training, for example, will necessarily express this dominance by inducing gradients with relatively high magnitudes. The goal of the GradNorm algorithm proposed by [57] is to mitigate these issues at their core by directly modifying the gradient magnitudes through adjusting the multitask loss function. In practice, the multitask loss function, given T tasks, is often considered linear in the losses of individual tasks L_i, i.e., $L = \sum_{i \in T} w_i L_i$. The proposed method is adaptive, so w_i can vary at each training step t: $w_i = w_i(t)$. This linear form of the loss function is convenient for implementing gradient balancing, as w_i is directly and linearly coupled to the magnitudes of the gradients propagated backward from each task. The challenge then lies in finding the best value for each w_i at each training step t to balance the contribution of each task for optimal model training.

GradNorm [57] focuses on two main objectives:

- Establishing a common scale for the magnitudes of gradients from different tasks, allowing us to reason about their relative sizes.

- Dynamically balance the magnitudes of gradients so that different tasks train at similar rates.

To achieve these objectives, GradNorm introduces a method for dynamically updating loss weights in multitask learning to adapt to the training needs of the tasks themselves. This method is based on the idea that the magnitude of backward-propagated gradients is indicative of the training effectiveness of a task. Therefore, tasks that are training more rapidly require gradients with larger magnitudes to encourage faster learning, while slower tasks require gradients with smaller magnitudes. GradNorm introduces an additional parameter, α, to control the strength with which task training rates are balanced. This parameter allows the algorithm to adapt to different task learning dynamics, such as when some tasks are significantly more complex than others.

The key innovation of GradNorm lies in its direct approach to gradient normalization, effectively mitigating task imbalances from the early stages of training. This results in increased network efficiency and performance, significantly reducing overfitting.

Below is the algorithm proposed in the paper [57]:

Changes respect reference Architecture

In the architecture used, several key modifications were introduced compared to the reference architecture to adapt it to the specific problem and the available resources, including the dataset used (RSCD). Notably, some of these changes were made primarily to tailor the network to the specific problem, while others were aimed at maintaining continuity with the research conducted in this work. The main changes made are as follows:

1. Data Input and Preprocessing: The reference paper [53] used the COCO dataset [58] for training the proposed architecture, which aimed at object recognition in images. The dataloader was configured with a maximum sequence of predictable labels, around 18, in addition to the $< start >$ and $< stop >$ tokens. In your case, the problem is different, although it involves multiple labels; it focuses on solving three sub-tasks, each of which is a simple classification problem with the prediction of a single label. Therefore, the approach taken was to define a set of labels formed by the union of all possible predictions for the three tasks, with a prediction limit of

Algorithm 1 Training with GradNorm

Initialize $w_i(0) = 1 \; \forall i$

Initialize network weights W

Pick a value for $\alpha > 0$ and pick the weights W (usually the final layer of weights shared between tasks)

for $t = 0$ to max_train_steps **do**

Input batch x_i to compute $L_i(t) \; \forall i$

$L(t) = \sum_i w_i(t) L_i(t)$ [standard forward pass]

Compute $G_W^{(i)}(t)$ and $r_i(t) \; \forall i$

Compute $\overline{G}_W(t)$ by averaging the $G_W^{(i)}(t)$

Compute $L_{\text{grad}} = \sum_i |G_W^{(i)}(t) - \overline{G}_W(t) \times [r_i(t)]^\alpha|_1$

Compute GradNorm gradients $\nabla w_i L_{\text{grad}}$, keeping targets $\overline{G}_W(t) \cdot [r_i(t)]^\alpha$ constant

Compute standard gradients $\nabla_W L(t)$

Update $w_i(t) \rightarrow w_i(t+1)$ using $\nabla w_i L_{\text{grad}}$

Update $W(t) \rightarrow W(t+1)$ using $\nabla_W L(t)$ [standard backward pass]

Renormalize $w_i(t+1)$ so that $\sum_i w_i(t+1) = T$

end for

3 for each image, excluding the start and stop words. This approach was designed to force the network to provide a maximum of 3 predictions, one for each task. For preprocessing, it was set equivalently to the one conducted in the paper associated with the dataset [13], with normalization like: 'transforms.Normalize([0.498, 0.498, 0.498], [0.500, 0.500, 0.500])' applied to each sample.

2. Modification of Network Outputs: Changing the problem at hand also impacted the network's output, particularly in the training phase. Several implementation changes were made to adapt the algorithm's implementation during training to match the description in the paper [53]. Additionally, following the guidelines of the paper [13], Cross-Entropy loss was used, Adam as the optimizer, and a batch size of 128.

3. Changes to the CNN Part: To make the most of the correlation between different tasks, several choices were made regarding the CNN used. In the reference paper, the pretrained ResNet-101 [55] on ImageNet [56] was employed. It underwent an initial training phase as a simple classifier with a basic head for the problem they were solving. Subsequently, this was set as the CNN for the entire architecture, with the classification head removed. This allowed for training only the RNN while keeping the CNN weights fixed or with fine-tuning of the CNN part. In line with the paper [53], an EfficientNet B0 [59] with pretraining on ImageNet was chosen, undergoing initial training with a classifier head. However, a multi-head architecture with three heads was adopted, with hard-sharing. The GradNorm algorithm [57], extensively discussed earlier, was introduced for gradient normalization.

Please note that the above description provides an overview of the modifications made to adapt the architecture for your specific task and dataset.

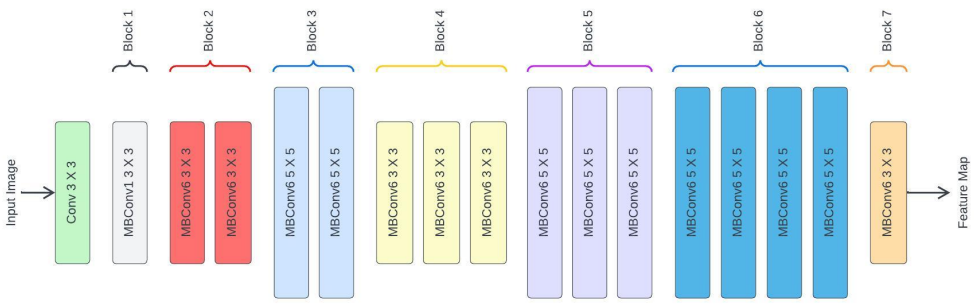

Figure 3.12: Architecture of EfficientNet-B0 [59].

The idea behind this choice was to leverage this combination to compel the network to simultaneously extract discriminative features for all three tasks, making it easier for the subsequent RNN to extract all possible information for relating the various tasks. Finally, the obtained CNN was set as the complete architecture's CNN.

3.4 Tools, technologies and models used for the realization

The following section is used to argument tools and technologies used to implement all the content required for this thesis.

3.4.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures,combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development and for use as a scripting or glue language to connect existing components. Python's simple, easy-to-learn syntax emphasizes readability and reduces the cost of program maintenance. [60]

Nowadays, Python is one of the most used programming languages, especially for the data-analysis, Machine Learning and AI applications. The main reason to support the usage of Python in programming for machine learning applications rely in the simplicity and intuitive application to coding when it comes o defining ML architectures, focusing on machine learning and not coding problems. The huge amount of innovative and up-to-date libraries and frameworks also makes it easier to implement brand new architectures thanks to the advanced support provided by the available codes and community support. Just a few examples are frameworks like Keras and PyTorch for neural network development, while pretrainedmodel or numpy as support libraries.

Moreover Python is platform-independent. That allows the user to remove the complexity of moving the application from one computer to another.

3.4.2 PyTorch

PyTorch [61] is an ML framework based on the Torch library. It was initially developed by Meta AI but is now part of the Linux Foundation umbrella. PyTorch is free and open-source software. Many famous AI applications are developed under PyTorch, i.e., Tesla Autopilot and Uber's Pyro. Some key features can be find in

PyTorch:

- ***Dynamic Computational Graph***: PyTorch utilizes a dynamic computational graph, allowing for flexible and dynamic graph construction during runtime.
- ***Tensors Usage***: PyTorch provides a multi-dimensional array called a tensor, which is similar to NumPy arrays but optimized for GPU acceleration. Tensors are the fundamental building blocks for creating and manipulating data in PyTorch.
- ***Automatic Differentiation***: One of PyTorch's standout features is its automatic differentiation library, called Autograd, tool that automatically computes gradients of tensors with respect to operations performed on them, making gradient-based optimization effortless.
- ***Easily buildable and dynamic Neural Networks***: Torch.nn module offers tools for building neural network architectures. Custom layers, loss functions and optimizers can be easily defined while enjoying the flexibility of building complex models. Moreover its dynamic nature enables creating neural networks that can change behavior during runtime. This is beneficial for research, experimentation, and working with sequences of varying lengths.
- ***GPU Acceleration***: PyTorch supports GPU acceleration, allowing for efficient training and inference on graphics hardware. This speeds up computations significantly, particularly for deep learning models.

Chapter 4

Experimental validation and application aspects

In this chapter, the underlying reasons for choosing the specific proposed model are explained. Subsequently, the metrics used to evaluate the model and the results achieved within the scope of these metrics are presented.

4.1 Description of the evaluation metrics

In this section, we delve into the choice of metrics employed to evaluate the performance of the algorithms selected during the experimentation phase. As mentioned in the previous chapters, this thesis aims to address three tasks concurrently:

- Vehicle-Road Friction Estimation: Predicting the coefficient of friction of the road surface, categorized into six different conditions: dry, wet, water-covered, fresh snow, melting snow, and ice.
- Road Material Identification: Recognizing the composition of the road surface material, distinguishing between common types such as asphalt, concrete, mud, and gravel.
- Irregularity Detection: Identifying surface irregularities caused by material degradation, including smooth surfaces with light or severe cracks.

Given the complexity of the problem, it is necessary to use metrics that can effectively capture the performance of the algorithms both individually, with respect

to all these multiple sub-tasks, and collectively. This is further important because the reference paper [13] uses a single-task network, where each class is a combination of the three classes belonging to the three initial tasks, making it possible to compare various results. The metrics that have been specifically employed are:

- Single-Task Accuracy: Accuracy is one of the most common evaluation metrics in machine learning and is widely used to measure the performance of classification models. Essentially, accuracy measures the percentage of correct predictions compared to the total predictions made by the model. The formula to calculate accuracy is as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{4.1}$$

  Accuracy has several positive aspects: ease of understanding, as it is an intuitive and easily comprehensible metric; a direct interpretation, as a high accuracy value indicates that the model is making correct predictions in most cases, which is often the primary goal of machine learning. Another positive aspect is that single-task accuracy allows us to evaluate the model's performance immediately in line with the assessments made within the previously discussed paper.

  However, accuracy has some limitations and negative aspects to consider: it is not suitable for imbalanced datasets. In the presence of datasets where some classes are much more numerous than others, accuracy can be misleading. The model might achieve high values simply by always predicting the majority class. This issue is particularly relevant in this work due to the dataset's imbalance. Accuracy also does not consider the importance of incorrect predictions. In certain contexts, errors in certain classes can be more costly or dangerous than errors in others, where they are less significant. Accuracy treats all errors the same and does not consider the relative importance of the classes.

  In light of these considerations, it should be noted that while the results obtained may not be representative of all classes in a highly imbalanced problem, this metric was used for consistency with the reference paper.

- In the context of a multitask problem, each task aims to address a specific challenge. Evaluating accuracy for each of these sub-tasks offers several advantages:

1. Understanding Specific Performances: By calculating accuracy for each sub-task, you gain insight into the model's performance on each individual aspect of the problem. This allows you to precisely identify which tasks the model is handling well and which might require improvements.

2. Customizing Adaptation: In some real-world applications, one or two sub-tasks may be more critical than others. Evaluating separate accuracies allows for the customization of model adaptation by focusing more on the more crucial areas.

Each sub-task in the road surface classification problem presents different challenges. Calculating accuracy separately for each sub-task provides information about the model's competence in addressing the specific nuances of each category. This metric is useful for understanding the model's strengths and weaknesses in different facets of the problem.

- This type of accuracy aims to provide additional information compared to other metrics used to evaluate the model's ability to make only one of the three predictions incorrectly, resulting in less severe errors compared to making errors in multiple tasks simultaneously. This information is useful when combined with the insights obtained from single-task accuracy.

- This metric addresses the issue of accuracy in the presence of imbalanced datasets. Its formula is as follows:

$$\text{Balanced Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{4.2}$$

Dove:

N the number of classes.

TP_i represents the true positives for the class i.

FN_i represents the false negatives for the class i.

The main advantage of this metric is its ability to effectively evaluate the quality of models in the presence of imbalanced datasets, where some classes

are much more numerous than others. In fact, it allows minority classes to have equal importance compared to the majority.

It should be noted that, in situations where some classes have a very small number of samples, the fact that these contribute equally to those provided by the more numerous classes could be a negative aspect. This is because if these few samples are either not truly representative of the class they belong to or, in general, are not reliable due to their limited quantity.

In our case, given the potential class imbalance in the single-task problem, balanced accuracy provides a more reliable assessment as it considers the performance of all classes. It takes into account sensitivity to each of them, ensuring that the model's success is not influenced by the predominance of one category over the others.

Through the use of this set of metrics, we have aimed to provide a comprehensive evaluation of the algorithm's performance in the context of road surface classification. These metrics allow us to understand the model's capabilities, its strengths in addressing individual sub-tasks, and its overall adaptability to the complex and dynamic nature of a wide range of conditions that may occur. Additionally, they facilitate a direct comparison with the results obtained in the reference paper and offer insights into the model's potential for real-world applications.

4.2 Description of parameters

In this section, we provide a detailed description of the parameters used in the training of our models, following the choices outlined in the reference paper. These parameters are important to provide a comprehensive view of what we have done. The values of these parameters are summarized in Table 4.1 for easy reference.

Table 4.1: Parameters Used during Experiments

Parameter	**Value**
Backbone	EfficieneNet-B0
Pre-Trained	ImageNet dataset
Loss	Cross-Entropy Loss
LR	0.001
Optimizer	Adam
LR Scheduler	Exponential decay of 0.8

These parameters ensure a comparison between the various models chosen and those published in the referenced work, as well as any justifications regarding their effectiveness and performance.

4.3 Experimentation

In the context of experimentation, our objectives are both to follow the path outlined by the analyses conducted in Chapter 3 in order to confirm the formulated assumptions and to evaluate the performance of the final architecture defined. Below, we outline the steps that we will examine in detail in the following sections:

- Verification of Issues in the Initial Organization of the RSCD Dataset: We will begin with the experimental confirmation of the issues that emerged during the analysis phase regarding the initial division made for this dataset into the train, validation, and test sets. This step will be crucial to further justify the need for reorganizing the various subsets.

- Comparison of the single-task model proposed by the dataset with a Multihead architecture trained on the reorganized dataset. This will allow us to assess the impact of addressing the divided problem as separate tasks rather than treating them together.

- Introduction of the 'GradNorm' Algorithm: We will experiment with introducing the 'GradNorm' algorithm into the previously mentioned multihead architecture. These results will help us understand how this influences the overall performance of the model.

- Class Balancing in the Batch: Given the unequal class distribution, we will assess the effectiveness of incorporating a class balancing mechanism during batch formation. This will help us determine if this solution can mitigate the issue of class imbalance.

- Final Model Evaluation: We will consider our final model, extensively discussed in the personal contributions chapter. We will carefully examine it to assess its effectiveness and the contribution it makes in the context of our research.

Through this process, we hope to gain a clear overview of our model's performance and the validity of the choices we have made throughout the experimental journey.

4.3.1 Validation of Issues in the initial organization of the RSCD Dataset

To conduct this experimentation, we decided to replicate the architecture proposed in the reference work [13], the complete code of which unfortunately has not been made public. We aimed to perform two training runs:

- One using the dataset organized as provided in the paper, essentially replicating the experiment they conducted with the goal of obtaining similar results.
- A second training run of the same architecture with the same parameters but on the reorganized dataset. In this reorganized dataset, the data was divided by acquisition days, maintaining the percentages they specified for validation. The aim was to verify a potential decrease in the network's generalization capability on the validation and test sets.

During this phase, to compare the experiments, we used single-task accuracy as the primary metric because the model we used followed this logic, and the results presented in the paper were calculated in this way. However, values for other metrics were also calculated, leveraging the correspondence between the single-task problem class and the triplet of corresponding classes in the Multi-task problem.

In the following tables, we present the results obtained, defining:

- 'Official Paper Dataset': the results published in the paper using the dataset they employed.
- 'Our Paper Dataset': our results under the same conditions as the 'Official Paper Dataset.'
- 'Our Reorganized Dataset': the results obtained by us using our reorganized version of the dataset.

Metric	Our Paper Dataset	Our Reorganized Dataset
Acc.	89.47	67.75
2/3 Acc	96.85	88.69
1 task Acc.	89.86	95.96
2 task Acc.	96.94	76.15
3 task Acc.	97.01	87.19

Table 4.2: Results on the Validation Set for Issues in the initial organization of the RSCD Dataset

Metric	Official Paper Dataset	Our Paper Dataset	Our Reorganized Dataset
Acc.	89.19	90.94	70.46
2/3 Acc	-	97.22	90.19
1 task Acc.	-	91.39	85.82
2 task Acc.	-	97.38	82.46
3 task Acc.	-	97.28	89.06

Table 4.3: Results on the Test Set for Issues in the initial organization of the RSCD Dataset

As highlighted in Table 1, for 'Our Paper Dataset,' the model achieves an overall accuracy of 89.47%, whereas the same architecture on 'Our Reorganized Dataset' exhibits a significantly lower accuracy of 67.75%. This substantial difference underscores how the dataset reorganization has a significant impact on model performance, which deteriorates by over 20%.

The '2/3 Accuracy' and metrics related to specific tasks, such as 'First task Accuracy,' 'Second task Accuracy,' and 'Third task Accuracy,' further emphasize this deterioration.

In Table 2, which presents the results obtained on the 'Test Set,' there is further confirmation of these disparities. We observe an accuracy of 70.46% in 'Our Reorganized Dataset' compared to 90.94% in 'Our Paper Dataset.' The differences in task-specific metrics also suggest that the change in data organization significantly affects the test set. It is worth noting that the performance published in the paper,

89.19% in the 'Official Paper Dataset,' is comparable to what we achieved in our case, 90.94%, rendering the correspondence of architectures between both works quite reliable.

In summary, these tables clearly highlight how model performance is highly sensitive to the initial dataset structure. Data reorganization has led to a significant drop in performance, suggesting that with proper refinement of the dataset structure, the model could achieve results much closer to what one would expect in a realistic context.

As a result, we plan to employ the dataset reorganization in the upcoming stages of the experiment to realistically assess the performance of the models to be tested. However, this time, we intend to use percentage splits of the data that ensure the presence of all classes in each of the three subsets, as previously mentioned in the analysis phase.

4.3.2 Investigating the Single-Task Model from the Dataset Against Multi-Head Architectures

Once the final dataset organization was established, we proceeded to adjust the model to align it more closely with the problem at hand. Given that we were dealing with a problem composed of three sub-tasks, we defined a model that included three heads, with each one focusing on a specific sub-task. We retained the same backbone structure as in the previous experimentation for continuity.

This time, the metrics that gained significant importance were the accuracies on individual tasks, as they were now decoupled from the problem seen in terms of a single task. Below are the tables illustrating this comparison:

Metric	Single Head Model	Three Heads Model
Acc.	70.39	71,35
2/3 Acc	90.36	88,76
1 task Acc.	85,63	86,83
2 task Acc.	86,55	87,22
3 task Acc.	86.57	85,27

Table 4.4: Results on the Validation Set for Single-Task Model Against Multi-Head Architectures

Metric	Single Head Model	Three Heads Model
Acc.	70,33	67,17
2/3 Acc	89,48	88,06
1 task Acc.	90,62	85,57
2 task Acc.	83,9	91,03
3 task Acc.	83,52	82,09

Table 4.5: Results on the Test Set for Single-Task Model Against Multi-Head Architectures

Analyzing the results of the validation set, we note that the Multi-task model has a slightly higher overall accuracy compared to the Single Head model, with a score of 71.35% versus 70.39%. This suggests that in terms of overall performance, the Three Heads model has a slight advantage.

However, it is essential to take a closer look at the different metrics, particularly the individual accuracies for each task. Here, we see that the Three Heads model has a significant advantage. In the validation set, the Three Heads model outperforms the Single Head model in two out of three individual task accuracies (1 task Acc., 2 task Acc.). This suggests that the Three Heads model can better handle each sub-task independently, improving overall performance.

However, the test set results tell a slightly different story. The Single Head model outperforms the Three Heads model with an overall accuracy of 70.33%

compared to 67.17%. This may indicate that the Single Head model generalizes better to new data compared to the Three Heads model.

In summary, the results in the tables suggest that the Three Heads model is capable of achieving superior performance on the validation set, particularly on individual tasks. However, the Single Head model seems to generalize better to unseen data, as demonstrated by the test set results. The choice between the two models will depend on the importance placed on individual task performance and the ability to generalize to new data.

Therefore, based on this experimentation, the Single Head model is likely slightly superior to the Multi-Head model. However, it's worth considering that the model designed to handle each of the three tasks individually has significantly more potential, especially when thinking about the promising applications that can be integrated, some of which will be discussed later.

4.3.3 'GradNorm' Algorithm Integration

In an effort to maximize the potential of the previously discussed multitask architecture that simultaneously tackles three distinct tasks, it was decided to integrate the GradNorm algorithm into the model, as seen in the preceding chapter and proposed by the following paper [57], to achieve an optimal balance among these tasks and enable a smooth progression of the model in addressing each of them.

The primary objective of this section is to explore how the integration of "GradNorm" can influence the performance of the multitask architecture. This algorithm, known for its role in regulating gradients in multitask systems, has the potential to mitigate task interference and facilitate an even distribution of training resources among activities. In this way, we aim to enhance the model's ability to simultaneously handle the three tasks of interest.

The following tables present a comparative analysis of the multitask model's performance with and without the use of "GradNorm" to assess how this algorithm concretely impacts our model's ability to manage the complexities of the three tasks. The outcomes of this comparison will be crucial in understanding its impact and determining whether "GradNorm" can make a significant contribution to the model's performance in this multitask context:

Metric	without GradNorm	with GradNorm
Acc.	71,35	73.79
2/3 Acc	88,76	90.08
1 task Acc.	86,83	88.44
2 task Acc.	87,22	88.31
3 task Acc.	85,27	86.26

Table 4.6: Results on the Validation Set for 'GradNorm' Algorithm Integration

Metric	without GradNorm	with GradNorm
Acc.	67,17	71.24
2/3 Acc	88,06	89.69
1 task Acc.	85,57	86.93
2 task Acc.	91,03	92.15
3 task Acc.	82,09	84.63

Table 4.7: Results on the Test Set for 'GradNorm' Algorithm Integration

In Table 1, we conducted a comparative evaluation of the performance of our multitask model on the validation set in two scenarios, with and without the integration of "GradNorm." With "GradNorm," the model demonstrates an overall accuracy increase compared to the implementation without it. Specifically, the model's accuracy without "GradNorm" is 71.35%, while with "GradNorm," it reaches a higher value of 73.79%.

Looking at the specific metrics for the various tasks, we also notice significant improvements with "GradNorm." For instance, the accuracy of the first task, without "GradNorm," is 86.83%, but with this integration, it reaches a higher value of 88.44%. The same principle applies to the second task, where accuracy increases from 87.22% to 88.31%. For the third task, the values were 85.27% without GradNorm and 86.26% with it. This suggests that "GradNorm" contributes to higher precision in entity classification for each specific task.

In Table 2, the idea from Table 1 is confirmed, with overall accuracy increasing

from 67.17% to 71.24%. The specific metrics for each task show results similar to those observed on the validation set. 1 Task goes from 85.57% to 86.93%, 2 Task from 91.03% to 92.15%, and 3 Task from 82.09% to 84.63%. Based on these values, it is presumed that "GradNorm" has contributed to a better capability of the model to handle specific tasks, promoting more balanced learning and greater generalization.

These results suggest that the integration of "GradNorm" has a positive effect on the generalization capabilities of our multitask model, facilitating better model convergence and a more even distribution of training resources among the various tasks, resulting in an overall improvement in performance on both sets.

We will now proceed to evaluate the feasibility of implementing batch balancing to mitigate the strong class imbalance present.

4.3.4 Class Balancing

In the ongoing pursuit of more robust and effective machine learning models, a crucial aspect that arises is the management of imbalanced classes in datasets. This imbalance can pose a significant challenge, as the model tends to favor the more numerous classes at the expense of the less frequent ones. The dataset used in this thesis work, as also revealed in the analysis phase, is deeply imbalanced, requiring intervention in this regard.

In this section, we will explore the impact of class balancing applied at the batch level using the multitask architecture based on the 'GradNorm' algorithm defined in the previous section. The goal is to understand if the introduction of this technique can lead to an improvement in model performance, despite the knowledge that it may lead to a degradation of performance in terms of overall accuracy. In this case, our focus is on the balanced accuracy metric, as it represents a more sensitive indicator of the less numerous classes.

To balance the batch, the following steps were carried out:

- Calculated the frequencies of single-task classes within the training set.
- Obtained class weights based on the calculated frequencies. These weights, in particular, are the inverses of the frequencies, meaning less frequent classes have higher weights, while more frequent classes have lower weights.

- The weights were subjected to a function called 'flattening,' which performs a flattening of the weights based on a parameter ranging from 0 to 1, making this operation stronger or weaker. Specifically, this function calculates the mean of the class weights, obtains the differences between each weight and the mean value of all weights, multiplies these differences by the previously defined flattening coefficient, and finally adds these values to the mean, defining the new weights associated with the classes. This means that if the coefficient is 0, all weights would be equated to the mean, resulting in equal probability for all samples to be drawn. Conversely, with a value close to 1, the weights would remain unchanged, producing complete batch balancing.

- The class weights are used to create a weighted sample. Samples have a higher probability of being drawn in batch construction if they belong to the less frequent classes.

- The weighted samples are then drawn during training, no longer following a uniform probability but the newly defined one from the previous steps, creating the various batches. This ensures that, during training, the model receives a proper class balance, helping improve performance on less frequent classes.

The experiments conducted are summarized in three settings. In the first one, the results of the model without any class balancing are presented. Subsequently, the results of the same model with partial class balancing using a function to make the balancing weights associated with classes smoother are shown. Finally, the results of complete class balancing are displayed. The key metrics used for these experiments include balanced metrics. Therefore, balanced accuracy for the single-task problem, along with mean and variance values, was calculated, as well as for each individual task in the multitask problem. Summary tables are provided below:

Metric	without Balancing	with partial Balancing	complete Balancing
Acc.	**73.79**	72.34	63.09
2/3 Acc	**90.08**	89.55	81.61
1 task Acc.	**88.44**	85.02	75.22
2 task Acc.	88.31	87.98	**89.29**
3 task Acc.	**86.26**	85.81	81.04
	mean, std	mean, std	mean, std
Single task Balanced Acc.	**0.39, 0.29**	0.31, 0.3	0.19, 0.3
1 task Balanced Acc.	**0.58, 0.39**	0.54, 0.36	0.38, 0.41
2 task Balanced Acc.	**0.88, 0.04**	0.87, 0.05	0.79, 0.09
3 task Balanced Acc.	**0.78, 0.23**	0.77, 0.24	0.72, 0.26

Table 4.8: Results on the Validation Set for Class Balancing

Metric	without Balancing	with partial Balancing	complete Balancing
Acc.	**71.24**	70. 32	63.09
2/3 Acc.	**89.69**	88. 96	81.6
1 task Acc.	**86.93**	85. 08	75.22
2 task Acc.	**92.15**	91. 85	89.28
3 task Acc.	84.63	**84.97**	81. 03
	mean, std	mean, std	mean, std
Single task Balanced Acc.	**0.34 0.31,**	0.29, 0.32	0.2, 0.32
1 task Balanced Acc.	**0.51 0.37,**	0.47, 0.37	0.32, 0.39
2 task Balanced Acc.	**0.87 0.12,**	0.85, 0.12	0.76, 0.24
3 task Balanced Acc.	**0.78 0.24,**	0.77, 0.25	0.75, 0.24

Table 4.9: Results on the Test Set for Class Balancing

In the section dedicated to Class Balancing, the results obtained from the comparative tables present some interesting considerations. In particular, a paradoxical phenomenon related to batch balance and performance metrics emerges. What the results show is a decrease in balanced accuracy, proportionally to the extent of batch balancing, with the worst results obtained when implementing complete balancing, 'Single task Balanced Acc.: 0.2' This result may appear counterintuitive, but it can be explained by several factors.

One possible explanation could be that, in the original dataset, samples from underrepresented classes may not actually be informative about real-world conditions.

Partial or complete balancing might, therefore, introduce samples from these classes that, not being truly representative, only introduce noise into the training process. This leads to a decrease in balanced accuracy, as well as overall accuracy, as the model loses strength for the more represented classes, failing to generalize effectively and, at the same time, not improving performance on the less represented classes while analyzing non-informative samples.

This is supported by the plateau of performance evident in the validation and test data, which differs significantly from the training data, with performance differences even exceeding 20%.

In general, these results highlight the complexity of class balancing in training data and the need for a careful evaluation of how balancing affects model performance. Further refinement of balancing strategies and improving the quality of samples for underrepresented classes may be necessary to achieve better performance.

4.3.5 Final Model Evaluation

Throughout this research journey, a significant part has been dedicated to optimizing the performance of our multitask model, aiming to enhance its ability to learn multiple tasks simultaneously.

The final model proposed represents the culmination of this constructed path, as emerged from the preceding sections, following incremental steps. The aim has been to find an architecture that can more efficiently extract relationships between tasks, attempting to overcome the limitations of simple weight sharing or the introduction of the 'GradNorm' algorithm to ensure smooth progress across various tasks.

A comparison will be made here between the final model extensively described in the technological advancements chapter and the main models that have accompanied the experiments up to this point, evaluating them on all the metrics considered, examining both balanced and unbalanced overall performance, as well as the model's behavior on each specific task in detail. The objective is to understand the strengths and potential shortcomings of this architecture, conducting a comparison with both the multitask architecture with GradNorm and the baseline proposed by the reference work [13]. The results obtained are presented below:

Metric	Final Model	Multi-task base	Single-Task
Acc.	73.67	**73,79**	70.39
2/3 Acc	88.66	**90,08**	90.36
1 task Acc.	81.62	**88,44**	85.63
2 task Acc.	**88.57**	88,31	86.55
3 task Acc.	85.38	**86,26**	86.57
	mean, std	mean, std	mean, std
Single task Balanced Acc.	**0.63, 0.19**	0,39, 0,29	0.55, 0.22
1 task Balanced Acc.	**0.85, 0.11**	0,58, 0,39	0.8, 0.12
2 task Balanced Acc.	**0.88, 0.04**	0,88, 0.04	0.82, 0.06
3 task Balanced Acc.	**0.87, 0.07**	0,78, 0,23	0.85, 0.05

Table 4.10: Results on the Validation Set for Final Model Evaluation

Metric	Final Model	Multi-task base	Single-Task
Acc.	**71.59**	71.24	70.33
2/3 Acc	87.42	**89.69**	89.48
1 task Acc.	79.88	**86.93**	90.62
2 task Acc.	**92.41**	92.15	83.9
3 task Acc.	83.86	**84.63**	83.52
	mean, std	mean, std	mean, std
Single task Balanced Acc.	**0.49, 0.24**	0.34, 0.31	0.45, 0.26
1 task Balanced Acc.	**0.71, 0.18**	0.51, 0.37	0.71, 0.17
2 task Balanced Acc.	**0.84, 0.17**	0.37, 0.12	0.84, 0.12
3 task Balanced Acc.	0.8, 0.15	0.78, 0.24	**0.81, 0.13**

Table 4.11: Results on the Test Set for Final Model Evaluation

The first table presents the results of the three models - the "Final Model," the "Multi-task" model, and the "Single-Task" model - on the validation dataset. The results clearly highlight the superiority of the "Final Model" in terms of performance

on all the considered metrics. In particular, the "Final Model" outperforms the "Single-Task" model in terms of overall accuracy, 2/3 Accuracy, and accuracy in individual tasks while remaining in line with the "Multi-task" model. However, the "Multi-task" model proves to be extremely problematic in terms of balanced performance, significantly deteriorating unlike the other two. These results indicate the "Final Model's" ability to effectively understand and learn multiple tasks and improve overall performance.

In the second table, which relates to the results obtained on the test dataset, the "Final Model" continues to demonstrate superior performance compared to the other models, confirming its ability to generalize effectively.

Overall, the results highlight the remarkable success of the "Final Model" in multi-task learning, not only often outperforming the other two models but also maintaining stable performance across all analyzed metrics. This demonstrates a deeper understanding of the relationships between tasks compared to the comparative models. These results can be attributed to the advanced architecture of the "Final Model," which appears to extract the most relevant information from multi-task data effectively.

Additionally, the "Final Model" offers the advantage of reducing complexity, as a single model effectively replaces any individual models created for each task. This results in significant resource and time savings in practical implementation.

In conclusion, the "Final Model" emerges as a highly effective and promising solution for multi-task learning challenges, improving performance and demonstrating advanced understanding of task relationships.

4.4 Significants of the obtained results

In this section, we will delve into the meanings and implications of the results obtained during the experiments and make general considerations about the various models and the dataset used. Additionally, we will discuss potential future developments for this area of research.

4.4.1 Experimental Insights and Observations

Throughout the experiments, several aspects were explored, ranging from comparing single and multi-task approaches to utilizing GradNorm, from seeking dataset balance to employing more promising models. In this section, we aim to provide more general observations that emerge from the experience gained, not only enriching our understanding of the addressed problem but also proposing potential future developments. Therefore, we will discuss both the models considered and the dataset in two separate parts, evaluating their facets and the actual potential they provided for obtaining robust models during our experiments. All of this is intended to help us better grasp the implications of the relationships between data, models, and the results obtained.

Model Evaluation

- Single-Task Model: It was the first model employed in the experiments and also represents the approach used in the reference paper [13] to address the problem at hand. While it was surpassed by the multi-task models, it continues to be an important reference point, showing that in some cases, independent learning of each task is less effective in less balanced performance, but at the same time, it offers fewer opportunities to explore relationships between various tasks. Ultimately, it is somewhat more limited and provides less room for innovative solutions. It certainly comes with greater simplicity in analysis and implementation, but it is unquestionably the least promising model.

- Multi-Task Model: Multi-task models can have a very wide range of definitions, which can be both a strength and an additional challenge for those seeking to

employ them. Among all possible models, there are certainly highly effective solutions for the specific problem at hand. However, understanding the ideal model can be extremely challenging. Throughout the thesis, due to obvious constraints of time and resources, I focused on only a small subset of possible models that yielded excellent results. It's important to note that there are other equally viable models for which there is no data available for comparisons.

The first implemented model features complete weight sharing prior to the classification modules and three final heads. Its results do not significantly differ from the single-task case and may even exhibit more generalization issues. It performs well in terms of overall accuracy but faces challenges in terms of balanced accuracy. Overall, it represents an intermediate solution that is not fully mature and certainly improvable, given the potential that this type of network architecture offers.

The second model follows the same architecture as the first but incorporates GradNorm. By adding this modification during the training phase, the performance has improved significantly, highlighting the strength of such a tool. It still faces challenges regarding balanced accuracy but undoubtedly provides the possibility of more suitable solutions.

Overall, these models show great promise, especially when combined with effective techniques to enhance training for various tasks based on the specific problem at hand.

- Final Model: The proposed final model has shown a clear improvement in performance compared to the previous models. It excels in achieving valid performance both in terms of balanced metrics and other metrics. It emerges as the most promising architecture in this thesis, even though it may have slight shortcomings in certain tasks, such as the first task when compared to the multi-task architecture with the use of GradNorm. The fact that the CNN part of this model originated from the backbone of the multi-head model with GradNorm undoubtedly provided it with the potential for superior performance, as it had access to representative features of all three tasks. The attention layer, finally, is certainly important to allow the network to extract

discriminative features from the images.

Compared to the multi-task models seen previously, this model is more advanced, requiring fewer modifications and additions to further extend its architecture. The use of CNN-RNN architectures for multi-task problems is highly promising, offering various unexplored possibilities.

Dataset Evaluation The dataset used in this research was an essential component in developing the proposed models. However, it's important to emphasize that, as in the vast majority of real-world datasets, there are intrinsic challenges to address. Here are some observations that emerged from the experiments:

One interesting aspect arising from using this dataset for model training is the complexity of the problem at hand. When comparing the performance on the validation and test sets to that of the training set, it's clear that there is a significant plateau of approximately 20% in terms of performance. This characteristic was present in every model experimented with, whether simple or complex. It illustrates the considerable heterogeneity that the data collected for this problem might have, especially considering that this applies to data from the same city and not from a global perspective where such a system could potentially be used. However, this plateau also leads to other considerations, as these models generally reach an optimal level very quickly during training, typically within the first three epochs. These characteristics likely indicate the limited information content within the dataset, which is absorbed rapidly during training. As a result, these models tend to have a limited ability to generalize effectively, further explaining the performance achieved.

Another significant challenge is the pronounced class imbalance in the dataset. Since the problem was formulated in a general manner for all classes, it would be desirable for the models to achieve similar performance for each class. However, this uniformity is hard to achieve due to the strong imbalance in the data. This issue even leads to problems associated with dividing the data into training, validation, and test sets. In particular, the test set, which constituted 10% of the entire dataset, contained a very limited number of samples for some classes. This makes it difficult to obtain representative performance for those classes and inevitably affects the overall evaluation of the models.

Finally, a positive aspect that emerged is the ease of using the data for training, along with the excellent quality of the provided labels. This is especially noteworthy considering the difficulty in labeling this type of data, which often lacks clear differences between samples from different classes. This complexity can be observed in the various facets present within samples of the "wet," "ice," and "fresh snow" classes, making labeling an even more challenging task.

4.4.2 Future Prospects and Potential Advancements

In this section, we will explore potential directions in which this research area could evolve. Based on the observations and insights gained through the conducted experiments, we can identify several areas of interest for future developments. These developments could further enhance model performance and address challenges related to the dataset.

- Exploring Advanced Multi-Task Models: Throughout this research, we have only scratched the surface in terms of potential multi-task models. There are numerous approaches and architectures that could be explored to tackle the problem at hand. Analyzing new advanced multi-task models may lead to further improvements in performance and generalization capabilities.

- Investigating Data Augmentation Techniques: Given the heterogeneity of the data related to the problem at hand, the use of data augmentation techniques could be extremely beneficial. These techniques could be employed to generate synthetic data covering a broader range of possible scenarios, helping models become more robust. While data augmentation solutions were also evaluated during the experiments, as they did not yield significant improvements, it was decided to focus attention on other aspects due to time and resource constraints. However, data augmentation remains a potentially interesting avenue for future development.

- Developing Class Imbalance Handling Methods: Addressing the significant class imbalance is a critical challenge. Identifying and developing specific methods to manage this issue, such as adaptable class weights, intelligent oversampling techniques, or balancing the loss function instead of the batch,

could lead to a significant performance improvement, especially for the underrepresented classes.

- Explore Convolutional and Recurrent Neural Networks: Combined CNN and RNN architectures have shown great potential in multitask contexts. Continuing to experiment with these architectures to address the problem could lead to promising results, enabling the capture of spatial and temporal correlations in the data.

- Expand the Dataset: Acquiring a larger dataset is the most pressing development, and it would significantly contribute to the effectiveness of the models analyzed so far. A richer dataset could cover a wider range of situations and provide a greater variety of data, allowing models to generalize better and offering more reliable evaluations for model selection.

- Explore Continuous Training Approaches: Considering the heterogeneity of the data and the potential evolution of real-world conditions, models that can be continuously trained, updating with new data over time, could be developed.

- Applications in Specific Domains: This multitask classification problem can have applications in various sectors, such as traffic monitoring, robotics, and environmental condition analysis. Exploring the specifics and requirements of these contexts could lead to targeted developments and more effective applications.

In summary, the experiments conducted and the observations gathered in this work provide a solid foundation for future advancements in this field. The ongoing development of advanced models, dataset expansion, and the adoption of innovative techniques could significantly enhance the ability to address the complex problem of multitask classification in real-world conditions.

Chapter 5
Conclusion

In the automotive industry landscape, especially in the context of Advanced Driver Assistance Systems (ADAS), the natural integration of machine learning technologies could significantly enhance driving safety margins. In particular, in this thesis, the focus was on developing solutions to address the "Road Surface Classification" problem, including challenges such as friction estimation, road material recognition, and the identification of any irregularities caused by deterioration. Each of these can be seen as a different task with varying requirements and inherent difficulties.

The main purpose of this work is to adapt driving based on these conditions by employing efficient and robust machine learning methodologies, greatly reducing the risk of accidents. Multi-task networks played a central role in this, allowing us to tackle these multiple related tasks simultaneously. This enabled us to obtain solutions that are both more efficient compared to using individual models for each task and more robust by training architectures capable of learning the correlations between the various tasks.

During the course of this thesis, several activities were undertaken. Initially, we examined the state of the art, analyzed available datasets, and common solutions to address this problem. Eventually, we selected the RSCD Dataset [13] as the foundation for the subsequent phases of this study. Subsequently, we developed a complex model as a personal contribution to this research and conducted an in-depth analysis of the dataset used.

The results obtained revealed that the final model, based on a multi-task neural network, achieved significant performance improvements compared to the

other evaluated models, enhancing performance compared to single-task models. This demonstrates a considerable potential contribution to this type of problem. Furthermore, the dataset used presented various challenges, highlighting the non-trivial nature of the problem and the need to expand it significantly to enable truly usable solutions.

As for possible future developments, several directions emerge. Exploring advanced multi-task models, developing innovative methods to handle the class imbalance inevitably present in this problem, all of this could further improve performance. The adoption of combined CNN and RNN neural networks that can assist in capturing spatial and temporal correlations between data, as already discussed in this work, could also be explored. Lastly, expanding the dataset is perhaps the most essential aspect of continued research progress. The results obtained indicate that road condition classification with neural networks holds significant potential. Future developments in advanced models, larger datasets, and specific applications can lead to significant advancements in this field.

In conclusion, it is believed that the integration of artificial intelligence with multi-task neural networks would greatly enrich the capabilities of advanced ADAS, opening up new horizons in terms of driving safety and reliability.

References

[1] Fisher Yu et al. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2636–2645.

[2] Pei Sun et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2446–2454.

[3] Marcus Nolte, Nikita Kister, and Markus Maurer. "Assessment of deep convolutional neural networks for road surface classification". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 381–386.

[4] Zachary Pezzementi et al. "Comparing apples and oranges: Off-road pedestrian detection on the National Robotics Engineering Center agricultural person-detection dataset". In: *Journal of Field Robotics* 35.4 (2018), pp. 545–563.

[5] Andreas Geiger et al. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[6] Will Maddern et al. "1 year, 1000 km: The oxford robotcar dataset". In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15.

[7] Mike Smith et al. "The new college vision and laser data set". In: *The International Journal of Robotics Research* 28.5 (2009), pp. 595–599.

[8] Alessandro Giusti et al. "A machine learning approach to visual perception of forest trails for mobile robots". In: *IEEE Robotics and Automation Letters* 1.2 (2015), pp. 661–667.

[9] Tobias Nothdurft et al. "Stadtpilot: First fully autonomous test drives in urban traffic". In: *2011 14th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE. 2011, pp. 919–924.

[10] Ronny Stricker et al. “Improving visual road condition assessment by extensive experiments on the extended gaps dataset”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[11] Kai Cordes et al. “RoadSaW: A Large-Scale Dataset for Camera-Based Road Surface and Wetness Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4440–4449.

[12] Tong Zhao and Yintao Wei. “A road surface image dataset with detailed annotations for driving assistance applications”. In: *Data in Brief* 43 (2022), p. 108483. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2022.108483. URL: https://www.sciencedirect.com/science/article/pii/S2352340922006771.

[13] 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* (2023).

[14] R. Rajamani et al. “Algorithms for real-time estimation of individual wheel tire-road friction coefficients”. In: *IEEE/ASME Transactions on Mechatronics* 17.6 (Dec. 2012), pp. 1183–1195.

[15] Y-H Liu et al. “Estimation of tire-road friction coefficient based on combined APF-IEKF and iteration algorithm”. In: *Mechanical Systems and Signal Processing* 88 (2017), pp. 25–35.

[16] J. Park et al. “Road surface classification using a deep ensemble network with sensor feature selection”. In: *Sensors* 18.12 (Dec. 2018). DOI: 10.3390/s18124342.

[17] M.-H. Kim, J. Park, and S. Choi. “Road type identification ahead of the tire using D-CNN and reflected ultrasonic signals”. In: *International Journal of Automotive Technology* 22.1 (Feb. 2021), pp. 47–54.

[18] A. Dhiman and R. Klette. “Pothole detection using computer vision and learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (Aug. 2020), pp. 3536–3550.

[19] M. Nolte, N. Kister, and M. Maurer. "Assessment of deep convolutional neural networks for road surface classification". In: *Proceedings of the 21st IEEE Intelligent Transportation Systems Conference*. Nov. 2018, pp. 381–386.

[20] S. Roychowdhury et al. "Machine learning models for road surface and friction estimation using front-camera images". In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. July 2018, pp. 1–8.

[21] B. Leng et al. "Estimation of tire-road peak adhesion coefficient for intelligent electric vehicles based on camera and tire dynamics information fusion". In: *Mechanical Systems and Signal Processing* 150 (Mar. 2021). DOI: 10.1016/j.ymssp.2020.107275.

[22] Tong Zhao and Yintao Wei. "A road surface image dataset with detailed annotations for driving assistance applications". In: *Data in Brief* 43 (2022), p. 108483.

[23] Weiwei Liu et al. "The emerging trends of multi-label learning". In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7955–7974.

[24] Min-Ling Zhang and Zhi-Hua Zhou. "Multilabel neural networks with applications to functional genomics and text categorization". In: *IEEE transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351.

[25] Jinseok Nam et al. "Large-scale multi-label text classification—revisiting neural networks". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14*. Springer. 2014, pp. 437–452.

[26] Chih-Kuan Yeh et al. "Learning deep latent space for multi-label classification". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.

[27] Kaixiang Wang et al. "Deep correlation structure preserved label space embedding for multi-label classification". In: *Asian Conference on Machine Learning*. PMLR. 2018, pp. 1–16.

[28] Xiaobo Shen et al. "Deep binary prototype multi-label learning". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 2675–2681.

[29] Feng Zhu et al. “Learning spatial regularization with image-level supervisions for multi-label image classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5513–5522.

[30] Renchun You et al. “Cross-modality attention with semantic graph embedding for multi-label classification”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 12709–12716.

[31] Yongcheng Liu et al. “Multi-label image classification via knowledge distillation from weakly-supervised detection”. In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 700–708.

[32] Jingzhou Liu et al. “Deep learning for extreme multi-label text classification”. In: *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 2017, pp. 115–124.

[33] Ronghui You et al. “Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[34] W Bingyu et al. “Ranking-Based Autoencoder for Extreme Multi-label Classification”. In: *arXiv preprint arXiv:1904.05937* (2019).

[35] Chuan Guo et al. “Breaking the glass ceiling for embedding-based classifiers for large output spaces”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[36] Wei-Cheng Chang et al. “Taming pretrained transformers for extreme multi-label text classification”. In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 3163–3171.

[37] Kunal Dahiya et al. “Deepxml: A deep extreme multi-label learning framework applied to short text documents”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 31–39.

[38] Anshul Mittal et al. “Decaf: Deep extreme classification with label features”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 49–57.

[39] Anshul Mittal et al. “ECLARE: Extreme classification with label graph correlations”. In: *Proceedings of the Web Conference 2021*. 2021, pp. 3721–3732.

[40] Deepak Saini et al. “GalaXC: Graph neural networks with labelwise attention for extreme classification”. In: *Proceedings of the Web Conference 2021*. 2021, pp. 3733–3744.

[41] Moustapha Cissé, Maruan Al-Shedivat, and Samy Bengio. “Adios: Architectures deep in output space”. In: *International Conference on Machine Learning*. PMLR. 2016, pp. 2770–2779.

[42] Zhi-Hua Zhou and Ji Feng. “Deep forest”. In: *National science review* 6.1 (2019), pp. 74–86.

[43] Jie Zhou et al. “Graph neural networks: A review of methods and applications”. In: *AI open* 1 (2020), pp. 57–81.

[44] Zhao-Min Chen et al. “Multi-label image recognition with graph convolutional networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5177–5186.

[45] Tianshui Chen et al. “Learning semantic-specific graph representation for multi-label image recognition”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 522–531.

[46] Ya Wang et al. “Multi-label classification with label graph superimposing”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 12265–12272.

[47] Jinseok Nam et al. “Maximizing subset accuracy with recurrent neural networks in multi-label classification”. In: *Advances in neural information processing systems* 30 (2017).

[48] Jiang Wang et al. “Cnn-rnn: A unified framework for multi-label image classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2285–2294.

[49] Zhouxia Wang et al. “Multi-label image recognition by recurrently discovering attentional regions”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 464–472.

[50] Shang-Fu Chen et al. “Order-free rnn with visual attention for multi-label classification”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[51] Jinseok Nam et al. “Learning context-dependent label permutations for multi-label classification”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4733–4742.

[52] Che-Ping Tsai and Hung-Yi Lee. “Order-free learning alleviating exposure bias in multi-label classification”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6038–6045.

[53] Vacit Oguz Yazici et al. “Orderless recurrent models for multi-label classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13440–13449.

[54] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[55] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[56] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115 (2015), pp. 211–252.

[57] Zhao Chen et al. “Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks”. In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[58] Divyansh Puri. “COCO Dataset Stuff Segmentation Challenge”. In: *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*. 2019, pp. 1–5. DOI: 10.1109/ICCUBEA47591.2019.9129255.

[59] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[60] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[61] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

List of Figures

List of Tables