



UNIVERSITÀ
DEGLI STUDI
FIRENZE



UNIVERSITÀ
DEGLI STUDI
DI PERUGIA

[iNSAM]
Istituto Nazionale
di Alta Matematica

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

**DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA
CURRICULUM IN INFORMATICA
CICLO XXXV**

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Matteo Focardi

**Robust Recognition of Objects in
the Safety Critical Systems: A
Case Study of Traffic Sign
Recognition**

Settore Scientifico Disciplinare INF/01

Dottorando:
Muhammad Atif

Tutore
Prof. Andrea Bondavalli

Coordinatore
Prof. Matteo Focardi

ABSTRACT

Computer vision allows to automatically detect and recognize objects from images. Nowadays, timely detection of relevant events and efficient recognition of objects in an environment is a critical activity for many Cyber-Physical Systems (CPSs). Particularly, the detection and recognition of traffic signs (TSDR) from images was and is currently being investigated, as it heavily impacts the behaviour of (semi-)autonomous vehicles. TSDR provides drivers with critical traffic sign information, constituting an enabling condition for autonomous driving and attaining a safe circulation of road vehicles. Misclassifying even a single sign may constitute a severe hazard to the environment, infrastructures, and human lives. In the last decades, researchers, practitioners, and companies have worked to devise more efficient and accurate Traffic Sign Recognition (TSR) subsystems or components to be integrated into CPSs. Mostly TSR relies on the same main blocks, namely: i) Datasets creation/identification and pre-processing (e.g., histogram equalization for improvement of contrast), ii) Feature extraction, i.e., Keypoint Detection and Feature Description, and iii) Model learning through non-deep or Deep Neural Networks (DNNs) classifiers. Unfortunately, despite many classifiers and feature extraction strategies applied to images sampled by sensors installed on vehicles that have been developed throughout the years; those efforts did not escalate into a clear benchmark nor provide a comparison of the most common techniques.

The main target of this thesis is to improve the robustness and efficiency of TSR systems. Improving the efficiency of the TSR system means achieving better classification performance (classification accuracy) on publicly available datasets, while the robustness of an image classifier is defined as sustaining the performance of the model under various image corruptions or alterations that in our case due to visual camera malfunctioning. Albeit TSDR embraces both detection and recognition of traffic signs, here we focus on the latter aspect of recognition. In the literature, many researchers proposed different techniques for the detection of traffic signs in a full-scene image. Therefore, this thesis starts by providing a comprehensive quantitative comparison of non-deep Machine Learning (ML) algorithms with different feature sets and DNNs for the recognition of traffic signs from three publicly available datasets.

Afterward, we propose a TSR system that analyses a sliding window of images instead of considering individual images sampled by sensors on a vehicle. Such TSR processes the last

image and recent images sampled by sensors through ML algorithms that take advantage of these multiple information. Particularly, we focused on (i) Long Short-Term Memory (LSTM) networks and (ii) Stacking Meta-Learners, which allow for efficiently combining base-learning classification episodes into a unified and improved meta-level classification. Experimental results by using publicly available datasets show that Stacking Meta-Learners dramatically reduce misclassifications of traffic signs and achieve perfect classification on all three considered datasets. This shows the potential of our novel approach based on sliding windows to be used as an efficient solution for TSR.

Furthermore, we consider the failures of visual cameras installed on vehicles that may compromise the correct acquisition of images, delivering a corrupted image to the TSR system. After going through the most common camera failures, we artificially injected 13 different types of visual camera failures into each image contained in the three traffic sign datasets. Then, we train three DNNs to classify a single image, and compare them to our TSR system that uses a sequence (i.e., a sliding window) of images. Experimental results show that sliding windows significantly improve the robustness of the TSR system against altered images. Further, we dig into the results using LIME, a toolbox for explainable Artificial Intelligence (AI). Explainable AI allows an understanding of how a classifier uses the input image to derive its output. We confirm our observations through explainable AI, which allows understand why different classifiers have different TSR performances in case of visual camera failures.

Visual camera failures have a negative impact on TSR systems as they may lead to the creation of altered images: therefore, it is of utmost importance to build image classifiers that are robust to those failures. As such, this part of the thesis explores techniques to make TSR systems robust to visual camera failures such as broken lens, blurring, brightness, dead pixels or no noise reduction by image signal processor. Particularly, we discuss to what extent training image classifiers with images altered due to camera failures can improve the robustness of the whole TSR system. Results show that augmenting the training set with altered images significantly improves the overall classification performance of DNN image classifiers and makes the classifier robust against the majority of visual camera failures. In addition, we found that no noise reduction and brightness visual camera failures have a major impact on image classification. We discuss how image classifiers trained using altered images have better accuracy compared to classifiers trained only with original images, even in presence of such failures.

Ultimately, we further improve the robustness of the TSR system by crafting a camera failure detector component in conjunction with image classifiers trained using altered images that further enhance the robustness of the TSR system against visual camera failures. We trained different ML-based camera failure detectors (binary classifiers) that work in sequence with DNN to check the images are altered to a certain level of failure severity. Based on the output of the failure detector, we decide that either image will be passed to DNN for classification or will alert the user that the image is severely degraded by visual camera failure and DNN may not be able to classify the image correctly. Experimental results reveal that the failure detector component in conjunction with image classifiers trained using altered images enhances the performance of the TSR system compared to the image classifiers trained using altered images, but it reduces the availability of the system which is 100% in case of image classifiers trained using altered images without camera failure detector component.

ACKNOWLEDGEMENTS

My participation in this Ph.D. has been an extraordinary life-changing experience, and I would not have been able to do it without the support and guidance of many individuals. First and foremost, with great gratitude, I extend my sincere appreciation to my supervisor, Prof. Andrea Bondavalli, Professor at the Department of Mathematics and Informatics and head of the Resilient Computing Lab (RCL) for his advice, encouragement, constant support, continuous guidance, and valuable suggestion during my Ph.D.

I would also wish to express my gratitude to Tommaso Zoppi, Andrea Ceccarelli, and Mohamad Gharib for extended discussions, valuable suggestions, and their continuous support from the first day that contributed greatly to the improvement of the thesis. I would also like to thank Paolo Lollini, who was a great help from the moment I received my doctoral offer until I arrived in Italy. Moreover, Paolo and Tommaso gave me a course on "Quantitative Analysis of Systems", which was very interesting and provided me with a lot of new information. Prof. Andrea Bondavalli, Andrea Ceccarelli and Mohamad Gharib taught me an important course "Distributed Real Time Cyber Physical Systems" to understand many new aspects of Safety Critical Systems and System of Systems that gave me the ability to understand and analyze the behavior of different real time systems. I have always found them to be very friendly and helpful and their doors were always open to me when I needed advice and support.

I am also thankful to the RCL members Maria Maqsood, Mirko Staderini, Francesco Terrosi, Francesco Mariotti, and Tommaso Puccetti with whom I shared the lab during these three years of my Ph.D. and always found them very welcoming and cooperative.

Finally, I would like to thank my parents for their love, support, and continuous sacrifices from my childhood till now to always see me better than yesterday. Without them, this day would not have been possible. I would also like to thank my siblings and wife for their support and good times in my life.

LIST OF RELEVANT PUBLICATIONS

This section highlights the publication produced in the context of the research work done in this thesis.

1. **Atif, M.**, Zoppi, T., Gharib, M., & Bondavalli, A. (2021, March). Quantitative comparison of supervised algorithms and feature sets for traffic sign recognition. In Proceedings of the 36th Annual ACM Symposium on Applied Computing (pp. 174-177).
2. **Atif, M.**, Ceccarelli, A., & Bondavalli, A. (2021) Reliable Traffic Sign Recognition System. Anais Estendidos do X Latin-American Symposium on Dependable Computing.
3. **Atif, M.**, Zoppi, T., Gharib, M., & Bondavalli, A. (2022). Towards enhancing traffic sign recognition through sliding windows. *Sensors*, 22(7), 2683.
4. **Atif, M.**, Ceccarelli, A., Zoppi, T., Gharib, M., & Bondavalli, A. (2022). Robust Traffic Sign Recognition against Camera Failures. *IEEE Open Journal of Intelligent Transportation Systems*.
5. **Atif, M.**, Ceccarelli, A., Zoppi, T., & Bondavalli, A. (2023). Tolerate Failures of the Visual Camera with Robust Image Classifiers. *IEEE Access*.

Other field related publications produced during the period of Ph.D. are the following

6. Zoppi, T., Gharib, M., **Atif, M.**, & Bondavalli, A. " Meta-Learning to Improve Unsupervised Intrusion Detection in Cyber-Physical Systems." *ACM Transactions on Cyber-Physical Systems*.
7. **Atif, M.**, Franzoni, V., & Milani, A. (2021, September). Emojis Pictogram Classification for Semantic Recognition of Emotional Context. In International Conference on Brain Informatics (pp. 146-156). Springer, Cham.
8. **Atif, M.**, & Franzoni, V. (2022). Tell Me More: Automating Emojis Classification for Better Accessibility and Emotional Context Recognition. *Future Internet*, 14(5), 142.

CONTENTS

1. Introduction.....	16
1.1 Structure of the Thesis	17
1.2 Comparative Analysis of Non-deep and Deep Neural Network Classifiers for Traffic Sign Recognition System.....	18
1.3 Enhancing Traffic Sign Recognition through Sliding Windows	19
1.4 Robust Traffic Sign Recognition against Visual Camera Failures	20
1.5 Improving Robustness of Deep Neural Network Image Classifiers against Visual Camera Failures	21
1.6 Main Contributions of the Thesis	22
1.7 Limitations to Validity	23
1.7.1 Usage of Public Data.....	23
1.7.2 Parameters of Classifiers	23
1.7.3 Classifier Performance Metric.....	23
1.7.4 Injection of Camera Failures to Traffic Signs	24
2. Basics and Literature Review	25
2.1 Basics of Dependability.....	25
2.1.1 System, Service, Errors, Faults, and Failures	25
2.1.2 Dependability and its Characteristics	27
2.2 Overview of Basic Building Blocks of TSR system.....	28
2.2.1 Feature Detectors and Descriptors.....	28
2.2.2 Deep Features	30
2.2.3 Non-deep Classifiers	30
2.2.4 Deep Neural Networks	31
2.2.5 Stacking Meta-level Classifiers.....	32
2.2.6 Long Short-Term Memory Networks (LSTM).....	33
2.2.7 Traffic Sign Datasets	33
2.2.8 Classification Performance Metric	34
2.3 Related Works on Different TSR Approaches	36
2.4 Background on Visual Camera Failures	39
2.5 Related Works on Robust Image Classifiers	40
2.5.1 Architectural Solutions for Robust DNNs.....	41
2.5.2 Out-of-Distribution Detection	41
2.5.3 Data Augmentation for Robustness.....	41
2.5.4 Data Augmentation for Adversarial Defense	41
3. Comparative Analysis of non-deep and Deep Neural Network Classifiers for Traffic Sign Recognition System	44
3.1 Motivation behind Comparative Analysis	44
3.2 A Fair Comparison of TSR Systems	45
3.2.1 TSR Datasets and Traffic Sign Categories	45
3.2.2 Keypoint Detectors and Feature Descriptors.....	45
3.2.3 Non-deep Classifiers and their Parameters.....	46
3.2.4 Deep Neural Networks for TSR	47

3.3	Experimental Setup	47
3.4	Experimental Results	48
3.4.1	Results of Non-deep Classifiers	49
3.4.2	Results of Deep Neural Networks	53
3.5	Lessons Learned	54
4.	Enhancing Traffic Sign Recognition through Sliding Windows	56
4.1	Sliding Windows to Improve TSR	57
4.1.1	Sliding Windows and Meta-Learning	57
4.2	A Stacking Meta-Learner	58
4.3	Methodology, Inputs, and Experimental Setup	59
4.3.1	Methodology for a Fair Comparison of TSR Systems	59
4.3.2	TSR Datasets and Traffic Sign Categories	60
4.3.3	Feature Descriptors	60
4.3.4	Classification Metrics	60
4.3.5	Non-deep Classifiers and Deep Neural Network Hyper-Parameters	60
4.3.6	Long-Short Term Memory (LSTM) Networks	62
4.4	Experimental Results	62
4.4.1	TSR Based on Single Image	62
4.4.2	TSR Based on Sliding Windows	65
4.4.3	Comparing Sliding Windows and Single-Image Classifiers	68
4.5	In-Depth View of BelgiumTSC	70
4.6	Timing Analysis	72
4.7	Comparison to the State of the Art TSR	73
4.8	Lessons Learned	74
5.	Robust Traffic Sign Recognition against Visual Camera Failures	76
5.1	TSR Strategies Under Consideration	77
5.1.1	Single-image Classification	77
5.1.2	Classification with a Sliding Window	78
5.2	Approach to Robustness Evaluation	79
5.2.1	Methodology	79
5.2.2	Base-level Classifiers	80
5.2.3	Meta-level Classifiers	80
5.2.4	Strategies to Inject Visual Camera Failures	80
5.2.5	Performance Metrics	82
5.3	Experimental Results	82
5.3.1	Results on the Clean Datasets	83
5.3.2	1SFC on the Injected Datasets	84
5.3.3	2SFC and 3SFC on the Injected Datasets	85
5.3.4	W1C on the Injected Datasets	87
5.3.5	W2C and W3C on the Injected Datasets	87
5.4	Explanation Of DNNS Robustness	90
5.5	Lessons Learned	91

6.	Improving Robustness of Deep Image Classifiers against Visual Camera Failures	94
6.1	Robust TSR System Against Visual Camera Failures	95
6.1.1	Main Building Blocks	95
6.1.2	Strategies for Building Robust DNNs	95
6.1.3	Comparison of Strategies to Improve Robustness	97
6.2	Experimental Campaign and Methodology	97
6.2.1	Injection of Visual Camera Failures	98
6.2.2	Building Train and Test Sets for Classifiers	98
6.2.3	DNNs for Traffic Sign Recognition	99
6.2.4	Building Train and Test Sets for Binary Failure Detector	99
6.2.5	Input Features for Binary Failure Detectors Training and Testing	100
6.2.6	Classifiers Parameters for Binary Failure Detectors.....	100
6.2.7	Performance Metrics	100
6.3	Experimental Results of Augmented Classifiers	101
6.3.1	Are we building Robust Image Classifiers?	101
6.3.2	Effect of Brightness and Noise on Image Classifiers	104
6.3.3	On the Impact of Individual Failures on Training	106
6.3.4	Explaining Performance of Regular and Augmented Classifiers on Clean Images.....	107
6.4	Experimental Results of Camera Failure Detector	109
6.5	Experimental Results of TSR System using CFD	110
6.6	Lessons Learned	113
7.	Conclusions and Future Directions	116
7.1	Conclusions	116
7.2	Future Works.....	117

LIST OF TABLES

Table 1: Details of the three datasets used in this study	33
Table 2: Categorization of Traffic Signs into 9 categories based on their shape, color, and content.	34
Table 3: Combination of key point detectors and descriptors	46
Table 4: Average \pm standard deviation of accuracy, specificity and recall of 8 non-deep classifiers on 3 datasets.	49
Table 5: Highest Accuracy achieved using different keypoint detectors and feature extractors on each dataset. Grayed lines in the bottom of the table highlight combinations of items above.....	51
Table 6: Accuracy achieved on three deep learning models for each of three datasets varying learning rates.....	53
Table 7: Highest accuracy achieved using different feature descriptors on each dataset (bold high-lighted values represent the highest achieved accuracy across each dataset).	64
Table 8: Accuracy achieved by deep learners for each of the three datasets with varying learning rates (bold highlighted values represent the highest achieved accuracy across each dataset by deep classifiers).	65
Table 9: Results achieved using different meta-learners considering non-deep classifiers as base classifiers varying window size (WS). We bolded the highest achieved accuracy using different combinations across each dataset and low accuracy achieved through AdaBoostM2 and Random Forest italicized numbers in the 10th and 11th columns.....	66
Table 10: Results achieved using different meta learners with DNNs as base classifiers with varying window size (WS). We bolded the perfect classification (100% accuracy).	67
Table 11: Accuracy of LSTM with window sizes 2 and 3.....	68
Table 12: Time required for (left) feature extraction, (middle) exercising individual classifiers, and (right) different TSR strategies, either sequential or parallel execution.	72
Table 13: Comparison with state of the art approaches. Gray area in the table shows our result approach using single frame and sliding windows result for 3 and 9 traffic sign categories.....	74
Table 14: Minimum and maximum accuracy drop of 1SFC when it is exercised on the injected datasets and the clean datasets.	84
Table 15: Minimum and maximum accuracy drop of 2SFC and 3SFC on the injected datasets with respect to the clean datasets.	86
Table 16: Minimum and maximum accuracy drop of W1C on the injected datasets with respect to the original datasets.	88
Table 17: Minimum and maximum accuracy drop of W2C and W3C on the injected datasets with respect to the clean datasets.	89
Table 18: Comparison of different strategies to build robust Image Classifiers.	97
Table 19: Accuracy achieved using regular classifiers on the three datasets.....	102
Table 20: Accuracy achieved using augmented classifiers on the three datasets.	103
Table 21: Difference between the minimum Accuracy of a regular classifier and the minimum accuracy obtained by MN2 on 14 test sets of DITS. MN2 is trained using train_clean plus data of a single visual camera failure.	106

Table 22: Accuracy achieved by augmented classifiers on 40% test set containing all failure samples with same proportion	110
Table 23: Overall TSR system performance on BelgiumTSC dataset, for each augmented and CFD classifiers records are selected by highest F-AA calculated between accuracy and availability.	111
Table 24: Overall TSR system performance on DITS dataset, for each augmented and CFD classifiers records are selected by highest F-AA calculated between accuracy and availability.....	112

LIST OF FIGURES

Figure 1: Structure of the thesis.....	17
Figure 2: Elementary fault classes from eight viewpoints [136]	26
Figure 3: A visual camera and its components: the Lens, and the camera body composed of Bayer Filter, Image Sensor, and Image Signal Processor.....	40
Figure 4: Block Diagram of Traffic Sign Recognition System	48
Figure 5: Highest Accuracy achieved by Non-deep Classifiers.....	50
Figure 6: Impact of training function for FFNN and CNN. Bar chart shows average and standard deviation of top accuracy configurations for each of the three datasets.	52
Figure 7: Impact of SVM's Kernels on maximum accuracy reached on each dataset. Plot also reports on Specificity and Recall.	52
Figure 8 a and b. Specificity, Accuracy and Recall achieved by most performing non-deep classifiers (Boosting, KNN, SVM) and deep learners (InceptionV3, MobileNet-V2) on BelgiumTSC (left, Figure 8a) and on DITS (right, Figure 8b) datasets.	55
Figure 9: Example of Sliding Windows. Dotted, dashed and solid boxes show sliding windows, respectively, at t5, t4, t3.	57
Figure 10: Diagram representing TSR which uses sliding windows. Blue solid boxes represent single-image classifier in Figure 4.....	59
Figure 11: Highest accuracy achieved by non-deep classifiers on each dataset.	63
Figure 12: Highest accuracy achieved by LSTM, stacker with non-deep base-level, and stacker with deep base-level on BelgiumTSC and DITS.	69
Figure 13: Instantiation of the stacking-meta learner with AlexNet base-learner and SVM meta-level learner, managing a sliding window of size 3 for BelgiumTSC. The three images we use as input DNNs describe a Diamond sign (Category 7) which is misclassified using all three images.....	70
Figure 14: Instantiation of the Stacking-Meta learner with AlexNet Base-learner and SVM meta-level learner, managing a sliding window of size 2 for the BelgiumTSC. The three images we use as input describe a Diamond sign (Category 7) which is misclassified using all three images (Figure 13) but may be classified correctly by using a shorter window.....	71
Figure 15: A TSR system that uses multiple (three) single-image classifiers with stacking.	77
Figure 16: A TSR which uses sliding windows of three images (on the left) with two base-classifiers (AlexNet, InceptionV3) and a meta-level classifier that produces the final classification result PTS_{final}	78
Figure 17: Injection of 13 different visual camera failures to a sample traffic sign. (best viewed in color).....	82
Figure 18: Highest accuracy achieved on the clean datasets GTSRB, BelgiumTSC, DITS.....	83
Figure 19: Explanation of AN, IC3, and MN2 models through LIME, highlighting the influential features of selected traffic sign images. (best viewed in color)	90

Figure 20: Accuracy achieved for each failure on the three datasets, when we consider the failure configurations which lead to the highest accuracy drop.	92
Figure 21: General structure of an image classifier that embeds a DNN, plus two strategies for robustness.	96
Figure 22: Process to assign binary labels to each test_clean and test_altered image of three datasets classified independently by each augmented classifier.	99
Figure 23: Box Plots showing accuracy of the three regular classifiers (Figure 23a) and the three augmented classifiers (Figure 23b) on the three datasets, for the test_clean and the 13 test_<failure>.	101
Figure 24: Accuracy achieved on test_brightness by regular classifiers (left) and augmented classifiers (right) using the 3 DNNs AN, MN2, IC3 on the 3 datasets DITS, BelgiumTSC, GTSRB and brightness levels in the range [0.1 to 15].	104
Figure 25: Accuracy achieved on test_NNR by regular classifiers (Figure 25a, on the left) and augmented classifiers (Figure 25b, on the right) using different DNNs on different datasets with different levels of noise (0 to 5).	105
Figure 26: Different brightness and noise levels applied to the same traffic sign.	106
Figure 27: Explanation of predictions of regular (on the left) and augmented (on the right) classifiers using LIME. Figure reads better when viewed in colors.	108
Figure 28: Performance of different binary CFD classifiers on two datasets i.e., BelgiumTSC and DITS by independently considering each Augmented Classifier.	109

1. INTRODUCTION

Intelligent transportation systems are nowadays of utmost interest for researchers and practitioners as they aim at providing advanced and automatized functionalities, such as obstacle detection, traffic sign recognition, car plate recognition, and automatic incident detection or stopped vehicle detection systems. Particularly, Traffic Sign Detection and Recognition (TSDR) systems aim at detecting (TSD) and recognizing (TSR) traffic signs from images sampled by sensors [1], [2], [3] installed on vehicles. Those systems synergize with the human driver, who may misinterpret or miss an important traffic sign, potentially leading to accidents that may generate safety-related hazards [4]. When integrated into intelligent vehicles [5], [6] for Advanced Driver-Assistance Systems (ADAS) [2], [7], [8], [9], TSDR can automatically provide drivers with actionable warnings or even trigger reaction strategies (e.g., automatic reduction of speed, braking) that may be crucial to avoid or reduce the likelihood of accidents [3], [10].

Humans are expected to naturally miss or misinterpret important traffic signs occasionally due to fatigue, adverse operating conditions or distraction [11]. Similarly, to humans, TSDR systems are also subject to errors as they may misinterpret or miss a traffic sign due to various reasons, such as unsatisfactory road situations, imperfect traffic sign state, adverse environmental conditions (e.g., foggy weather [12]) or imperfect analysis processes. Nevertheless, researchers and practitioners are trying to minimize those events, called misclassifications, expecting automatic TSDR systems to provide drivers with accurate and timely notifications. Available TSD systems can precisely extract areas of an image, which are supposed to contain a traffic sign. Thereto, TSR systems that embed Machine Learning (ML) algorithms [13], [14], [15], [16], process features that are extracted from those images through feature descriptors (e.g., Histogram of Oriented Gradients (HOG) [17], Local Binary Pattern (LBP) [16] to recognize traffic sign categories [18], [19]. Alternatively, Deep Neural Networks (DNNs), such as AlexNet and googLeNet [20] can directly process images as they embed internal representation learning processes, which are orchestrated through multiple convolutional and fully connected layers. The focus of this study is to design a robust and efficient TSR system that provides correct service and minimize the catastrophic failures that impact the users (drivers) and environment. To investigate the performance of TSR, we perform a quantitative comparison and based on the results of comparative study, we adopt sliding windows based approach that increase the efficiency of TSR. Furthermore, to increase

the robustness of TSR against camera failures, we adopted different strategies such as data augmentation and data augmentation in conjunction with camera failure detector component for the prevention of faults to design a robust TSR system that is safe enough for the users and environment.

1.1 Structure of the Thesis

This section provides an overview of our contribution towards this thesis with the help of Figure 1, where different chapters of the thesis are highlighted with the main contributions. First, chapter 2 provides an overview of different state of the art approaches adopted for TSR, different building blocks to implement a TSR system and different camera related failures. This thesis mainly focuses on two aspects such as i) improving the efficiency of TSR system, and ii) improving the robustness of TSR system against visual camera failures. Chapter 3 and Chapter 4 focuses on the enhancement of the TSR efficiency, while Chapter 5 and Chapter 6 explore different strategies when visual camera sensor is malfunctioning. To such extent, Chapter 3 is the first experimental chapter, where we perform the quantitative comparison of different non-deep classifiers train on different feature descriptors and DNNs for single image TSR. While in Chapter 4, we proposed a sliding windows-based approach for enhancing the TSR performance considering subsequent sequences of images of a single traffic sign, and the proposed approach for TSR is experimentally evaluated on three publicly available datasets.

In the second part of the thesis, the focus is on the robustness of TSR system against visual camera failures. In Chapter 5, we implement different classification strategies such as single

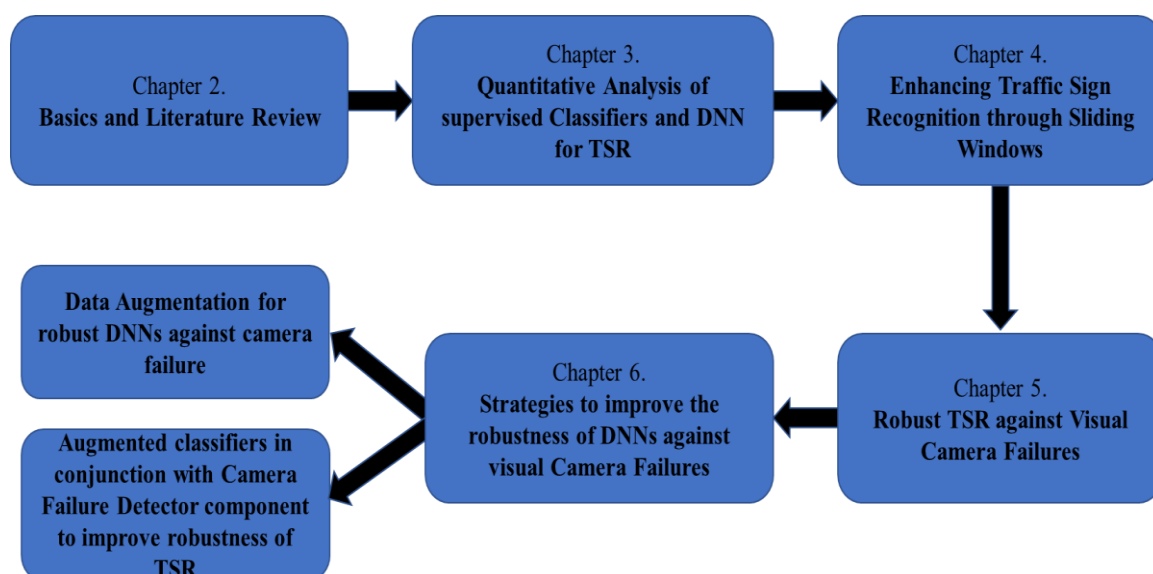


Figure 1: Structure of the thesis

classifier for single image classification, multiple classifiers for single image classification, single classifier for sliding window based classification and multiple classifiers for sliding window based classification, to overcome the effect of visual camera failures on TSR system. Furthermore, Chapter 6 discuss two different strategies for robust TSR system against visual camera failures such as (i) data augmentation and (ii) data augmentation in conjunction camera failure detector and these different approaches are validated through extensive experiments. Finally, Chapter 7 concludes the thesis with possible future directions.

Further sections briefly explain the contribution of each experimental chapter shown in Figure 1 i.e., chapter 3 to chapter 6.

1.2 Comparative Analysis of Non-deep and Deep Neural Network Classifiers for Traffic Sign Recognition System

Throughout the years, many studies tackles TSR [21], [22], [23] using different feature descriptors and ML-based classifiers. Different combinations of such classifiers and features have been proven to generate heterogeneous classification scores [15], [19], [24], [25], [26], [27] motivating the need for comparisons to discover the optimal classifier for a given TSR problem [3], [28], [29]. Most of these studies exercise a few classifiers or a single feature set on a specific dataset. This does not provide sufficient means to compare different approaches. To such extent, our study considers 3 public datasets namely i) the GTSRB dataset [31], ii) the BelgiumTSC dataset [32], and iii) the Dataset of Italian Traffic Signs (DITS) [33], which report on sequences of images related to a single traffic sign. We are using a pool of eight non-deep ML algorithms for the recognition of traffic signs. Images are first processed to extract 14 different feature sets and additionally 12 different combination of feature descriptors i.e., fusing 2 or more feature descriptors: each feature set is then provided as input to the non-deep ML algorithms for TSR. Furthermore, we employ also 3 DNNs– namely InceptionV3 [40], MobileNet-V2 [41] and AlexNet [34] - that do not need feature sets as they automatically extract feature through convolutional layers while training through transfer learning. Altogether, these experiments build a quantitative comparison of different ML algorithms to benchmark different approaches by using heterogeneous datasets that provides a baseline for practitioners and researchers who are willing to setup a TSR component to be embedded in autonomous vehicles. Results allow to elaborate the performance of non-deep classifiers trained on different feature sets, and DNNs trained using transfer learning. But still, we did not find a single classifier that achieve perfect classification i.e., 100% accuracy on all three datasets.

1.3 Enhancing Traffic Sign Recognition through Sliding Windows

Based on the results of extensive quantitative comparison, we could not find a single classifier that perform perfect classification on all three datasets. This is mostly due to the fact that existing solutions for TSR process a single image and output a classification result. Instead, vehicles are meant to gradually approach traffic signs during their road trips, generating sequences of images: the closer the vehicle is to the traffic sign, the better the quality of the image, even under slightly different environmental conditions. Therefore, the problem of TSR naturally scales from single images to knowledge extraction from a set or sequence of images that potentially contain traffic signs. Accordingly, the classification process should not depend only on a single image to make a decision; instead, it should take advantage of the knowledge acquired as the vehicle moves forward, i.e., the sequence of images. We consider a sliding window of images to commit classification rather than classifying images individually. First, we process each image with the most effective single image classifier for TSR: then, we combine classification scores assigned to images in the sliding window to provide a unified and improved classification result. Such a combination is performed by appropriate Meta-Learners [30], which suit model combination, and therefore, show potential to be applied in such a context. We use non-deep classifiers both as single-image classifiers and as base-level learners of a Stacking meta-learner, which aggregates individual classification scores into sliding windows. The meta-level classifier for Stacking are experimentally chosen out of non-deep classifiers, the Majority Voting [42] and Discrete Hidden Markov Model (DHMM, [43]). Additionally, we compare the classification performance of those meta-learners with Long Short-Term Memory (LSTM) networks, which naturally deal with sequences of data coming at different time instants. We trained those LSTM networks on the same sliding windows of images processed through Stacking. Results show how single-image classifiers achieve 100% accuracy on the GTRSB, 99.72% on BelgiumTSC and 96.03% on DITS datasets. Then, we applied our approach based on sliding windows by using LSTM networks and stacking meta-learners, finding that both approaches greatly improve the accuracy of TSR: particularly, specific stacking meta-learners achieved perfect accuracy (i.e., no misclassifications at all) on the three datasets by using a sliding window of two or three images.

1.4 Robust Traffic Sign Recognition against Visual Camera Failures

Automatic TSR systems embed ML, and especially DNN classifiers which process image images captured by visual cameras installed on the vehicle. Those TSR systems are known to accurately classify traffic signs, even reaching perfect classification performance under nominal operating conditions [3], [29], [87]. Unfortunately, the adverse environmental conditions, or the malfunctions of the visual camera, may produce low-quality images that may negatively impact the performance of classifiers. Examples include, but are not limited to occlusions, shadows, defects of the visual camera lens, changes in environmental light, raindrops on the camera lens, out of focus, and flare [85], [89]. Therefore, to guarantee safety of the driving task, it is necessary to study the robustness of TSR systems against the threats, and develop solutions to tolerate them [90], [91]. Robustness of an image classifier is defined as sustaining the performance of the model under various image alterations. To accomplish this task, first reviewing the challenges and the state of the art, and then proposing and evaluating alternative solutions. To such extent, we corrupt images by applying visual camera failures that are successively fed to TSR classifiers. We simulate 13 visual camera failures, each one with different parameters, obtaining a total of 103 failure configurations to be individually injected into each image of the three datasets. The produced data allows examining the robustness of three DNNs: AlexNet, InceptionV3, and MobileNet-V2. Based on our previous experimental results, we choose DNNs as they performed better compared to non-deep classifier. We apply each DNN independently to perform single image classification, and we also run them in parallel for single image classification; in this second case, classification is performed by a stacking meta-learner which is fed with the outputs of the 2 or 3 DNNs. Furthermore, we also perform sliding window based classification. we consider the following classification strategies based on a sliding window: Sliding window is applied using only one DNN as base classifier, Sliding window is applied using two DNNs, where all images in the sliding window are individually classified by two base-level DNNs and the predictions of each image are input meta-features to the meta-level classifier. Similarly, sliding window is applied using three classifiers where all images in the sliding window are individually classified by three base-level DNNs; the predictions of each image are input meta-features to the meta-level classifier.

Our results indicate that the occurrence of visual camera failures degrades classification performance, especially for single-image classifiers. Instead, approaches based on the sliding

window are significantly more robust. Further, we dig into the results using LIME [92], a toolbox for explainable Artificial Intelligence (AI). Explainable AI allows understanding how a classifier uses the input image to derive its output: we use LIME to explain why the injection of the visual camera failures alters the behaviour of the classifiers, and why certain classifiers are more robust than others against visual camera failures.

1.5 Improving Robustness of Deep Neural Network Image Classifiers against Visual Camera Failures

In this part of the thesis, our focus is on devising different strategies to improve the robustness of image classifiers and tolerate visual camera failures. Examples include, but are not limited to: icing/raindrops on the visual camera lens, broken camera lens, sudden change of brightness, and blurring [85], [89]. We present two different strategies to build DNN image classifiers that are robust against visual camera failures. The first strategy aims at enriching the training set with images altered by injecting the effects of visual camera failures whereas the second strategy employs a Camera Failure Detector (CFD) component to be exercised in conjunction with the DNN. This discussion paves the way for an experimental campaign in which we inject visual camera failures into public datasets that are commonly used for TSR. We train each DNN twice on each dataset:

- *Using a clean training set that contains only images from the original datasets, without injecting the effects of any visual camera failure. We call those regular classifiers.*
- *Using an altered training set that merges the clean training set with altered images in which we inject the effects of visual camera failures. This results in an augmented classifier.*

Then, we use the six models (clean and altered for each of the three DNNs) to perform TSR using different test sets that contain images from the original dataset plus images altered with visual camera failures. Our experimental campaign reveals that augmented classifiers have far better classification accuracy than regular ones when processing both clean images and images altered with visual camera failures. Furthermore, we explore which visual camera failures had a major impact on the classification performance of DNNs, finding that specific configurations of noise and brightness failures have the most negative impact. Also, we explain our results using the LIME framework and we show why augmented classifiers have less misclassification than regular ones. Lastly, we train a binary classifier as CFD component and apply in conjunction with augmented classifier. First image is passed through CFD and based on the output of CFD, we decide that either image will be provided as input to the augmented

classifier, or it will be discarded. Experimental results show that the addition of CFD further improve the performance of TSR, but on the other side it reduces the availability of the TSR system.

1.6 Main Contributions of the Thesis

This section describes the main contribution of the thesis.

- *In the literature, many studies have tried to tackle the problem of TSR by using different features and ML algorithms. However, they mostly exercise a few classifiers or a single feature set on a specific dataset. This does not provide sufficient means to compare different approaches. To such an extent, in chapter 3 of the thesis, we consider 3 traffic sign datasets and a pool of eight non-deep ML algorithm that are exercised independently by feeding them 14 different feature sets and 12 combinations of different feature descriptors for the recognition of traffic signs. we employ also 3 DNNs – namely InceptionV3, MobileNet-V2, and AlexNet.*
- *Regardless of the outcomes of comparison studies, majority of the existing solutions for TSR process a single image and output a classification result. Instead, vehicles gradually approach traffic signs during their road trips, generating sequences of images: the closer the vehicle is to the traffic sign, the better the quality of the image, even under slightly different environmental conditions. Therefore, the problem of TSR naturally scales to knowledge extraction from a set or sequence of images that potentially contain traffic signs. To such extent in chapter 4 of the thesis, we propose a sliding windows-based framework for TSR that processes the sequences of traffic sign images. To the best of our knowledge, there is no study available in the literature which consider sliding windows approach for traffic sign recognition.*
- *The adverse environmental conditions, or the malfunctions of the visual camera, may produce low-quality images that may negatively impact the performance of classifiers. To guarantee safety of the driving task, in chapter 5 we inject a complete set of visual camera failures to study the robustness of TSR systems against the visual camera failures, and to apply different classification strategies to tolerate them.*
- *In chapter 6 of the thesis, we have two contributions to increase the robustness of TSR against camera failures: data augmentation with altered images and addition of camera failure detector. To the best of our knowledge, there is no study that applies data augmentation to tolerate failures of the visual camera. Furthermore, addition of*

camera failure detector component filters out the images that a TSR system may incorrectly classify, that ultimately increase the robustness of TSR.

1.7 Limitations to Validity

We report here possible limitations to the validity and the applicability of our study. These are not to be intended as showstoppers when considering the conclusions of this thesis. Instead, they should be interpreted as boundaries or possible future implications which may impact the validity of this study.

1.7.1 Usage of Public Data

The usage of public image datasets and public tools to run algorithms was a prerequisite of our analysis, to allow reproducibility and to rely on proven-in-use data. However, the heterogeneity of data sources and their potential lack of documentation may limit the understandability of data. In addition, such datasets are not under our control; therefore, possible actions, such as changing the way data is generated, are out of consideration. For example, creating longer sequences of traffic signs or creating a time-sequenced version of the BelgiumTSC is not possible at all.

1.7.2 Parameters of Classifiers

Each classifier relies on its own parameters. Finding the optimal values of parameters is a substantial process that requires sensitive analyses and is directly linked with the scenario in which the classifier is going to be exercised. When applying classifiers to different datasets it is not always possible to precisely tune these parameters: instead, in this study, we perform grid searches, which run a classifier with different parameter values and choose the parameter that maximizes accuracy. This does not guarantee finding the absolute optimum value of a parameter for a given classifier on a given dataset but constitutes a good approximation [84].

1.7.3 Classifier Performance Metric

We are using accuracy as a performance metric that is a quite conservative metric for TSR as it considers all misclassifications at the same level. Instead, we may not be too worried about misclassifying an informative sign with a stop sign, whereas the opposite represents a very dangerous event. Accuracy and all other classifier performance metrics are originally designed for binary classification, while TSR is a multi-class classification problem. Instead of accuracy, other performance metrics can be used to analyse the effectiveness of TSR system.

1.7.4 Injection of Camera Failures to Traffic Signs

The validity of some part of the work also depends on whether the injected malfunctions / distortions of camera failure will fit, what would happen to the image in practice because of actual natural hazards or camera distortions.

2. BASICS AND LITERATURE REVIEW

This chapter provide a brief overview of dependability basics, state-of-the-art approaches for Traffic Sign Recognition (TSR) systems, visual camera failures, and explain briefly about different building blocks of the TSR system.

2.1 *Basics of Dependability*

Dependability of a system is especially important characteristic of a system and must be considered during design phase of the system. Before going to define dependability, we describe briefly different basic notations of dependability defined in [136] such as: System, service, faults, errors, and failures.

2.1.1 *System, Service, Errors, Faults, and Failures*

First, we define the system, boundary, and environment of the system.

Definition 1. *A system is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena. These other systems are the environment of the given system. The system boundary is the common frontier between the system and its environment.*

After defining system, environment and boundary of a system we describe the service delivered by a system. A system can deliver more than one services by implementation more functions.

Definition 2. *The service delivered by a system is its behaviour as it is perceived by its user(s); a user is another system that receives provided service. The part of the provider's system boundary where service delivery takes place is the provider's service interface.*

Definition 3. *Correct service is delivered when the service implements the system function. A service failure, also abbreviated with failure, is an event that occurs when the service delivered deviates from correct service.*

When correct service output is different from the correct output are called service failure modes. Failure modes are ranked based on the severity of failure. Fault and error are defined as follows:

Definition 4. *The part of the system state that is liable to lead to subsequent failure is called an error.*

Definition 5. *The adjudged or hypothesized cause of an error is called a fault.*

An active fault causes an error, otherwise if the fault is not active then it is dormant. Error propagation to the service interface causes service failure that results in a different delivered service than the correct service. All faults are categorized from total eight viewpoints into different fault types. Figure 2 shows eight fault classes that can be divided into three partially overlapped categories such as development faults, physical faults and interaction faults. Development faults occur during the development phase of the system, while physical faults include all those faults that effect the hardware of the system. Finally, all external faults are under the umbrella of interaction faults.

To describe taxonomy of failures there are different ways in which the eccentricity of a failed service reveals itself are the modes of service failure. Each mode can have multiple austerities of service failure that characterize the failed service based on four aspects:

1. the consistency of failures
2. the consequences of failures on environment

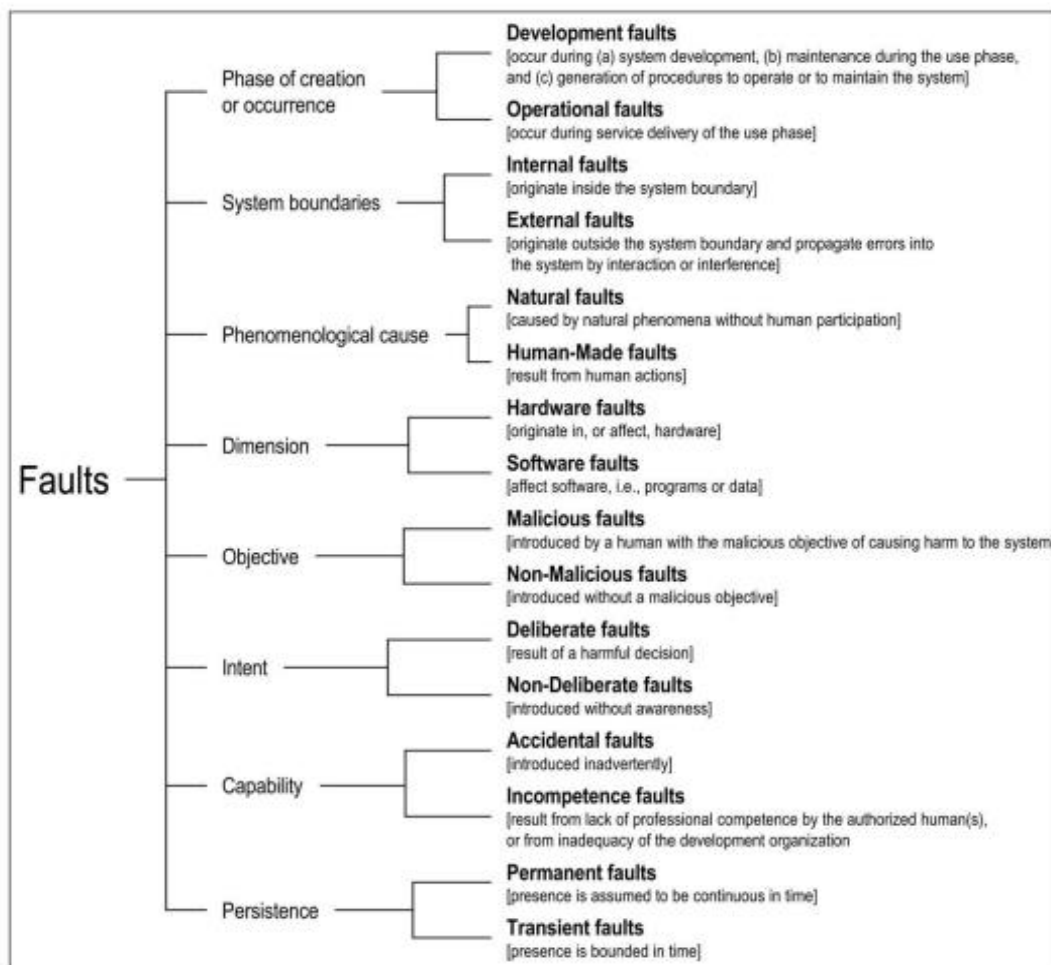


Figure 2: Elementary fault classes from eight viewpoints [136]

3. the failure domain
4. the detectability of failures

Details about these four failure viewpoints can be found in [136]. Two limiting levels can be defined based on the consequences of failure and in the absence of failures the benefit provided by the service delivered. There are two types of failures such as minor failures and catastrophic failures. Catastrophic failures are risky, and they may have more negative effects in order of magnitude than the advantages of correct service delivery. While the costs of repercussions for minor failures are same to the advantages that are offered by correct service delivery.

2.1.2 Dependability and its Characteristics

Dependability can be defined as follows:

Definition 6. *Dependability is the ability of a system to deliver a service that can justifiably be trusted.*

The definition of dependability emphasises the need for trust justification. Dependability has five main characteristics such as availability, reliability, safety, integrity, and maintainability. The first characteristic availability means that the how much the system is ready for correct services, where reliability is concerned with the continuity of service. Safety is concerned to avoid catastrophic consequences on the environment and user, while integrity means the absence of improper alterations. The ability to undergo repairs and modifications is maintainability. There are diverse ways to achieve dependability are as follows:

1. Fault prevention means to avoid occurrence of faults.
2. Fault removal means to reduce the severity and number of faults.
3. Fault tolerance means in the presence of faults avoid the service failures.
4. Fault forecasting means to calculate the present number of faults and predicting the likelihood of future incidents and their consequences.

If we consider camera as a component, the acquired degraded image is considered as failure that may happen due to malfunctioning of the part of camera or due to out-of-distribution samples. Then we are performing the robustness testing, where we provide the faulty images as input to the classifier and the output of the classifier maybe a failure or correct service to the user interface. This thesis focuses on the fault tolerance and fault removal. Through different approaches we are trying to minimize the number of misclassifications of a TSR system and for that purpose quantitative evaluation of different approaches are performed. Furthermore,

we are applying different strategies to avoid service failures of TSR even in the presence of faults.

2.2 Overview of Basic Building Blocks of TSR system

TSR system in most of the cases depends on the feature sets that are provided as input for training and testing of conventional non-deep classifiers, while DNN classifiers do not need explicit feature extraction and meta classifiers (when processing the sliding windows of images) are fed the output of base classifiers as input meta features. We have also utilized LSTM for processing sliding windows of images and it's required the input features. This section provides a brief overview of the main building blocks of the TSR system.

2.2.1 Feature Detectors and Descriptors

Many different Keypoint Detectors and Feature Descriptors were proposed throughout the years. While the detectors aim at identifying interesting keypoints/regions of an image, and the descriptor describes features around the keypoints extracted to be processed by classifiers.

2.2.1.1 Feature detectors

Key points and interest points are the same thing that makes an image intriguing or specific spatial positions or spots within the image. First step is to identify the main keypoints inside an image. Most of the existing studies concerning Keypoint detectors (e.g., [21], [22], [24], [44]) employ KAZE, SURF, MSER, BRISK, FAST, Harris and Minimum Eigenvalue keypoint detectors to identify keypoints in images, we discuss them as follows:

Maximally Stable Extremal Regions (MSER) [121] are defined through the difference in intensity inside and on the outer boundary of a region. First images are grey-scaled, then pixels are converted into black and white according to a specific threshold. Afterwards, black pixels equivalent to local intensity minima will be grouped and progressively merged to build connected components. When two connected components are recursively merged, all pixels of the smaller components are joined into the larger component. Finally, the area of each connected component is stored as a function of intensity, which is used to define thresholds that construct MSER regions. As a result, each MSER is characterized by i) a threshold and ii) the location of a local intensity minima/maxima.

Harris-Stephens [122], FAST [123] and Minimum eigenvalue [124] algorithms are used to detect the corner points. Corners are the locations in an image that have large intensity changes in more than one direction. Moving a window in any direction will result in a large intensity

change. FREAK and HOG descriptors used to extract features around the corners selected by Harris-Stephens algorithm, FAST and Minimum eigenvalue algorithm.

Robust Invariant Scalable Keypoints (BRISK) [125] provides high-quality keypoints with the minimal computational cost. It is a feature detection and description algorithm with rotation and scale invariance. It constructs a scale-space pyramid and in continuous scale space the stable extreme points of sub-pixel precision are extracted. To increase computational efficiency, keypoints are distinguished in octave layers of the image pyramid and in between layers. The local image binary feature descriptor is constructed by using the gray-scale relationship of the random sample point pairs in the image neighbourhood.

Fast Retina Keypoint (FREAK) [126] detects keypoints through a process inspired by the human retina that computes pixel-wise comparison of intensities to devise a sequence of one-bit Difference of Gaussians (DoG). It is similar to BRISK as both extraction processes are circular; on the other hand, BRISK points are equally spaced on the circle while FREAK concentrates keypoints close to the center.

2.2.1.2 Feature descriptors

A technique for extracting the feature descriptions for a keypoint or full image is called a feature descriptor. The ability to distinguish between several characteristics is made possible by the way feature descriptors encode important information in a string of numbers. Our literature review also allowed identifying the most commonly used feature descriptors, which are summarized below.

Speeded Up Robust Features (SURF) [127] is both a keypoint detector and feature descriptor as it first derives features by using integral images, which guarantees low computational complexity. SURF builds rotation and scale invariant descriptors through Hessian matrix approximation, which can be calculated efficiently with respect to other keypoint detectors. Features are then extracted by SURF descriptor by using Haar wavelet filters [128] that calculate the sum of the Haar wavelet responses.

KAZE features [129] describe multiscale 2D features in nonlinear space. Due to nonlinear diffusion filtering, it retains the object boundaries and achieves higher localization accuracy. On the other hand, it is computationally more expensive than features extracted with other descriptors, e.g., SURF.

Histograms of Oriented Gradients (HOG) mostly provide information about each keypoint in images. The process partitions an image into small squared patches and computes HOG for

keypoint in the square. Lastly, the result is normalized and a descriptor for each cell is returned [17].

Local Binary Patterns (LBP) encode the local texture [16] of the image by partitioning each image into non-overlapping cells: the larger the cell, the less accurate the description. This method aims at isolating local binary patterns and is therefore recommended with small gray-scale discrepancies, as its behaviour is invariant to the monotonic transformation of gray-scale.

2.2.2 Deep Features

Two pre-trained DNNs such as AlexNet and ResNet-18 are utilized to extract deep features from traffic sign images are described below.

AlexNet Features (AFeat) are extracted through a pre-trained AlexNet [34], composed of five convolutional layers and three fully connected layers. Convolutional layers are basically extracting deep features from RGB images of size 227×227 . We extract a feature vector of 4096 items by fetching data at the fully connected layer “fc7”.

ResNet Features (RFeat) are extracted from a ResNet-18 [35], a convolutional neural network with 18 hidden layers. Convolutional layers are extracting deep features from RGB images of size 224×224 . Similarly to AlexNet, we extract 512 features by extracting data at the global average pooling layer “pool5”.

2.2.3 Non-deep Classifiers

Non-deep classifiers process features extracted from images as described in Section 2.2.1. Amongst the many alternatives, we summarize below those algorithms that frequently appear in most studies about TSR.

K Nearest Neighbours (KNN) algorithm [13] classifies a data point based on the class of its neighbours, or rather other data points that have a small Euclidean distance with respect to the novel data point. The size k of the neighbourhood has a major impact on classification, and therefore, needs careful tuning, which is mostly achieved through grid or random searches.

Support Vector Machines (SVMs) [14], instead, separate the input space through hyperplanes, whose shape is defined by a kernel. This allows performing either linear or non-linear (e.g., radial basis function RBF kernel) classification. When SVM is used for multi-class classification, the problem is divided into multiple binary classification problems [79].

Decision Tree provides a branching classification of data and is widely used to approximate discrete functions [36]. The split of internal nodes is usually driven by the discriminative power

of features, measured either with Gini or Entropy Gain. Training of decision trees employs a given number of iterations and a final pruning step to limit overfitting.

Boosting (AdaBoostM2) [39] ensembles combine multiple (weak) learners to build a strong learner by weighting the results of individual weak learners. Those are created iteratively by building specialized decision stumps that focus on “hard” areas of input space.

Feed-Forward [130] and Cascaded [131] Neural Networks (FFNN and CNN) are instead inspired by the human brain. As such, they consist of many interlaced layers that are bridged by nodes (neurons) through weighted edges. Initial weights are assigned randomly and adjusted during training until a stop condition or a given number of training epochs is reached. Training may rely on multiple training functions as Gradient Descent algorithms (traingd, traingdm, trainrp, traingcg), Quasi-Newton algorithms (trainbfg,trainlm) and Conjugate Gradient algorithms (trainscg, traingcf) [132].

Linear Discriminant Analysis (LDA) is used to find out the linear combination of features that efficiently separates different classes by distributing samples into the same type of category [38]. This process uses a derivation of Fisher discriminant to fit multi-class problems.

Random Forests [37] build ensembles of Decision Trees, each of them trained with a subset of the training set extracted by random sampling with replacement of examples.

2.2.4 Deep Neural Networks

DNNs may be either built from scratch or more likely—by adapting existing models to a given problem through transfer learning (i.e., knowledge transfer). Through transfer learning, we fine tune the fully connected layers of DNN, letting all convolutional layers remain unchanged. Commonly used DNNs for the classification of images and object recognition are below.

AlexNet [34] is composed of eight layers, i.e., five convolutional layers and three fully connected layers that were previously trained on the ImageNet database [80], which contains images of 227×227 pixels with RGB channels. The output of the last fully connected layer is provided to the SoftMax function, which provides the distribution of overall categories of images.

InceptionV3 is a deep convolutional neural network built by 48 layers that were trained using the ImageNet database [80], which includes images (299×299 with RGB channels) belonging to 1000 categories. InceptionV3 builds on (i) the basic convolutional

block, (ii) the Inception module and finally (iii) the classifier. A 1x1 convolutional kernel is used in the InceptionV3 model to accelerate the training process by decreasing the number of feature channels; further speedup is achieved by partitioning large convolutions into small convolutions [40].

MobileNet-V2 [41] embeds 53 layers trained on ImageNet database [80]. Differently from others, it can be considered a lightweight and efficient deep convolutional neural network with fewer parameters to tune for mobile and embedded computer vision applications. MobileNet-V2 embeds two types of blocks: the residual block and a downsizing block, with three layers each.

Those DNNs can be tailored to TSR through transfer learning. Fully connected layers are trained on defined categories of traffic signs with different learning rates (LR) to fine-tune the models which are already trained on the ImageNet database of 1000 categories.

2.2.5 Stacking Meta-level Classifiers

Stacking meta-learners orchestrate a set of base-learners, which provide meta-data to the meta-level learner. In our study, we foresee the usage of different meta-level learners as listed below.

Majority Voting [42] commits the final decision based on the class the majority of base-learners agree upon. This technique is not very sophisticated, albeit it was and is widely used to manage redundancy in complex systems [81] and to build robust machine learners [82].

Discrete Hidden Markov Model (DHMM) [43]. For each class, a separate Discrete HMM returns the probability of an image belonging to that class. The classification result of the images within the sliding window is given as input to all three DHMMs. Each DHMM returns the likelihood of the sequence to a specific class. The higher the likelihood to a specific class is decided as a final label for that specific sequence.

Non-deep Classifiers in Section 2.2.3. These classifiers can be employed as meta-level learners as meta-data resembles a set of features coming from base-learning episodes.

We are using Non-deep classifiers as meta-level classifier, because the meta level classifiers have the ability the process the input numeric feature vectors and furthermore, computationally its efficient compared to deep classifiers.

2.2.6 Long Short-Term Memory Networks (LSTM)

As an alternative to stacking, we plan the usage of LSTM networks [75], [76]. An LSTM network is a Recurrent Neural Network that learns the long-term dependencies between time steps of sequence data by orchestrating two layers. Those networks do not have a meta-learning structure as a stacker: however, they perfectly fit the analysis of sliding windows of traffic signs as they are intended to be used for the classification of sets or sequences by directly processing multiple images. The first layer contains a sequence of inputs, which are then forwarded to the LSTM fully connected layer, and finally, the output layer shows the classification result.

2.2.7 Traffic Sign Datasets

We conducted extensive research to identify commonly used labeled datasets reporting on sequences of traffic signs with overlapping categories. We selected three public datasets which report on sequences of images of traffic signs, namely: (i) the BelgiumTSC dataset [32], (ii) the GTSRB dataset [31], and (iii) the DITS [33]. We performed the data augmentation with scaling and translation. Details about their structure and the categories of traffic signs are in Table 1 and Table 2, respectively.

2.2.7.1 German traffic signs recognition benchmark dataset

The German Traffic Signs Recognition Benchmark (GTSRB [31]) dataset is widely used in the literature [15], [18], [19], [31] as it reports on images of traffic signs belonging to eight categories with heterogeneous illumination, occlusion and distance from the camera. The dataset contains sequences of 30 images for each traffic sign, which were gathered as the vehicle was approaching it. The authors made available 1307 training and 419 testing sequences of images for a total of 51,780 images contained in the dataset. Table 2 depicts examples of traffic signs for each category of traffic sign contained in this dataset. Importantly, the rectangular traffic signs we mapped into category 8 in the table do not appear in the GTSRB dataset but appear in other datasets considered in this study.

Table 1: Details of the three datasets used in this study

Dataset	Train Images	Test Images	Images per Sequence	Training Sequences	Test Sequences
GTSRB	39210	12570	30	1307	419
DITS	7500	1159	15	500	123
BelgiumTSC	4581	2505	3	1527	835

2.2.7.2 *BelgiumTSC dataset*

The BelgiumTSC dataset [32] is another dataset of traffic signs which was extensively used in the last decade [32], [70]. The BelgiumTSC contains eight categories of traffic signs, shown from category 1 to category 8 in Table 2. The dataset is smaller than the GTSRB: the BelgiumTSC contains only 2362 sets of three images taken with different cameras from different viewpoints. It follows that this dataset reports triple images for each traffic sign which are all taken at the same time and thus are not time-ordered.

2.2.7.3 *Dataset of Italian traffic signs dataset*

The Dataset of Italian Traffic Signs (DITS) dataset is considered more challenging than others in the literature [33] as it contains traffic signs images that were taken under non-optimal lighting conditions, e.g., day, night-time, foggy weather. The DITS contains 623 sequences containing a varying, time-ordered, number of frames. We point out that DITS is the only dataset in this study that contains all the nine categories of traffic signs reported in Table 2 and as such, it provides a complete view of all potential traffic signs. The dataset contains 500 training sequences and 123 testing sequences of varying lengths as summarized in Table 1.

2.2.8 *Classification Performance Metric*










The performance of classifiers for TSR is usually compared by means of classification metrics. These metrics are mostly designed for binary classification problems, but they can be adapted also to measure multi-class classification performance. Correct classifications i.e., True Positive (TP) and True Negative (TN) reside in the diagonal of the confusion matrix, whereas any other item such as True Negative (TN) and False Negative (FN) of the confusion matrix is counted as a misclassification.

2.2.8.1 *Accuracy*

Amongst the many alternatives, TSR mostly relies on accuracy [77], [78], which measures the overall correct and incorrect classifications. Accuracy can be defined as:

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

Table 2: Categorization of Traffic Signs into 9 categories based on their shape, color, and content.

Category	1	2	3	4	5	6	7	8	9
Traffic Signs									
GTSRB								✗	✓
BelgiumTSC	✓	✓	✓	✓	✓	✓	✓	✓	✗
DITS								✓	✓

It should be noticed that this is a quite conservative metric for TSR as it considers all misclassifications at the same level. Instead, we may not be too worried about misclassifying an informative sign (e.g., Category 8 in Table 2) with a stop sign, whereas the opposite represents a very dangerous event. That being said, for ease of comparison with existing studies, we calculate accuracy according to its formulation, thus considering each misclassification as equally harmful.

2.2.8.2 *Specificity*

It also called true negative rate, can be defined as a ratio of the true negatives to all negatives. The purpose behind specificity is to reduce the false alarms. Higher specificity means low false positives and vice versa. Based on confusion matrix, we can define specificity as:

$$Specificity = \frac{TN}{TN+FP} \quad (2)$$

2.2.8.3 *Recall*

Recall is also known as true positive rate or sensitivity. It can be defined as the ratio of true positives to total positives with the aim to reduce false negatives. In the TSR system the FN are very crucial and intolerable, e.g., classifying a stop sign as a speed limit sign that will result in catastrophic event. Mathematically it can be defined as:

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

2.2.8.4 *Precision*

It can be defined as the ratio of true positives to the total positives predicted. Higher number of false positives will result in lower precision and vice versa. We can define precision as:

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

2.2.8.5 *F1-Measure*

F1-measure can be defined as the harmonic mean of precision and recall. Based on confusion matrix, we can define F1-measure as follows:

$$F1 - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5)$$

2.2.8.6 *F-AA*

We are calculating F-AA for TSR system inspired by the formulation of F1-measure between precision and recall. In formulation of F-AA precision and recall are replaced by the

accuracy and availability of the TSR system, where we are giving equal importance to accuracy and availability of the TSR system. F-AA can be defined as:

$$F - AA = 2 \times \frac{\text{Accuracy} \times \text{Availability}}{\text{Accuracy} + \text{Availability}} \quad (6)$$

2.3 Related Works on Different TSR Approaches

In the last decade, researchers, practitioners, and companies devised automatic TSR systems to be integrated into Advance Driving Assistance System (ADAS). Amongst all the possible approaches, most TSR systems rely on the same main blocks, namely: (i) Dataset creation/identification, (ii) pre-processing (e.g., resizing, histogram equalization), (iii) Feature extraction and non-deep model learning, or (iv) model learning through Deep Neural Networks (DNNs) i.e., deep learners.

Feature extractors and non-deep classifiers have been arranged differently to minimize misclassifications in a wide variety of domains. Soni et al. [24] processed the Chinese traffic sign dataset through SVM, trained on the HOG or LBP after Principal Component Analysis (PCA), reaching an accuracy of 84.44%. A similar setup was used by Manisha and Liyanage [21], who achieved 98.6% accuracy on vehicles moving at 40–45 km/h. Moreover, Matoš et al. [22] used an SVM trained on HOG features and achieved recognition of 93.75% accuracy on the GTSRB dataset. The same dataset was used in [44], where Extreme Learning Machine (ELM) improved accuracy to 96%. Agrawal and Chaurasiya [45] extracted HOG features from the traffic signs of the GTSRB dataset and applied PCA for the dimensionality reduction to obtain an accuracy of 73.99%, 66.46%, 91.86% on denial, mandatory and danger traffic sign categories. Similar studies as [15], [46], [47] processed the same datasets with different feature sets and algorithms, obtaining similar scores.

DNNs and the Viola–Jones framework allowed the authors of [8] to enhance classification on the GTSRB dataset with up to 90% of accuracy. Li et al. [28] proposed a new Convolutional Neural Network and trained on the GTSRB and BelgiumTSC datasets. The proposed architecture achieved an accuracy of 98.1% and 97.4% on BelgiumTSC and GTSRB datasets, respectively. In [48], the fifteen-layer WAF-LeNet network reached a detection accuracy of 96.5% on GTSRB. The authors of [49] proposed an approach for Traffic Sign Detection and Recognition (TSDR) using SegU-Net and a modified Tversky loss function With L1-Constraint that achieved 94.60% and 80.21% precision and recall, respectively, on the CURE-TSD dataset. Liu et al. [50] proposed traffic sign recognition and detection approaches, which

first extract the region of interest and after verification of the traffic sign through an SVM classifier, it classifies the traffic sign into traffic sign categories. The proposed approach achieves the highest accuracy 94.81%.

Another study [51] used the InceptionV3 model trained with transfer learning on the BelgiumTSC dataset, obtaining an accuracy of 99.18%. In [52], the authors found that the Tiny-YOLOv2 network is fast but outperformed by YOLOv2 or YOLOv3 DNNs. While the authors of [53] introduced real time image enhancement CNN and achieved an accuracy of 99.25% for the BelgiumTSC, 99.75% for GTSRB, and 99.55% for Croatian Traffic Sign (rMASTIF). Authors of [54], developed a real time TSR by using the You Only Look Once (YOLO) algorithm to train the model for Malaysian traffic sign recognition and tested it on five types of warning traffic signs. In [55] authors propose a lightweight CNN architecture for the recognition of the traffic sign GTSRB dataset, and they achieved 99.15% accuracy. In one another study [56], a novel semi supervised classification technique is adopted for TSR with weakly-supervised learning and self-training. An ensemble of CNN was used for the recognition of the traffic signs and achieved higher than 99% accuracy for the circular traffic signs of the German and BelgiumTSC datasets [57]. Lu et al. [58] use multi-modal tree structure embedded multitask learning for the GTSRB dataset and achieved an overall accuracy of 98.27%. In [59], the authors improved the VGG-16 DNN by removing some redundant convolutional layers and adding Batch Normalization and global average pooling layer to improve the performance of the network, while [60] proposed a hybrid 2D-3D CNN. In [61], the authors proposed a traffic sign recognition system that learns learning hierarchical features based on multi-scale CNNs. In one another study [62], the authors proposed a real-time TSDR for Chinese and German roads. In [63] authors proposed a robust custom feature extraction method and multilayer artificial neural network for the recognition of traffic signs in real time.

Only a few comparative studies have been proposed in the literature. For example, Jo [15] trained different non-deep classifiers on HOG features extracted from the GTSRB dataset. Similarly, Schuszter [70] reported on experiments with the BelgiumTSC dataset [32], where HOG features were extracted from images and then fed to the SVM to classify one of the six basic traffic sign subclasses. Yang et al. [19] provide a comparison of different classifiers, such as the *KNN*, SVM, Random Forest and AdaBoost trained by using combinations of features. This study reported the highest accuracy by using Random Forest with the combination of LBP and HOG features. Another study [29] compared non-deep classifiers and DNNs on three datasets, i.e., GTSRB, BelgiumTSC and DITS considering three broad categories of traffic

signs, i.e., red circular, blue circular and red triangular. Noticeably, both non-deep classifiers and DNNs achieved perfect accuracy on GTSRB. Moreover, the authors of [18] trained different classifiers for traffic sign recognition. They considered the GTSRB dataset and extracted HOG features to train LDA and Random Forest. Additionally, they used the committee of Convolutional Neural Networks (CNN) and multiscale-scale CNN. While in the study [31] authors organized a competition to classify GTSRB dataset traffic signs. These traffic signs were categorized by human and ML algorithms and an accuracy of 98.98% was achieved which is comparable to human performance on this dataset. Therefore there is need to perform systematic comparative analysis of different non-deep machine learning techniques, where different feature sets are provided as an input and DNN for TSR to benchmark these different approaches on heterogeneous datasets.

Furthermore, we know that as vehicle is approaching to a traffic sign during this time input visual camera can capture multiple images that can be processed to commit the final decision about a traffic sign. In this context there are only a few works that perform classification depending considering multiple images. In a study [64], authors considered the sequences of images of the street view and synthetic images and they achieved an 87.03% evaluation score, i.e., the ratio of true positive and true positive + false positive + false negative. In another study, Yuan et al. [65] proposed a video-based traffic sign detection and recognition mechanism to fuse the result of all images for final classification. They utilized a multi-class SVM with two different fusion strategies, i.e., equal weighting and a scale-based weighting scheme which achieved 99.48% accuracy on the MASTIF dataset. In the literature, there are many studies [66], [67], [68], [69] focusing on single image TSR, and very few studies [64], [65] that process multiple images. The difference with our work is that they fuse the result of all frames for final decision that means the system will wait till the last frame, while in our case sliding window-based method will start giving decision as we have the minimum number of traffic sign images available i.e., 2 and 3 according to the sliding window size. Furthermore, they are not achieving the perfect classification on the data sets they are using for the experiments. According to our knowledge based on the literature review, there is no study available that considers the sliding windows approach for traffic sign recognition.

Automatic TSR systems embed Machine Learning (ML), and especially DNN classifiers which process image images captured by visual cameras installed on the vehicle. Those TSR systems are known to accurately classify traffic signs, even reaching perfect classification performance under nominal operating conditions [3], [29], [87]. Unfortunately, the adverse

environmental conditions, or the malfunctions of the visual camera, may produce low-quality images that may negatively impact the performance of classifiers. Autonomous vehicles contain visual cameras that observe the environment and capture sequences of images. Several studies in the literature utilize a visual camera for autonomous driving tasks, such as obstacle detection, pedestrian detection and lane detection, and obviously traffic sign recognition [60], [86], [115], [98]. If the visual camera has a malfunction, the quality of the captured images is degraded, and this may result in a critical situation and possible serious consequences. The possible occurrence of visual camera failures must be considered to guarantee the safe operation of autonomous vehicles [88]. These failures may occur in any component of the visual camera, and especially they may be malfunctions of the lens, the image sensor, or the ISP (Image Signal Processor): all these components cooperate to create every image produced by the visual camera. Only few works focused specifically on the effect that visual camera failures may have on the produced images. In [89], the authors adopted a CNN to deal with different harsh conditions such as out-of-focus, illumination, and missing information applied on the GTSRB dataset. Authors apply an attention mechanism to build a convolutional pooling for performance improvement. In [88], authors systematically define different failure modes of a vehicular visual camera, and they discuss the visible effects on the captured images. Also, they propose a Python library to inject failures into images. Lastly, Morozov et al. [120], while not strictly considering the image acquisition, simulate hardware failures: they used three CNNs for classification and a Bayesian Network for the analysis of the trustworthiness of results. There is a need to consider failures that happens due to visual camera that will result in a degraded quality image and to observe the performance of the classifier on the degraded images and ways to improve the robustness of classifiers. To such extent, in this thesis, we use the failure categorization of [88] as reference, because we believe it is the most complete representation of visual camera failures available in the state of the art, and it is supported by a software library that allows reproducing the failure effects on target images.

2.4 Background on Visual Camera Failures

Misbehavior(s) of the visual camera may generate altered images that are delivered to the image classifier. To explain visual camera failures, we present the different components of a visual camera [137], also represented in Figure 3. The lens senses the scene from the environment, in the form of light. This light is processed by the Image Sensor, whose photodiodes transform light in its electrical encoding producing a raw file [138]. The Bayer Filter, which acts on top of the Image Sensor, colors of the color-blind photodiodes into the

red-green-blue (RGB) pattern. The Image Signal Processor (ISP) processes the raw file to produce the digital image; its functions are multiple, to name the most famous: demosaicing, noise reduction, image sharpness correction, lens distortion correction, no chromatic aberration correction, image compression, and JPEG encoding [137].

While many works like [139] elaborate on the effect of modified images on the classification process, only a few studies in the literature focus on the effects that failures of the visual camera may have on the produced image and consequently on the image classification. Even if the risk of accidental alterations of the output image of the visual camera is acknowledged as realistic [140], this consideration is usually ancillary to the main contribution of the work. Examples are [89], where the authors focus on environmental conditions and build a DNN that implements an attention mechanism for performance improvement, or [141], where driving situations analyze how sensors respond when used in real circumstances as well as to confirm the impacts of environmental conditions.

2.5 Related Works on Robust Image Classifiers

We define the robustness of a DNN image classifier as sustaining the performance of the model under various image corruptions or alterations [142]. In our study, image corruptions and alterations are due to visual camera failures and lead to the effects discussed in Section 2.4 and Section 5.2.4. We now review robustness approaches, organized in architectural solutions for building robust DNNs, out-of-distribution detection, and data augmentation strategies for robustness and adversarial defense, and we present differences with respect to our work.

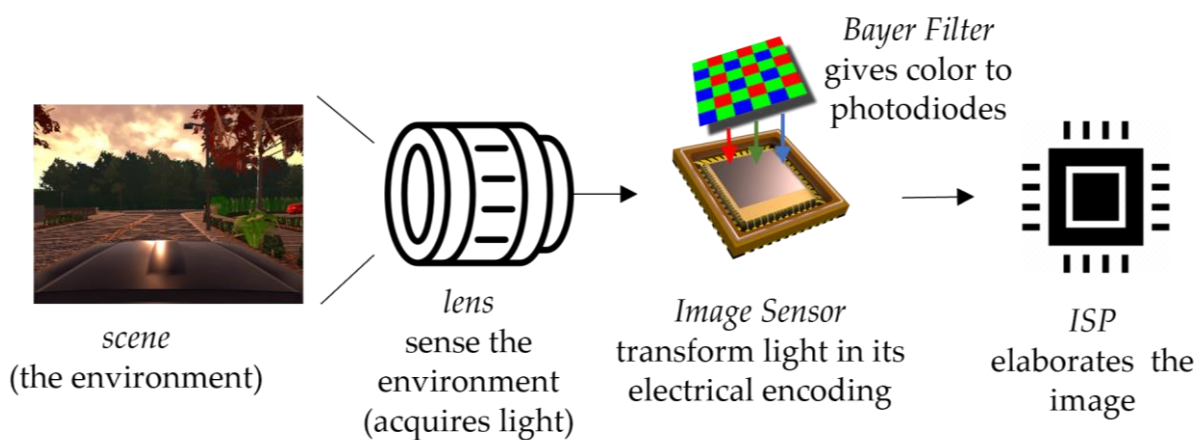


Figure 3: A visual camera and its components: the Lens, and the camera body composed of Bayer Filter, Image Sensor, and Image Signal Processor [88].

2.5.1 Architectural Solutions for Robust DNNs

There have been several recent proposals of robust architectures for DNN image classifiers. In [116], the authors analyzed the classification performance of several DNNs against images degraded by Gaussian noise, blur, and compression. They end up proposing a two-step (master-slave) process in which the master classifies the quality of the degraded input image, which is then used to select the most suitable slave DNN for classification. In [116], the authors provide a review of methods that combine two or more images and pre-process them to delete specific regions of the images that may make the DNN lean towards misclassifications. Experimental results show that this method builds image classifiers that are more robust and have better classification performance even when dealing with altered images. In [144], the authors estimated the confidence of the DNN in the response to unexpected execution contexts.

2.5.2 Out-of-Distribution Detection

Many works like [145], and [146] set the goal of detecting out-of-distribution samples, which can be detected without knowledge at training time of the kind of image alteration. Especially, [145] trains four different DNNs with three different supervisors at various stages of training, to detect at what point during training the performance of the supervisors starts to decline. In our thesis, the failures that degrade the acquired image quality such as Rain, Ice and dirt etc., are the examples of out-of-distribution samples.

2.5.3 Data Augmentation for Robustness

Another group of works is composed of studies that aim for DNN robustness through data augmentation strategies [147]. Authors of [117] use data augmentation to improve the generalization capability of DNNs using smoothness regularization against perturbations to improve the classification performance. In another study [118], authors employ a Pixel Mask to diminish the sensitivity of DNNs against the corruption of images. Moreover, the study [119] proposes a data augmentation pipeline to accelerate MRI reconstruction. Instead, our study is focused on cyber-physical systems that have a visual camera: to the best of our knowledge, there is no study that applies data augmentation to tolerate failures of the visual camera. To address this issue, we consider a complete set of failures that may happen due to malfunctions of the visual camera.

2.5.4 Data Augmentation for Adversarial Defense

Many recent works on data augmentation set the goal of adversarial defenses, i.e., defending from images explicitly designed by an attacker to fool a classifier. Briefly, these

works propose adversarial training i.e., they increase the robustness of the target classifier, which is trained using genuine and adversarial images [148]. Even if the approach is conceptually similar to ours, our study does not consider the adversarial activity as one of the potential sources of altered images, therefore those are only marginally related works and are interesting to our study only regarding the approach they follow.

3. COMPARATIVE ANALYSIS OF NON-DEEP AND DEEP NEURAL NETWORK CLASSIFIERS FOR TRAFFIC SIGN RECOGNITION SYSTEM

This chapter of the thesis provides a comparison of Machine Learning (ML) and Deep Neural Networks (DNNs) for Traffic Sign Recognition (TSR) systems. Many studies have tried to tackle the problem of TSR [21], [22], [98], by using different features and ML algorithms. However, they mostly exercise a few classifiers or a single feature set on a specific dataset. This does not provide sufficient means to compare different approaches. To such an extent, in this chapter of the thesis, we consider 3 datasets and a pool of eight non-deep ML algorithms for the recognition of traffic signs. These algorithms are exercised independently by feeding them 14 different feature sets and 12 combinations of different feature descriptors: this allows also debating on the efficacy and information content of such feature sets. To complete the pool of algorithms, we employ also 3 DNNs – namely InceptionV3 [40], MobileNet-V2 [41], and AlexNet [34] - that do not need feature sets as they automatically extract features using convolutional layers while training. Altogether, these experiments build a quantitative comparison of different ML algorithms to benchmark different approaches by using heterogeneous datasets that provides a baseline for practitioners and researchers who are willing to set up a TSR component to be embedded in autonomous vehicles. Results allow elaborating on the performance of non-deep classifiers trained on different feature sets, a fusion of different feature descriptors, and DNNs that are trained using transfer learning.

3.1 Motivation behind Comparative Analysis

Indeed, only a few comparative studies have been proposed in the literature. For example, Jo, K. [15] trained different non-deep classifiers on Histogram of Oriented Gradients (HOG) features extracted from the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Similarly, Schusztter, C. [70] reported on experiments with the BelgiumTSC dataset [32], where HOG features were extracted from images and then used to train Support Vector Machine (SVM) and deep Convolutional Neural networks, obtaining a maximum of 95.83% accuracy. Yang et al., [19] provide a comparison of different classifiers like K-Nearest Neighbors (*KNN*), SVM, Random Forest, and AdaBoost trained by using combinations of features. This study reported the highest average accuracy of 97.39 % achieved using Random Forest with the

combination of Local Binary Patterns (LBP) and HOG features considering GTSRB and Swedish Traffic Signs Dataset.

Consequently, it was not possible to find a comprehensive comparison of different approaches, as different authors followed heterogeneous experimental methodologies. To fill this gap, in this thesis we (i) identify 3 public datasets reporting on images of traffic signs, (ii) define a pool of 14 combinations of keypoint detectors and feature descriptors and 12 combinations of different feature descriptors, (iii) select 8 non-deep and 3 DNNs, (iv) adopt the most used scoring metrics to evaluate the quality of classifiers, and finally (v) present and discuss our results. In summarizing, we strongly believe that the comparative analysis of this thesis will provide researchers and practitioners with concrete means to set up their TSR systems through a fair comparison of existing techniques by means of state-of-the-art metrics.

3.2 A Fair Comparison of TSR Systems

To prepare our experimental comparison, we started looking for datasets (Section 3.1), then we gathered implementations for 14 couples of keypoint detectors and feature descriptors and a combination of 12 feature descriptors to be used in conjunction with non-deep classifiers (Section 3.2.2 and Section 3.2.3). DNNs in Section 3.2.4, instead, embed representation learning and therefore do not need to be fed with features. Finally, Section 3.3 describes the experimental setup.

3.2.1 TSR Datasets and Traffic Sign Categories

We conducted extensive research to identify commonly used labelled datasets reporting on traffic signs with overlapping categories, e.g., danger signs are red circular for all datasets. Datasets used in this chapter of thesis include i) the GTSRB dataset [31], ii) the BelgiumTSC dataset [32], and iii) the Dataset of Italian Traffic Signs (DITS) [33] that are described in Section 2.2.7. To be able to compare the result of different datasets, we consider only the three categories of traffic signs that appear in all datasets, namely red circular, blue circular, and red triangular.

3.2.2 Keypoint Detectors and Feature Descriptors

To identify key points and describe features to be extracted from images we considered the seven key point detectors KAZE, SURF, MSER, BRISK, FAST, Harris, and Minimum Eigenvalue algorithm described in Section 0, and four key point descriptors LBP, HOG, SURF, and KAZE (Section 2.2.1.2). A group of connected pixels in an image that have some common properties such as brightness, colour etc., are called blobs, and the goal of blob detector is to

Table 3: Combination of key point detectors and descriptors

Keypoint Detector	Detector Category	Feature Descriptor
FAST	Corner based	HOG
FAST	Corner based	FREAK
Harris	Corner based	HOG
Harris	Corner based	FREAK
Minimum Eigenvalue	Corner based	HOG
Minimum Eigenvalue	Corner based	FREAK
Brisk	multi-scale corner features	HOG
Brisk	multi-scale corner features	FREAK
KAZE	Blob based	SURF-like
KAZE	Blob based	HOG
SURF	Blob based	SURF
MSER regions	Blob based	SURF
Full image	-	LBP
Full image	-	HOG

detect and mark those regions, whereas corner detector aims to detect the corner features that are described by feature descriptors. Table 3 highlights the 14 combinations of key point detectors and descriptors used in this study. In addition, to complete this analysis, we created combinations of either 2 or 3 feature descriptors that are fed simultaneously to classifiers, namely HOG + KAZE, HOG + LBP, HOG + LBP + SURF, HOG + MSER, HOG + SURF, LBP + KAZE, LBP + MSER, LBP + SURF + MSER, MSER + KAZE, SURF + KAZE, SURF + LBP, and SURF + MSER.

3.2.3 Non-deep Classifiers and their Parameters

Our analysis includes all the non-deep classifiers we listed in Section 2.2.3, namely *KNN*, *SVM*, *Decision Tree*, *Boosting (ADABOOSTM2)*, *Random Forests*, *Linear Discriminant Analysis*, *Feed Forward*, and *Cascaded Neural Network*. Each algorithm requires its own parameter setup: therefore, we repeatedly executed each of them with slightly different values of parameters to find the most beneficial parameter setup through grid search processes. To such an extent, we employed the following parameter setups.

- *KNN* with different values of k , i.e., different odd values of k from 1 to 15, namely $k \in \{1, 3, 5, 7, 9, 11, 13, 15\}$.

- *SVM: three different kernels Linear, Radial Basis Function (RBF), and Polynomial (quadratic) was used to perform linear and nonlinear classification.*
- *Decision Tree: we used the default configuration of MATLAB with $n - 1$ for `MaxNumSplits`, where n is the training sample size.*
- *Boosting: AdaBoostM2 algorithm with 100 trees.*
- *Random Forest with 100 trees.*
- *Linear Discriminant Analysis (LDA): we used pseudo linear discriminant type.*
- *Feedforward Neural Networks (FFNN) and Cascaded Neural Networks (CNN) were set with either one or two hidden layers each with 10 neurons. Moreover, we employed 9 different neural network training functions `traingd`, `traingdm`, `trainrp`, `traingcg`, `trainbfg`, `trainlm`, `trainscg`, `traingcf`.*

3.2.4 Deep Neural Networks for TSR

Three DNNs of Section 3.2.4, i.e., InceptionV3, MobileNet-V2 and AlexNet are trained using transfer learning. Fully connected layers are trained on 3 categories of traffic signs with different learning rates (LR) to fine-tune the models which are already trained on ImageNet database of 1000 categories. We varied learning rate as follows: {0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001} for InceptionV3 / MobileNet-V2, and {0.00009, 0.00005, 0.00001, 0.000009, 0.000005, 0.000001} for AlexNet. Instead, we always used 10 as the minimum batch size, with 10 train epochs for all the experiments on each dataset to fine-tune the models for TSR. The process of transfer learning relied on data augmentation to avoid overfitting; this was conducted through X and Y translation with a random value between [-30 30] and a scale range within a range [0.9 1.1].

3.3 Experimental Setup

We orchestrate the inputs above as follows. Images go through a pre-processing phase to enhance the contrast between background and foreground through histogram equalization. Then, each pre-processed image is analyzed to extract 14 feature sets (Table 3) plus additional 12 combinations to be used with non-deep classifiers, while DNNs are directly fed with pre-processed images. All classifiers, both non-deep and DNNs are trained independently with their own parameter setup. Starting from the left of Figure 4, these three building blocks interact with each other sequentially. Each image in the dataset is pre-processed and then analysed to identify key points to allow feature extraction. These features are then fed to the classifier, either for training or for testing (right of Figure 4), if the training model was already learned.

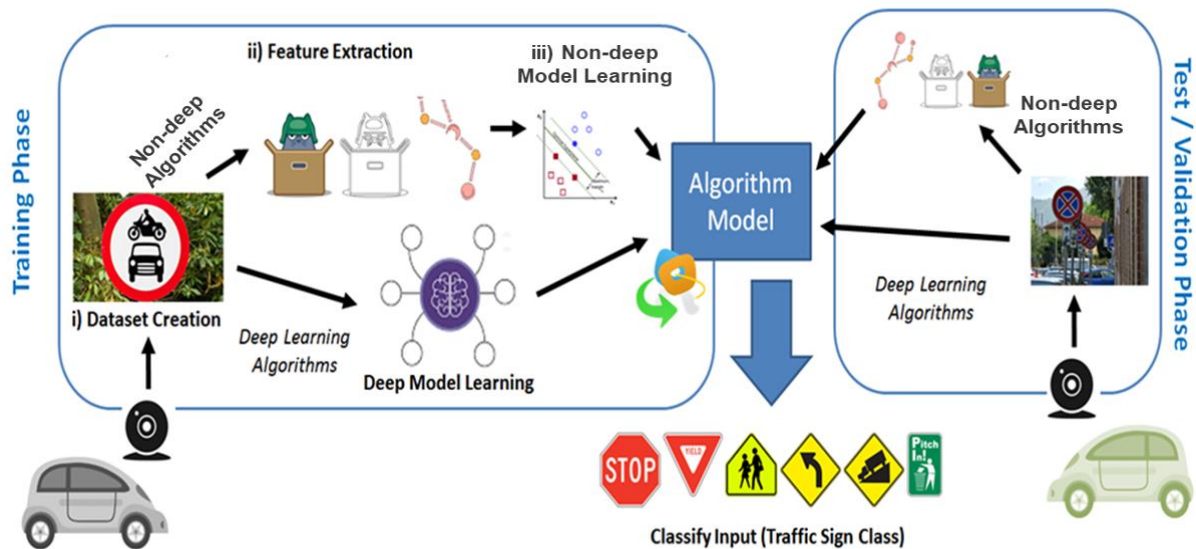


Figure 4: Block Diagram of Traffic Sign Recognition System

An alternative is shown in the lower part of Figure 4, DNNs, which, unlike non-deep classifiers, combine steps ii) and iii). Briefly, DNNs are consisting of multiple convolutional layers that allows to learn representations of data. Although DNNs are very resource and time-consuming, they show promising classification capabilities and are being currently used in TSR.

Classification performance is scored through a confusion matrix (True Positives TP, True Negatives TN, False Positives FP, False Negatives FN) and through metrics [99] that build on them, such as Accuracy, Precision, and Recall defined in Section 2.2.8. It is worth mentioning that these performance metrics are designed for binary classification, while we deal with 3 categories of traffic signs. We adapt our categories to the confusion matrix by considering that the traffic signs that identify an “immediate danger” are the red circular ones, we consider the following

- TP if the traffic sign was red circular and was classified as such
- TN if the traffic sign was red triangular and was classified as such or if it was blue circular and classified as such.
- FN if the traffic sign was red circular and was misclassified
- FP if the traffic sign was either red triangular or blue circular and was misclassified.

The experiments were conducted on an Intel(R) Core (TM) i5-8350U CPU@1.7GHz 1.9 GHz running MATLAB. MATLAB implementations of DNNs also use our NVIDIA Quadro RTX 5000 GPU. All experiments required approximately 6 weeks of execution.

3.4 Experimental Results

This section of the thesis provides the detailed experimental results achieved using non-deep machine learning and DNNs.

3.4.1 Results of Non-deep Classifiers

This section reports and discusses the results of our experimental campaign that *explicitly focus on non-deep classifiers*. We also expand on the impact of kernels for SVMs and training functions for Neural networks, which are known to be heavily impacted by them.

3.4.1.1 On intrinsic difficulty of datasets

Before analysing the metric scores of individual classifiers, we try to understand the intrinsic difficulty of our datasets, as traffic signs contained in one or more datasets could be easier to classify than others. To such an extent, we calculated the average (and standard deviation) values that all classifiers achieve on each individual dataset by using optimal parameters setup. Accuracy, Specificity (True Negative Rate), and Recall values in Table 4 are on average higher for the GTSRB dataset, which can be therefore considered the easiest to classify out of the three. More in detail, *KNN*, Random Forest, and SVM achieve 100% accuracy on the GTSRB dataset, while other classifiers show accuracy above 98.88%. These perfect scores may be due to the size of the training set, which is the largest out of the three for GTSRB; in addition, the quality of the images is better with respect to DITS and BelgiumTSC. On the other hand, metric values are on average lower when dealing with DITS. We explain this result as DITS contains a variety of images taken at different lighting conditions (i.e., day and night time, even with foggy weather), which heavily affect the capabilities of classifiers that end up being error-prone.

3.4.1.2 Highest accuracy for each dataset

Figure 5 depicts a chart with bars reporting the highest accuracy achieved by classifiers in each of the 3 datasets. It is clear from the blue solid bars in the figure that almost all classifiers give better performance on the GTSRB dataset compared to the other 2 datasets, which matches the average results we obtained in the previous Section 3.4.1.1. While as we said above *KNN*, SVM, and Random Forest give 100% accuracy on GTSRB, in the BelgiumTSC dataset the highest accuracies are provided by *KNN and AdaBoostM2* (97.58%). Instead, *AdaBoostM2*

Table 4: Average \pm standard deviation of accuracy, specificity and recall of 8 non-deep classifiers on 3 datasets.

Datasets	Accuracy	Specificity	Recall
BelgiumTSC	93.53 \pm 2.97	91.10 \pm 2.67	97.42 \pm 3.77
DITS	89.74 \pm 3.13	88.90 \pm 2.79	90.96 \pm 4.31
GTSRB	99.49 \pm 0.45	99.40 \pm 0.51	99.58 \pm 0.40

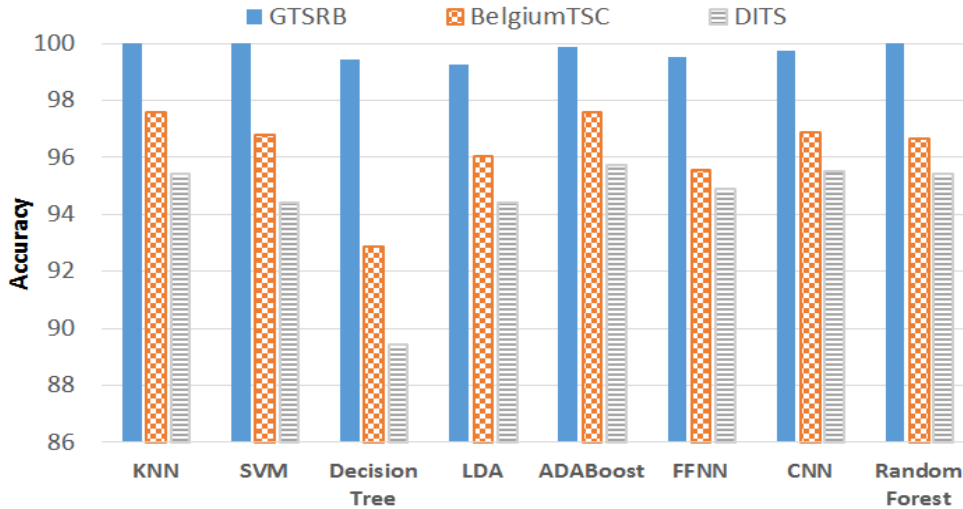


Figure 5: Highest Accuracy achieved by Non-deep Classifiers

provides the highest accuracy of 95.70% in DITS, with *KNN* and *SVM* that come close at 95.50%. Depending on those results, we observe how *KNN* almost always achieves the highest accuracy in all the three datasets, while *SVM* and Boosting are still good – albeit non-optimal – choices overall.

3.4.1.3 Impact of keypoint detectors and feature descriptors

The results discussed in the previous Section 3.4.1.2 are strongly linked to the features that are fed to algorithms. To carefully expand on this aspect, Table 5 highlights the highest Accuracy achieved by classifiers on each dataset by using a given keypoint detector and feature descriptor to extract features. Corner detection with FREAK features (see rows 2, 4, 6, 8 in Table 5) outputs lower classification accuracy with respect to corner detectors with HOG features (as in rows 1, 3, 5, 7), which allow achieving the highest accuracies albeit they are costly to compute. LBP, HOG, and MSER with SURF descriptor reach 100% accuracy (see rows 12, 13, 14 of the table) on GTSRB data, while the accuracy of LBP and MSER with SURF descriptor degrades on DITS. Instead, HOG features extracted from the full image allow reaching high accuracy scores also on BelgiumTSC and DITS, being the best choice of a single feature descriptor in all the three datasets. To complete the discussion about the relevance of feature descriptors, we focus on the grayed rows in Table 5 which repeat the analysis above using more than one feature descriptor at a time. Overall, we observe how providing more features to classifiers through combinations of feature descriptors allows improving maximum accuracy in DITS (from 93.61 to 95.71) and BelgiumTSC (from 96.52 to 97.58). More in detail, combinations such as HOG + LBP and HOG + LPB + SURF achieve the highest accuracy in the BelgiumTSC dataset and top-notch values in DITS and GTRSB.

Table 5: Highest Accuracy achieved using different keypoint detectors and feature extractors on each dataset. Grayed lines in the bottom of the table highlight combinations of items above.

Keypoint Detector (Feature Descriptor)	Highest Accuracy (%)		
	GTSRB	DITS	BelgiumTSC
FAST(HOG)	93.93	73.95	79.85
FAST(FREAK)	61.71	67.16	59.52
Harris (HOG)	97.10	69.46	77.84
Harris (FREAK)	66.32	77.64	67.38
MinEig. (HOG)	97.55	69.86	75.51
MinEig. (FREAK)	67.74	79.04	67.01
Brisk (HOG)	95.99	82.23	87.57
Brisk (FREAK)	65.95	69.36	60.8
KAZE(SURF)	99.98	80.93	93.19
KAZE(HOG)	98.32	82.23	90.36
SURF (SURF)	79.62	91.61	89.21
MSER Regions (SURF)	100.00	84.93	91.95
LBP	100.00	83.63	93.51
HOG	100.00	93.61	96.52
HOG + KAZE (SURF)	100.00	95.61	96.53
HOG + LBP	100.00	95.51	97.58
HOG + LBP + SURF (SURF)	99.99	95.71	97.58
HOG + MSER (SURF)	100.00	95.31	96.98
HOG + SURF (SURF)	100.00	95.41	96.53
LBP + KAZE (SURF)	99.92	84.43	95.25
LBP + MSER (SURF)	99.86	87.62	93.56
LBP + SURF (SURF) + MSER (SURF)	99.92	90.62	94.75
MSER (SURF) + KAZE (SURF)	100.00	86.53	94.75
SURF (SURF) + KAZE (SURF)	100.00	89.72	94.15
SURF (SURF) + LBP	99.82	90.22	94.20
SURF (SURF) + MSER (SURF)	100.00	90.52	93.56

3.4.1.4 NN: contribution of training functions

Feedforward and cascaded neural network performance strongly depends on the training function that is used to learn the model. Throughout our study, we used 9 different training functions, which were applied independently on each dataset for the training of FFNN and CNN. Figure 6 highlights the average (and standard deviation) scores of Accuracy, Specificity and Recall that we obtained by averaging metrics achieved by the best algorithm using a given train function across the 3 datasets. Train functions are sorted from left to right according to a decreasing average accuracy score: trainlm on the left achieved the highest, while traingdx

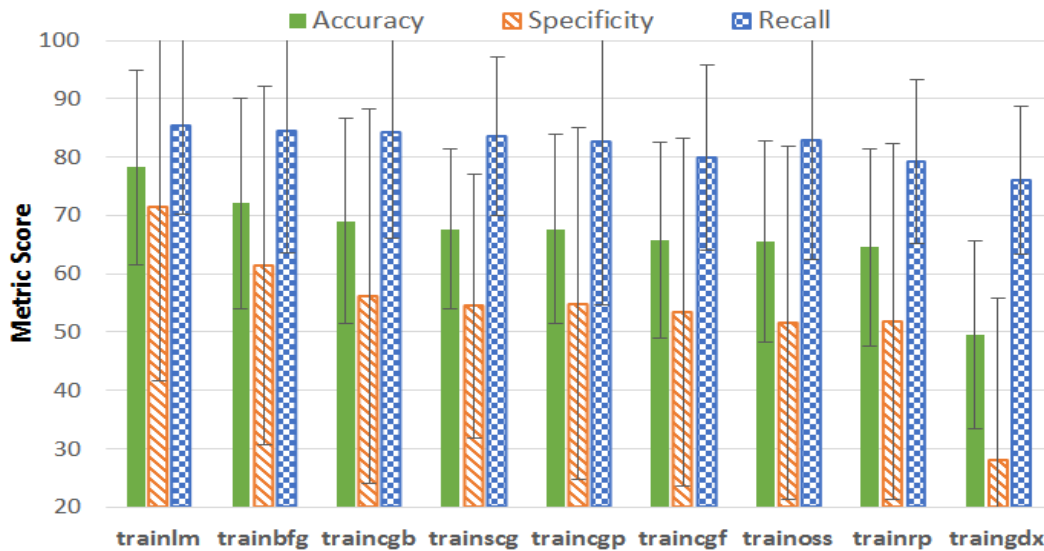


Figure 6: Impact of training function for FFNN and CNN. Bar chart shows average and standard deviation of top accuracy configurations for each of the three datasets.

proved to be the least effective in our experiments. Average accuracy scores of other train functions do not highlight relevant fluctuations. However, while Recall – which depends on false negatives – is almost constant, it seems that the usage of trainlm allows lowering false positives, consequently raising Specificity (see orange-striped bars in Figure 6).

3.4.1.5 SVM: The impact of kernels

A similar sensitivity analysis was conducted also for SVM kernels. Figure 7 shows the Accuracy, Specificity, and Recall obtained by SVMs on our 3 datasets using Linear, RBF, and polynomial (2nd degree) kernels. RBF kernel enables SVM to provide perfect accuracy (100.0%) on the GTSRB dataset, almost on par with the linear kernel (99.7%). Nevertheless, the linear kernel is the preferred choice for DITS and BelgiumTSC datasets as it is shown by higher triples of bars in Figure 7. We explain this result as follows: RBF is usually preferable

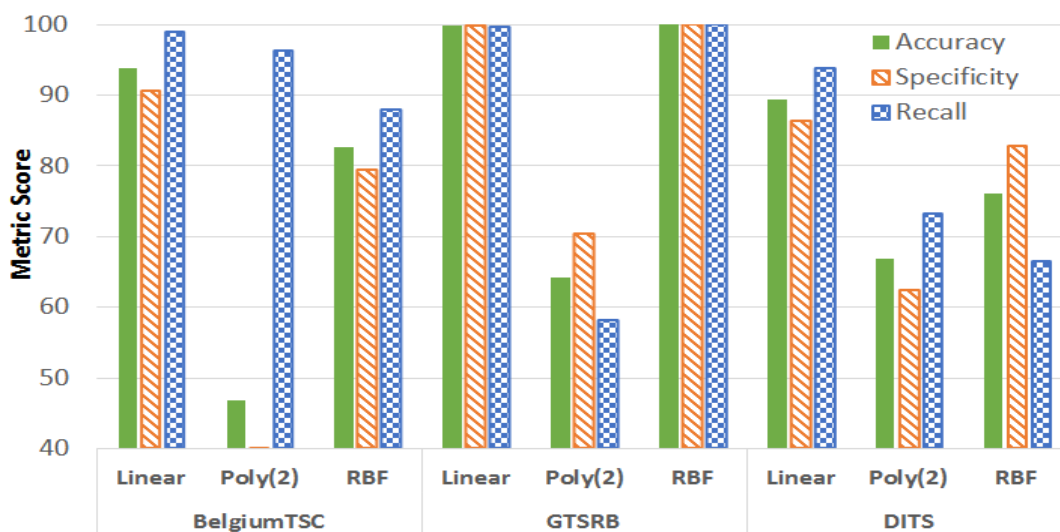


Figure 7: Impact of SVM’s Kernels on maximum accuracy reached on each dataset. Plot also reports on Specificity and Recall.

when data is not linearly separable, while linear kernel should be the preferred choice when we assume linear separability of the hyperplane. As a result, we can conjecture that data is linearly separable in BelgiumTSC and DITS datasets, leading Linear kernel to be more effective than others in performing classification.

3.4.2 Results of Deep Neural Networks

After discussing the results of non-deep classifiers, we explore here the results of DNNs, i.e., InceptionV3, MobileNet-V2, and AlexNet. Table 6 indicates accuracy scores achieved by DNNs by varying learning rates as described in Section 3.2.4, highlighting higher scores with bold-font numbers with grayed background. Results indicate that MobileNet-V2 and InceptionV3 provide 100% accuracy in GTSRB dataset with a learning rate of 0.01 and 0.005 respectively. The highest accuracy for BelgiumTSC was achieved by the InceptionV3 network

Table 6: Accuracy achieved on three deep learning models for each of three datasets varying learning rates.

	<i>InceptionV3</i>		<i>MobileNet-V2</i>		<i>AlexNet</i>	
	LR	Accuracy	LR	Accuracy	LR	Accuracy
<i>GTSRB</i>	0.01	100.00	0.01	99.97	0.00009	99.99
			0.05	99.10	0.00005	99.99
			0.005	100.00	0.00001	99.97
					0.000009	99.97
					0.000005	99.95
					0.000001	99.63
<i>Belgium TSC</i>	0.05	87.43	0.05	91.50	0.00009	98.81
	0.01	98.21	0.01	97.16	0.00005	98.62
	0.005	99.22	0.005	98.26	0.00001	98.30
	0.001	99.17	0.001	98.17	0.000009	98.35
	0.0005	99.31	0.0005	98.08	0.000005	98.40
	0.0001	98.67	0.0001	98.26	0.000001	98.30
<i>DITS</i>	0.05	97.30	0.05	45.40	0.00009	99.10
	0.01	94.41	0.01	97.90	0.00005	96.90
	0.005	97.30	0.005	99.70	0.00001	97.00
	0.001	99.10	0.001	99.70	0.000009	97.20
	0.0005	98.90	0.0005	99.80	0.000005	97.10
	0.0001	99.50	0.0001	98.40	0.000001	96.90

with LR of 0.0005, while the same learning rate allowed MobileNet-V2 to reach top accuracy of 99.80% in DITS. Instead, AlexNet results indicate that the results of this network, albeit still high, are not as high as the ones achieved by its competitors. To the best of our knowledge, this difference may be due to the number of hidden layers DNNs employ: while InceptionV3 and MobileNet-V2 respectively rely on 48 and 53 hidden layers, only 8 hidden layers are used in AlexNet. This lower number of convolutional layers may have negatively been impacting transfer learning procedures, making AlexNet unable to provide excellent results as the other two DNNs do. As a last remark, Table 6 shows how a learning rate of 0.0009 is always beneficial for AlexNet with respect to the other learning rates we used in our experiments.

3.5 *Lessons Learned*

Independent analyses and discussions of results in Section 3.4.1 and Section 3.4.2 provided interesting findings concerning both non-deep classifiers and DNNs. To complete and wrap up our analysis, we proceed here to compare the results of these two approaches to finding conclusive lessons and takeovers of this part of the thesis. It is worth mentioning that the GTRSB dataset turned out to contain images of traffic signs that were classified perfectly by both non-deep classifiers (e.g., *KNN*, *SVM*) and DNNs (e.g., MobileNet-V2, InceptionV3). The discussion on the other two datasets requires additional thinking we carry out with the aid of Figure 8, which reports on the Accuracy, Specificity and Recall scores for *KNN*, *SVM*, Boosting, InceptionV3, MobileNet-V2, which showed overall good metric scores with respect to their competitors. The plot in Figure 8a expands on BelgiumTSC dataset. It is straightforward to notice how InceptionV3 outperforms other algorithms, with particularly good specificity scores. Such a DNN has a very low number of false positives, or rather it hardly misclassifies either a red triangular or blue circular traffic sign for a red circular one. Such scores are very close to the optimum (accuracy is 99.31%), as only 0.7% of the traffic signs are being misclassified in BelgiumTSC by InceptionV3.

Instead, Figure 8b details the performance of classifiers on the DITS dataset. Differently from Figure 8a, it is evident that DNNs remarkably outperform non-deep classifiers. In fact, while the best non-deep classifiers (*KNN* and Boosting) hover around a 4.2% of misclassification rate, the accuracy of MobileNet-V2 reaches 99.80% with perfect Recall. This means that only 0.2% of the images are being misclassified by MobileNet-V2 on DITS, and that all misclassifications are false positives, or rather either blue circular or red triangular traffic signs that were classified as red circular “danger” signals. This is a very good result if we consider that the DITS dataset is very challenging due to traffic signs that appear with non-

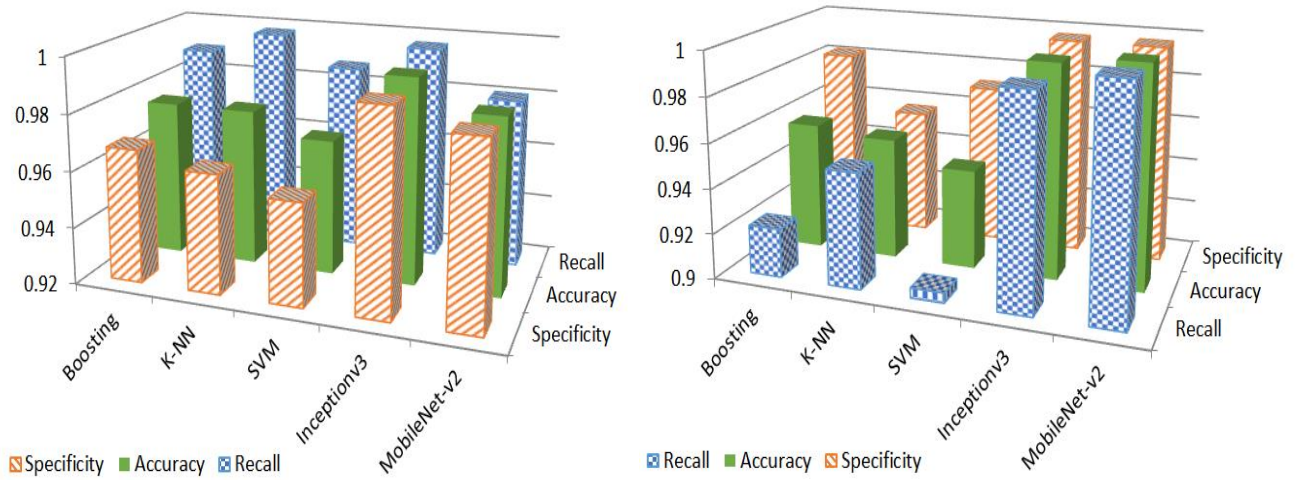


Figure 8 a and b. Specificity, Accuracy and Recall achieved by most performing non-deep classifiers (Boosting, KNN, SVM) and deep learners (InceptionV3, MobileNet-V2) on BelgiumTSC (left, Figure 8a) and on DITS (right, Figure 8b) datasets.

optimal lighting conditions and with blurred areas of several images. Moreover, MobileNet-V2 never misclassifies red circular signals (no false negatives at all), which may directly harm the driver or other objects in the environment. We can suppose that misclassifying a blue circular “informative” traffic sign for a “danger” signal may trigger a false alarm to the driver or even let the car to automatically slow down, but it is nowhere nearly dangerous as missing a stop or pedestrian crossing sign and entering the street with potentially catastrophic consequences.

Overall, results showed that *KNN*, *SVM*, Random Forest, and Neural Networks are the most adequate non-deep classifiers, and that HOG features provided actionable information to be processed by those algorithms. We also verified how employing more than one feature set extracted from heterogeneous descriptors can increase the recognition accuracy. Overall, non-deep classifiers achieved the highest accuracy of 100% on GTSRB, 97.58% on BelgiumTSC, and 95.71% on DITS with a combination of HOG, LBP, and SURF features. Furthermore, we compared those results with DNNs AlexNet, MobileNet-V2, and InceptionV3, which were adapted to TSR through transfer learning and achieved higher accuracy than classifiers on DITS and BelgiumTSC i.e., 99.80% and 99.31% respectively.

Wrapping up this chapter of the thesis, our results suggest that DNNs have the potential to reduce misclassifications for TSR with respect to non-deep classifiers, albeit transfer learning to tailor those networks to a specific domain must be conducted and planned carefully.

4. ENHANCING TRAFFIC SIGN RECOGNITION THROUGH SLIDING WINDOWS

Traffic Sign Recognition (TSR) techniques that use Machine Learning (ML) algorithms have been proposed, but no agreement on a preferred ML algorithm nor perfect classification capabilities were always achieved by any existing solutions. Regardless of the outcomes of comparison studies, most of the existing solutions for TSR process a single image and output a classification result. Instead, vehicles gradually approach traffic signs during their road trips, generating sequences of images: the closer the vehicle is to the traffic sign, the better the quality of the image, even under slightly different environmental conditions. Therefore, the problem of TSR naturally scales to knowledge extraction from a set or sequence of images that potentially contain traffic signs. Consequently, the classification process should not depend only on a single image to make a decision; instead, it should build on the knowledge acquired as the vehicle moves forward, i.e., the sequence of images.

This chapter of the thesis considers a sliding window of images to commit classification rather than classifying images individually. First, we process each image with the most effective single image classifier for TSR: then, we combine classification scores assigned to images in the sliding window to provide a unified and improved classification result. Such a combination is performed by appropriate Meta-Learners [30], which suit model combination, and therefore, show potential to be applied in such a context. We conduct an experimental evaluation by processing three public datasets, namely, (i) German Traffic Sign Recognition Benchmark (GTSRB) [31] (ii) BelgiumTSC [32], and (iii) the Dataset of Italian Traffic Signs (DITS) [33], which report on sequences or unordered sets of images of traffic signs. From each image, we extracted 12 different feature sets, which use handcrafted features (Histogram of Oriented Gradients (HOG) [17], Local Binary Pattern (LBP) [16]), deep features (from AlexNet [34], ResNet18 [35]), and their combinations, to debate their impact in TSR. Those features were fed to non-deep classifiers as Decision Trees [36], Random Forests [37], k-th Nearest Neighbour (*KNN*, [13]), Linear Discriminant Analysis Classifier (LDA) [38], Support Vector Machines (SVMs) [14], and AdaBoost [39]. We also exercised single-image classifiers that do not rely on feature descriptors as Deep Neural Networks (DNNs), namely InceptionV3 [40], MobileNet-V2 [41] and AlexNet [34]. We used the classifiers above both as single-image classifiers and as base-level learners of a Stacking meta-learner, which aggregates individual

classification scores into sliding windows. The meta-level classifier for Stacking was experimentally chosen out of non-deep classifiers, the Majority Voting [42] and Discrete Hidden Markov Model (DHMM, [43]). Additionally, we compare the classification performance of those meta-learners with Long Short-Term Memory (LSTM) networks, which naturally deal with sequences of data coming at different time instants. We trained those LSTM networks on the same sliding windows of images processed through Stacking.

4.1 Sliding Windows to Improve TSR

TSR naturally fits the analysis of sequences of images being collected as the vehicle approaches the traffic sign. Therefore, we organize a complex classifier that processes sliding windows of images as shown in Figure 9, a sliding window of size s contains (i) the most recent image sampled by the sensors on the vehicle plus (ii) the $s-1$ most recent images. The figure represents how sliding windows of size $s = 2$ and $s = 3$ evolve as time passes by as the vehicle approaches a speed limit sign. Intuitively, the closer the vehicle gets to the traffic sign, the more visible and clearer the traffic sign gets. On the other hand, the sooner the TSR correctly classifies a traffic sign, the better it is for the Advance Driving Assistance System (ADAS), e.g., it may provide more time for emergency braking, whenever needed.

4.1.1 Sliding Windows and Meta-Learning

Adopting sliding windows of s images calls for a rework of the TSR system. In particular, classification should be carried out using s subsequent classifications, which contribute to the final decision on the traffic sign. Those single-image classifications for subsequent images have

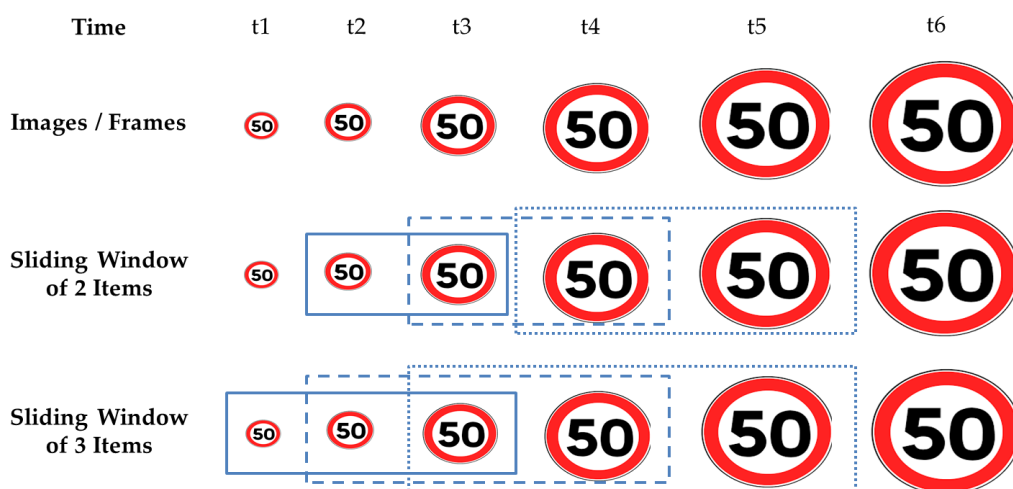


Figure 9: Example of Sliding Windows. Dotted, dashed and solid boxes show sliding windows, respectively, at t5, t4, t3.

to be combined by utilizing an independent strategy that delivers the result of this ensemble of

single-image classifiers. Such a combination is usually orchestrated through meta-learning [30], [71], which uses knowledge acquired during base-learning episodes, i.e., meta-knowledge, to improve classification capabilities. More specifically [72], a base-learning process starts feeding images into one or more base classifiers to create meta-data at the first stage. Results of those base learners, i.e., meta-data are provided alongside other features to the meta-level classifier as input features, which in turn provides the classification result of the whole meta-learner. The paradigm of meta-learning can be adapted to TSR as shown in Figure 10. Let k be the number of different categories of traffic signs (i.e., classes), and let s be the size of the sliding window. Starting from the left of the figure, images are processed by means of single-image base-level classifiers, which provide k probabilities $PTS_i = \{pts_{i1}, \dots, pts_{ik}\}$ to classify each image. Depending on the current time t_j , we create a sliding window of at most $s \times k$ items, namely $sw_{sj} = \{PTS_j, PTS_{j-1}, \dots, PTS_{j-s+1}\}$, which builds the meta-data to be provided to the meta-level classifier. On the right side of Figure 10, the meta-level classifier processes such meta-data and provides the k probabilities PTS_{final} , which will constitute the classification result of the whole sequence within the sliding window. As time moves on, we will have newly captured images and the sliding window will process the most recent $s \times k$ items. Note, that the sliding window sw_{sj} may contain less than $s \times k$ items when $j < s$ (e.g., the window of size 3 at time t_2 in Figure 9). In those cases, the TSR system will decide based on a single-image classification of the most recent image.

4.2 A Stacking Meta-Learner

The structure of the meta-learner we described previously is traditionally referred to as Stacking. Stacking [73] builds a base-level of different classifiers as base learners. Base-learners can be trained with the exact same training set or with different training sets, mimicking Bagging [74]. Each of the n base-learners generates meta-features (PTS_i , $1 \leq i \leq n$ in Figure 10) that are fed to another independent classifier, the meta-level classifier, which calculates and delivers the final output (PTS_{final} in Figure 10). In our instantiation of the Stacker, we use the same base-level classifier, which can either be a DNN or a non-deep

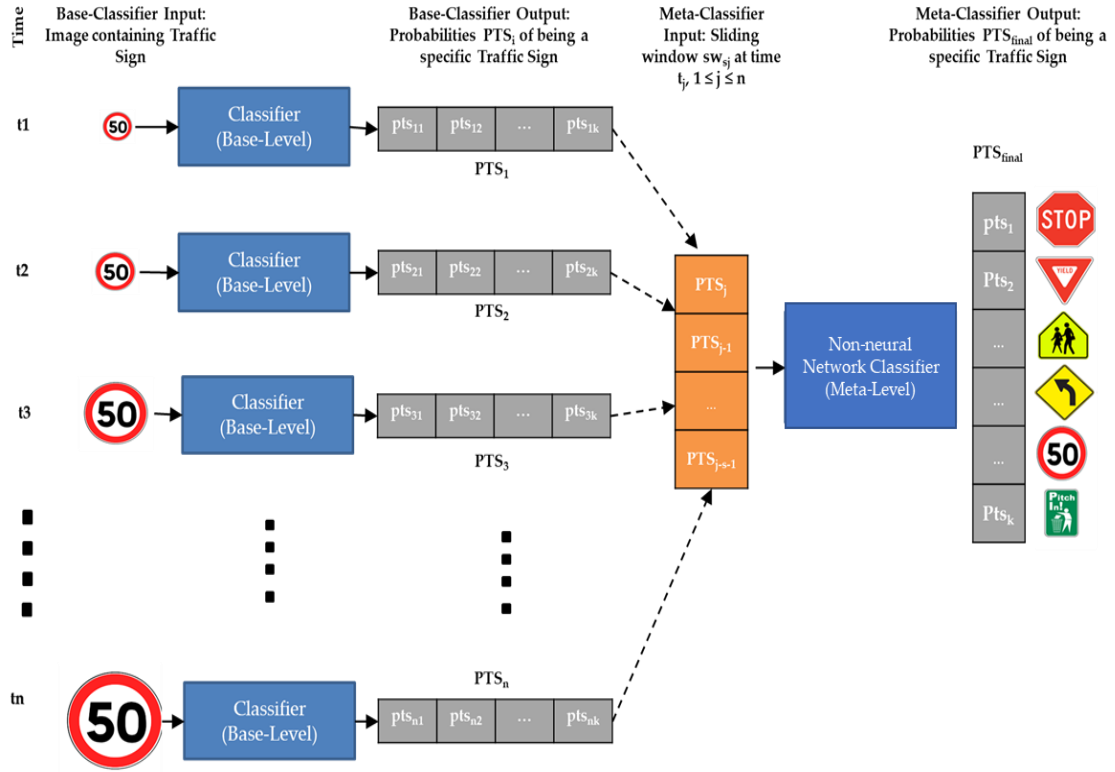


Figure 10: Diagram representing TSR which uses sliding windows. Blue solid boxes represent single-image classifier in Figure 4.

classifier but feed each base-learner with a different image. The meta-level classifier is necessarily a non-deep classifier as it has to process numeric features contained in sw_{sj} rather than images.

4.3 Methodology, Inputs, and Experimental Setup

This section describes the methodology, inputs, and experimental setup to compare single-image classifiers and approaches built upon sliding windows, such as Stacking and LSTM networks.

4.3.1 Methodology for a Fair Comparison of TSR Systems

We orchestrate our experimental methodology as follows:

- **Datasets and Pre-processing.** Images go through a pre-processing phase to resize them to the same scale and enhance the contrast between background and foreground through histogram equalization.
- **Feature Extraction (Section 4.3.3)** Then, each pre-processed image is analyzed to extract features: these will be used with non-deep classifiers, while DNNs will be directly fed with pre-processed images.

- **Classification Metrics (Section 4.3.4).** Before exercising classifiers, we select metrics to measure the classification capabilities of ML algorithms which apply both to single-image classifiers and to others based on sliding windows.
- **Single-Image Classification.** Both non-deep (Section 4.3.5.1) classifiers and DNNs (Section 4.3.5.2) will be trained and tested independently, executing grid searches to identify proper values for hyper-parameters.
- **Sliding Windows with Stacking Meta-Learners (Section 4.3.5.3).** Results of single-image classifiers will then be used to build Stacking learners as described in Section 4.2 and by adopting different meta-level classifiers.
- **Sliding Windows with LSTM (Section 4.3.6).** Furthermore, sliding windows will be used to exercise LSTM networks as described in Section 2.2.6.

4.3.2 TSR Datasets and Traffic Sign Categories

We conducted extensive research to identify commonly used labelled datasets, namely: (i) the BelgiumTSC dataset [32], (ii) the GTSRB dataset [31], and (iii) the DITS [33] described in Section 2.2.7, that are reporting on sequences of traffic signs with overlapping categories.

4.3.3 Feature Descriptors

We extract features from images by means of *handcrafted*, i.e., HOG, LBP and *deep*, i.e., AlexNet and ResNet, feature descriptors, as described in Section 2.2.1.2 and 2.2.2 respectively. In addition, we combine hand-crafted and deep feature descriptors that are consequently fed simultaneously to classifiers: couples as {HOG U LBP}, {AFeat U HOG}, {AFeat U LBP}, {RFeat U HOG}, {RFeat U LBP}, {AFeat U RFeat}, and triples of {AFeat U HOG U LBP} and {RFeat U HOG U LBP}.

4.3.4 Classification Metrics

The performance of classifiers different strategies for TSR are evaluated using a performance metric accuracy that is described in detail in Section 2.2.8.

4.3.5 Non-deep Classifiers and Deep Neural Network Hyper-Parameters

This Section provide the detailed of the hyperparameters that we used to train non-deep classifiers and DNNs.

4.3.5.1 Hyper-parameters of non-deep classifiers

Each non-deep algorithm has its own set of hyper-parameters. To such an extent, we identified the following parameter values to exercise grid searches.

- *KNN with different values of k , i.e., different odd values of k from 1 to 25. Additionally, we observe that DITS contains nine categories of traffic signs: therefore, we disregard the usage of $k = 9$ to further avoid ties.*
- *SVM: we used three different kernels: Linear, RBF and Polynomial (quadratic), leaving other parameters (e.g., ν) as default.*
- *Decision Tree: we used the default configuration of MATLAB which assigns `MaxNumSplits = training sample size 1`, with no depth limits on decision trees.*
- *Boosting: we created boosting ensembles with AdaBoostM2 by building 25, 50, 75, and 100 trees (decision stumps) independently.*
- *Random Forest: we build forests of 25, 50, 75, or 100 decision trees.*
- *LDA: we Trained LDA using different discriminants, namely: pseudo-linear, diag-linear, diag-quadratic, and pseudo-quadratic.*

4.3.5.2 Hyper-parameters of deep neural networks

DNNs may be either built from scratch or more likely—by adapting existing models to a given problem through transfer learning (i.e., knowledge transfer). Through transfer learning, we fine-tune the fully connected layers of the DNN, letting all convolutional layers remain unchanged. Those DNNs can be tailored to TSR through transfer learning. Fully connected layers are trained on defined categories of traffic signs with different learning rates (LR) to fine-tune the models which are already trained on the ImageNet database of 1000 categories. Additionally, we employ data augmentation to avoid model overfitting; this was conducted through X and Y translation with a random value between $[-30, 30]$ and scale range within a range $[0.7, 1]$. The hyper-parameter learning rate controls how fast weights are updated in response to the estimated errors, and therefore, controls both the time and the resources needed to train a neural network. Choosing the optimal learning rate is usually a tricky and time-consuming task: learning rates that are too big may result in fast but unstable training, while small learning rates usually trigger a heavier training phase which may even get stuck without completing correctly. In our experiments, we varied learning rate as follows: $\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$ for InceptionV3 and MobileNet-V2, and $\{0.0001, 0.0005, 0.00001, 0.00005, 0.000005, 0.000001\}$ for AlexNet, which resulted in very low accuracy when using the same learning rates of the InceptionV3 and MobileNet-V2. Noticeably, training a DNN with the highest learning rate in the interval reduce the training time with respect to using the smallest value in the interval (e.g., training InceptionV3 with a learning rate of 0.05 instead of using 0.0001). We set a minimum batch size of 32, with 10 train epochs and stochastic gradient

descent with momentum (sgdm) optimizer for all the experiments on each dataset to fine-tune the models for TSR. Furthermore, we used the loss function ‘crossentropyex’ at the classification layer and the fully connected weights and biases were updated with a learning factor (different from learning rate) of 10. We had the weights vector size associated with the last fully connected layers $[\text{Num_cat} \times 4096]$, $[\text{Num_cat} \times 1280]$, and $[\text{Num_cat} \times 2048]$ for AlexNet, MobileNet-V2 and InceptionV3 models, respectively, where Num_cat represented the number of traffic sign categories in each dataset.

4.3.5.3 Hyper-parameters of stacking meta-level learners

The parameters we used to execute grid searches and train meta-level learners above are as follows.

- *Majority Voting: no parameter is needed.*
- *Each DHMM model was trained with 500 iterations.*
- *Non-deep Classifiers: we used the same parameter values we already presented in Section 4.3.5.1.*

4.3.6 Long-Short Term Memory (LSTM) Networks

LSTM networks are artificial recurrent neural networks, which efficiently process sequences of images, and therefore, suit the classification of sequences of traffic signs. LSTM networks are trained on all 12 feature sets in Section 4.3.3 independently considering three different training functions or optimizers, i.e., Adaptive Moment Estimation (adam) [133], Stochastic Gradient Descent with Momentum (sgdm) [134], and Root Mean Square Propagation (rmsprop) [135] with a learning rate of 0.001.

4.4 Experimental Results

This section reports and discusses the results of our experimental campaign. We split the results into two sub-sections: First Section 4.4.1 describes the experimental results of single-image classifiers, while the Section 4.4.2 reports on the results achieved by classifiers that consider sliding windows of images.

4.4.1 TSR Based on Single Image

First, we elaborate on the classification performance of TSR systems that process images individually. The results described in this section are similar to Section 3.4 but with different number of traffic sign classes. Initially, in Section 3.4 we described the result of non-deep and

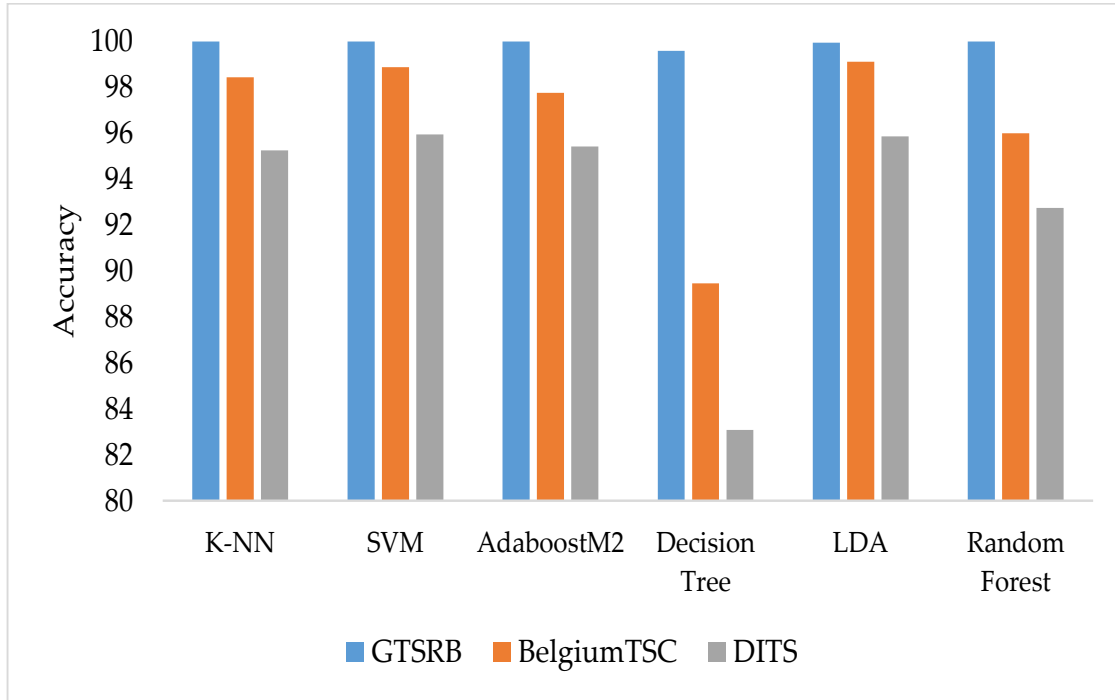


Figure 11: Highest accuracy achieved by non-deep classifiers on each dataset.

deep classifiers considering three broad categories of traffic signs i.e., red circular, blue circular and red triangular, while in this sections we increased the complexity of the dataset and instead of 3, we use 9 categories of traffic signs as shown previously in Table 2.

4.4.1.1 Highest accuracy for each dataset

Figure 11 depicts a bar chart diagram reporting the highest accuracy achieved by classifiers in each of the three datasets. It is clear from the blue solid bars in Figure 11 that almost all classifiers give better performance on the GTSRB dataset compared to the other two datasets, i.e., BelgiumTSC and DITS. All classifiers in the figure but Decision Tree and LDA achieve perfect accuracy on the GTSRB dataset. The reason behind the high accuracy may be the higher number of training samples and better image quality of the GTSRB dataset compared to the other two datasets. Instead, SVM provides the highest accuracy of 95.94% in DITS, with LDA that comes close at 95.85%. Consequently, the highest accuracy in each dataset is not always achieved by the same algorithm, despite *KNN*, SVM, and LDA performing better overall compared to other non-deep classifiers.

4.4.1.2 Impact of feature descriptors

Table 7 further elaborates on the impact of features on accuracy scores achieved by non-deep classifiers on each dataset. Non-deep classifiers achieve perfect accuracy with all feature descriptors on GTSRB. Instead, the combination of AFeat and RFeat builds a feature descriptor that allows algorithms to achieve the highest accuracy of 95.94% for DITS and 99.12% for BelgiumTSC. Additionally, AFeat and RFeat descriptors provide features that allow algorithms

Table 7: Highest accuracy achieved using different feature descriptors on each dataset (bold high-lighted values represent the highest achieved accuracy across each dataset).

Feature Descriptor (s)	GTSRB	DITS	BelgiumTSC
AFeat	100.00	95.51	98.84
RFeat	100.00	94.13	97.76
LBP	100.00	79.98	93.49
HOG	100.00	87.92	96.24
HOG \cup LBP	100.00	88.26	96.56
AFeat \cup RFeat	100.00	95.94	99.12
AFeat \cup HOG	100.00	95.68	98.96
AFeat \cup LBP	100.00	95.85	98.96
RFeat \cup HOG	100.00	95.51	98.72
RFeat \cup LBP	100.00	95.85	98.80
AFeat \cup HOG \cup LBP	100.00	95.42	98.88
RFeat \cup HOG \cup LBP	100.00	95.34	98.84

to reach higher accuracy. By using just a single feature descriptor AFeat always achieves the highest accuracy on all three datasets, while the second highest accuracy is achieved by RFeat. Instead, using only LBP, HOG or their combination generates accuracy scores that are lower than potential alternatives. Moreover, it is worth noticing how combining feature descriptors provides features that increase the classification performance of non-deep classifiers, such as: from 95.51% to 95.94% in DITS, and from 98.84% to 99.12% in BelgiumTSC.

4.4.1.3 Results of deep neural networks

We explore the results of the DNNs considered in this study with the aid of Table 8, which shows accuracy scores achieved by those classifiers for different learning rates. MobileNet-V2 achieves the highest accuracy out of the three DNNs for the GTSRB dataset with a learning rate of 0.001, whereas a learning rate of 0.00005 maximizes the accuracy scores of AlexNet on the BelgiumTSC dataset. Instead, the learning rate of 0.0001 allows InceptionV3 to reach the maximum accuracy of 96.03% for the DITS dataset, outperforming MobileNet-V2 and AlexNet, which instead achieves the highest accuracy in the BelgiumTSC dataset with a learning rate of 0.0005. Interestingly, whereas accuracy scores for GTSRB do not vary a lot when using different learning rates, the choice of the learning rate becomes of paramount importance when classifying DITS and BelgiumTSC datasets. Particularly, the bottom of Table 8, the third column, shows a 14.97% accuracy on the BelgiumTSC dataset using learning rates of 0.05 and 0.005, which is a very poor achievement. For these learning rates, the training process was unstable, with weights that were updated too fast and ended up with a classifier that has semi-random classification performance. Unfortunately, we could not identify a single DNN that outperforms others in all three datasets.

Table 8: Accuracy achieved by deep learners for each of the three datasets with varying learning rates (bold highlighted values represent the highest achieved accuracy across each dataset by deep classifiers).

	InceptionV3		MobileNet-V2		AlexNet	
	LR	Acc	LR	Acc	LR	Acc
GTSRB	0.01	96.62	0.01	96.11	0.0001	95.98
	0.05	93.56	0.05	93.38	0.0005	94.92
	0.001	96.95	0.001	99.35	0.00001	94.86
	0.005	97.06	0.005	96.42	0.00005	95.64
	0.0001	96.81	0.0001	96.83	0.000001	95.83
	0.0005	98.03	0.0005	96.65	0.000005	96.07
DITS	0.01	80.06	0.01	93.52	0.0001	87.40
	0.05	80.67	0.05	85.93	0.0005	86.45
	0.001	88.17	0.001	94.99	0.00001	95.51
	0.005	84.98	0.005	88.78	0.00005	92.06
	0.0001	96.03	0.0001	95.77	0.000001	92.23
	0.0005	91.88	0.0005	95.94	0.000005	95.16
BelgiumTSC	0.01	89.58	0.01	97.16	0.0001	99.24
	0.05	14.97	0.05	94.49	0.0005	92.57
	0.001	98.12	0.001	98.72	0.00001	99.52
	0.005	14.97	0.005	94.73	0.00005	99.72
	0.0001	99.64	0.0001	99.24	0.000001	97.92
	0.0005	99.68	0.0005	98.96	0.000005	99.24

4.4.2 TSR Based on Sliding Windows

This section elaborates on the classification performance of TSR systems that process a sliding window of multiple images.

4.4.2.1 Meta learning with non-deep base classifiers

Table 9 reports scores achieved by stacking meta-learners built using (i) the three non-deep classifiers that performed better in Section 4.4.1.1 as base learners, and (ii) different meta-level learners, such as *KNN*, SVM, LDA, Decision Tree, Majority Voting, Boosting, Random Forest and DHMM. The GTSRB dataset does not appear in Table 9 since single-image non-deep classifiers alone already achieved perfect classification. The table reports the highest accuracy scores achieved by each stacking meta-level classifier by using different combinations of base-learners (*KNN*, SVM, LDA) and window sizes of two and three items.

Overall, LDA as a base-level classifier with a *KNN* meta-level classifier is the preferred choice (bolded values in Table 9) on DITS and on BelgiumTSC with a sliding window of three items. Instead, using ensembles of Decision Trees as AdaBoost and Random Forests sparingly gives very low accuracy scores (see italicized numbers in the 10th and 11th columns of Table 9), showing that those two classifiers do not always adequately play the role of a meta-level classifier for a stacker. Results for DITS in Table 9 show that using a sliding window of three

items generally improves accuracy with respect to using a sliding window of only two items. A sliding window of three items allowed stacking meta-learners, which used *KNN* or *LDA* as meta-level classifiers to reach perfect accuracy (100%) on the DITS dataset using either *LDA* or *SVM* as base-learners. This result was largely expected: the more information is available (i.e., wider sliding window), the fewer misclassifications we expect from a given classifier. Instead, we obtained maximum accuracy for the BelgiumTSC by using a sliding window of two items, whereas using three items often degrades classification performance. At a first glance, this result is counter-intuitive with respect to previous discussions. However, the reader should note that the BelgiumTSC dataset reports on a set of images of the same traffic signs which are captured with multiple input visual cameras without any temporal order. Consequently, the sliding window for the BelgiumTSC contains images of the traffic sign which are taken from different angles and may lead the meta-learner to lean towards misclassifications rather than improving accuracy. In fact, for this dataset, there is no direct relation between the size of the window and accuracy values, which instead turned out to be evident for the other datasets.

Table 9: Results achieved using different meta-learners considering non-deep classifiers as base classifiers varying window size (WS). We bolded the highest achieved accuracy using different combinations across each dataset and low accuracy achieved through AdaBoostM2 and Random Forest italicized numbers in the 10th and 11th columns.

Dataset	Base-Level Classifier	Single Image Accuracy	WS	Stacking Meta-Level Classifier							
				Majority Voting	KNN	SVM	LDA	Decision Tree	AdaBoostM2	Random Forest	DHMM
BelgiumTSC	<i>KNN</i>	98.44		99.40	99.04	98.8	98.80	98.68	98.80	<i>51.86</i>	98.56
	SVM	98.88	2	99.52	99.64	99.76	98.68	99.64	99.28	98.80	99.04
	LDA	99.12		99.64	99.40	99.28	99.28	98.32	98.92	97.84	99.40
	<i>KNN</i>	98.44		99.40	99.40	99.04	98.92	98.44	98.32	63.47	98.20
	SVM	98.88	3	99.52	99.52	99.64	99.40	98.56	<i>14.97</i>	99.28	98.80
	LDA	99.12		99.64	99.64	99.40	98.2	99.28	97.96	98.32	98.80
DITS	<i>KNN</i>	95.25		97.56	97.56	97.56	96.75	97.56	97.56	82.93	96.75
	SVM	95.94	2	96.75	97.56	98.37	97.56	95.12	<i>31.71</i>	95.12	95.93
	LDA	95.85		98.37	98.37	97.56	97.56	98.37	95.93	96.75	96.75
	<i>KNN</i>	95.25		99.00	99.00	99.00	99.00	99.00	99.00	85.00	98.00
	SVM	95.94	3	99.00	100	99.00	99.00	97.00	<i>36.00</i>	97.00	96.00
	LDA	95.85		99.00	100	98.00	100	99.00	98.00	99.00	98.00

4.4.2.2 Meta learning with base-level deep neural network classifiers

Table 10 has a structure similar to Table 9 but employs base-level DNNs to build the stacking meta-learner, and also reports on all datasets as DNNs based on a single image but did not achieve perfect accuracy on any of the three datasets. Base-level DNNs in conjunction with *KNN* as a meta-level classifier achieved perfect classification on all three datasets, as shown by bold values in Table 10. GTSRB turns out to be the dataset that provides the higher average accuracy by using a different base and meta-level classifiers. The highest achieved accuracies are highlighted in Table 10 with bold typeset. It is very interesting to discuss that all three DNNs (base-level classifiers) with meta-level classifiers *KNN*, LDA, Boosting and Random Forest give 100% accuracy, while MobileNet-V2 achieves 100% accuracy with all meta-level classifiers for a sliding window of size 2 or 3 on the GTSRB dataset. InceptionV3 and MobileNet-V2 with meta-level classifiers (*KNN*, AdaboostM2) achieve 100% accuracy on the DITS dataset for sliding windows of size 2 and 3, respectively, While AlexNet base-level

Table 10: Results achieved using different meta learners with DNNs as base classifiers with varying window size (WS). We bolded the perfect classification (100% accuracy).

Dataset.	Base-Level Classifier	Single Image Accuracy	WS	Majority Voting	KNN	SVM	LDA	Decision Tree	Ada-BoostM2	Random Forest	DHMM
BelgiumTSC	AlexNet	99.72		99.88	100.00	99.64	99.88	99.88	99.40	99.16	99.76
	InceptionV3	99.68	2	99.88	99.88	99.64	99.88	99.52	99.88	99.64	99.64
	MobileNet-V2	99.24		99.52	99.88	99.64	99.04	99.64	99.52	98.68	99.64
	AlexNet	99.72		100.00	100.00	99.28	99.88	99.88	99.76	99.76	99.28
	InceptionV3	99.68	3	99.88	99.88	99.40	99.52	99.40	14.97	99.76	99.52
	MobileNet-V2	99.24		99.88	99.88	98.80	99.16	99.40	14.97	99.76	99.52
DITS	AlexNet	95.51		96.75	97.56	97.56	97.56	96.74	96.74	96.74	98.37
	InceptionV3	96.03	2	98.37	100.00	97.56	98.37	98.37	96.74	95.93	99.19
	MobileNet-V2	95.94		97.56	99.18	99.18	99.18	99.18	100.00	98.37	99.19
	AlexNet	95.51		97.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00
	InceptionV3	96.03	3	98.00	100.00	99.00	98.00	99.00	100.00	99.00	98.00
	MobileNet-V2	95.94		98.00	100.00	99.00	100.00	99.00	100.00	100.00	100.00
GTSRB	AlexNet	96.07		97.37	100.00	99.76	100.00	98.09	100.00	100.00	98.09
	InceptionV3	98.03	2	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.76
	MobileNet-V2	99.35		100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	AlexNet	96.07	3	0.9737	100.00	99.76	100.00	98.09	100.00	100.00	98.09
	InceptionV3	98.03		100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.76
	MobileNet-V2	99.35		100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 11: Accuracy of LSTM with window sizes 2 and 3.

Dataset	WS	Optimizer		
		adam	Sgdm	rmsprop
DITS	2	97.56	97.56	96.74
DITS	3	99.00	99.00	98.00
BelgiumTSC	2	99.40	99.16	99.16
BelgiumTSC	3	99.64	99.28	99.40

classifier with Majority voting and *KNN* as the meta-level classifier achieves 100% accuracy for both sliding windows of size 2 & 3 on BelgiumTSC dataset.

Similarly, to Table 9, we observe that AdaboostM2 does not show up as a reliable meta-level classifier as it provides very low accuracy for the BelgiumTSC with a sliding window of three images. All meta-level classifiers with base-level classifier Mobilenet-V2 achieve 100% accuracy on the GTSRB dataset, whose sequences contain 30 images of the same traffic sign, and therefore, provide much information for the stacking classifier to classify traffic signs as the window slides.

4.4.2.3 Results of LSTM networks

Table 11 reports accuracy scores of LSTM networks on the BelgiumTSC and DITS datasets with a sliding window of size 2 or 3. Similarly to Section 4.4.2.1, we omit the GTSRB dataset since it is perfectly classified by single-image non-deep classifiers. We independently trained the LSTM by using each of the 12 feature sets in Section 4.3.3, with different window sizes (WS) and by using three different optimizers: adam, sgdm and rmsprop. Table 11 reports the highest accuracy achieved by LSTM by using a given WS and optimizer function. It is evident how the adam optimizer always allows achieving the highest accuracy scores in both datasets and with different WS. Additionally, accuracy is always higher when using a window of size 3 with respect to a window containing only two items: this was expected for DITS, whose images are time-ordered, but it is also verified for the BelgiumTSC, which does not have such ordering. Overall, the results of the LSTM are slightly lower than stacking meta-learners using non-deep base-level classifiers and clearly worse than stacking using base-level DNNs, which achieves perfect accuracy on all datasets.

4.4.3 Comparing Sliding Windows and Single-Image Classifiers

Independent analyses and discussions of results in Sections 4.4.1 and 4.4.2 provided interesting findings concerning both non-deep classifiers and base-level DNNs and the usage of sliding windows to improve the classification performance through meta-learning.

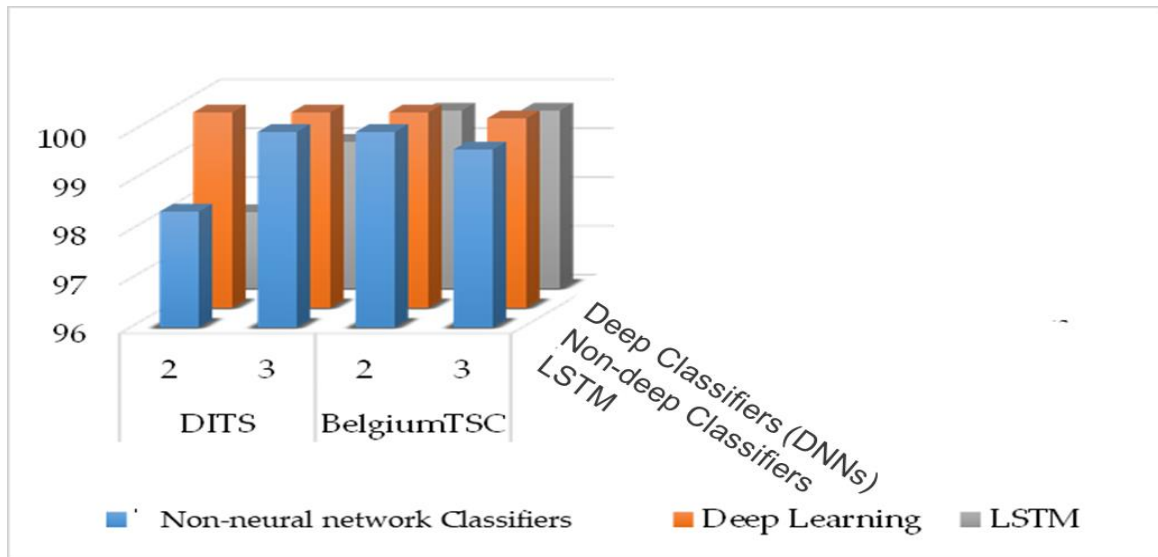


Figure 12: Highest accuracy achieved by LSTM, stacker with non-deep base-level, and stacker with deep base-level on BelgiumTSC and DITS.

Non-deep classifiers, such as *KNN*, SVM, AdaboostM2, and Random Forests achieved a perfect classification of each image contained in the GTRSB dataset. Moreover, we observed how combining deep features descriptor {AFeat \cup RFeat} allowed non-deep classifiers to reach the highest accuracy in any of the three datasets, achieving 100%, 95.94%, 99.12% on the GTSRB, DITS and BelgiumTSC datasets, respectively. On the other hand, DNNs outperform non-deep classifiers on the DITS and BelgiumTSC datasets but still cannot reach a perfect classification accuracy.

Noticeably, stacking meta-learners that take advantage of sliding windows achieve perfect classification accuracy on all three datasets when using base-level DNNs and *KNN* as meta-level classifiers. These results show that orchestrating sliding windows critically increases the classification performance compared to single image classifiers. Differently, LSTM networks achieve 97.56% and 99% of accuracy on the DITS dataset for a sliding window of size 2 or 3, respectively, which is better than single image classifier performance, but still inferior with respect to stacking meta-learners.

Figure 12 compares the accuracy achieved by stacking meta-learners and LSTM networks by means of a bar chart. Base-level non-deep classifiers with meta learners achieved 98.37% and 100% accuracy on the DITS dataset considering a sliding window of two and three inputs, respectively, which is slightly higher than the LSTM scores. A similar trend can be observed

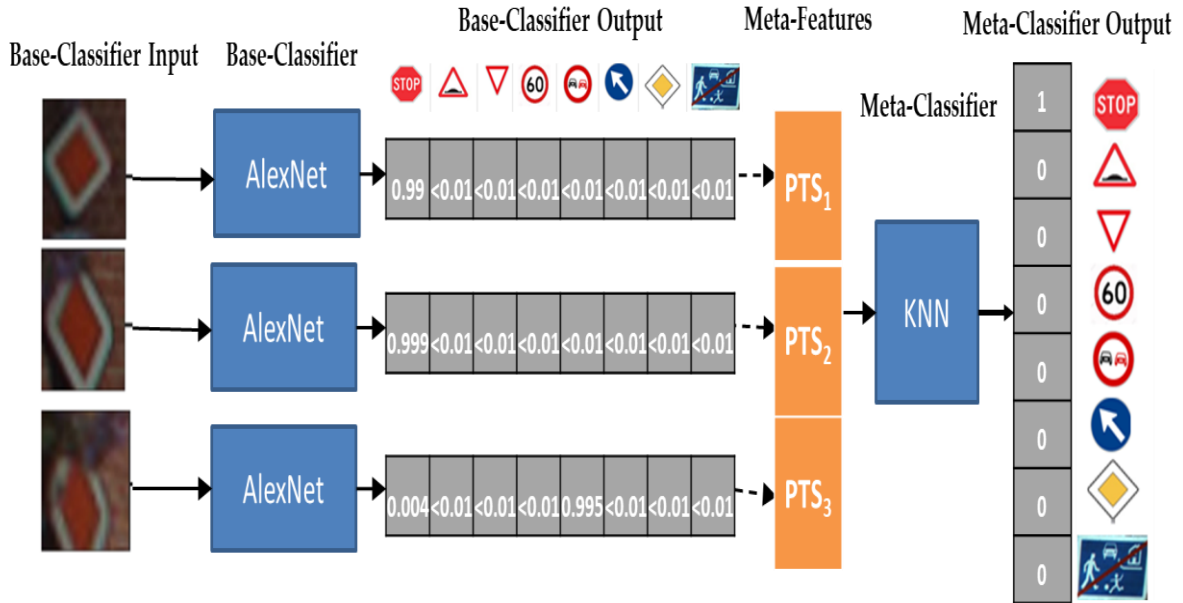


Figure 13: Instantiation of the stacking-meta learner with AlexNet base-learner and SVM meta-level learner, managing a sliding window of size 3 for BelgiumTSC. The three images we use as input DNNs describe a Diamond sign (Category 7) which is misclassified using all three images.

for the BelgiumTSC, while the GTSRB scores are not reported in the chart as it does not require sliding windows to achieve perfect accuracy.

4.5 In-Depth View of BelgiumTSC

Similarly, to the GTSRB and DITS, we observed perfect classification by using a stacker with base-level DNNs also with the BelgiumTSC dataset, which contains unordered sets of images rather than sequences. Consequently, our meta-learning strategy proves to be beneficial even if images in the sliding window are not time-ordered.

However, Figure 13 showed that a sliding window of three items performs poorly with respect to using only two items, which may seem counterintuitive. Figure 13 shows one of those cases in which using a window of two items is beneficial with respect to using three items. Figure 13 represents the process adopted for the classification of a Diamond traffic sign (Category 7) when using a window of three images. All the three images taken from different viewpoints are individually classified by the base-level classifier AlexNet, which returns the probabilities PTS of belonging to all classes (see Base-Classifiers output in the figure). These three probability vectors (which match the PTS_i in Section 4.1.1) are fed to the meta-level classifier to commit the final decision. We observe that PTS_1 and PTS_2 give almost a certain probability of belonging to class 7 (0.999), while PTS_3 gives a higher probability for class 1 (i.e., stop traffic sign). With those results, the SVM meta-learner decides that the traffic sign is

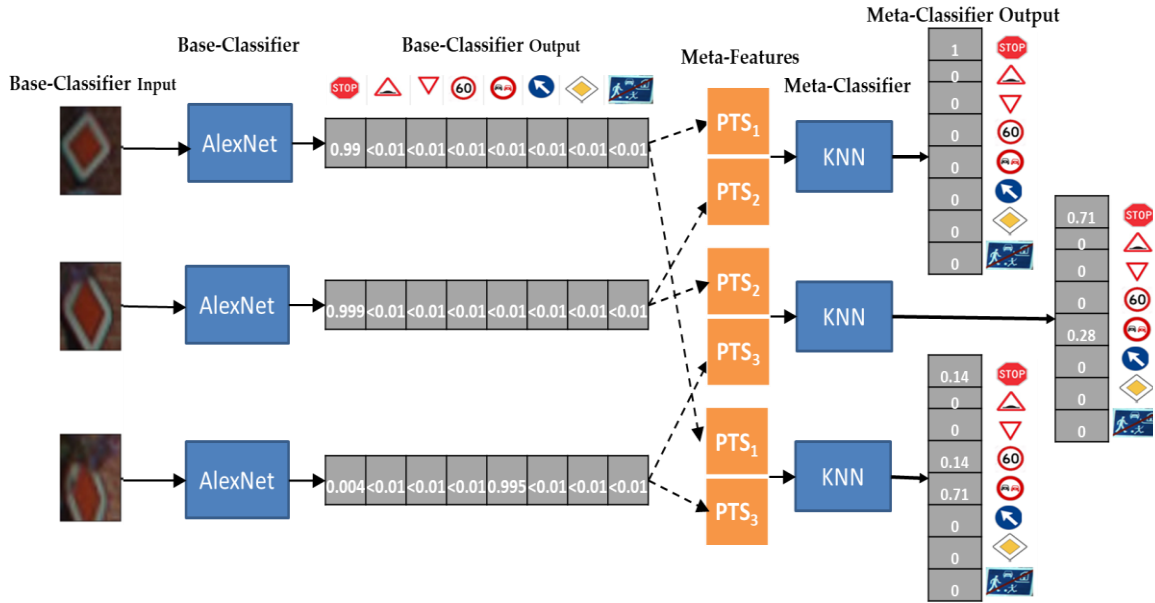


Figure 14: Instantiation of the Stacking-Meta learner with AlexNet Base-learner and SVM meta-level learner, managing a sliding window of size 2 for the BelgiumTSC. The three images we use as input describe a Diamond sign (Category 7) which is misclassified using all three images (Figure 13) but may be classified correctly by using a shorter window.

a stop sign, ending up with a misclassification. Clearly, the third image is taken from a different angle, has some blurring and makes the meta-learner lean towards a misclassification rather than helping. Similarly, to the GTSRB and DITS, we observed perfect classification by using a stacker with base-level DNNs also with the BelgiumTSC dataset, which contains unordered sets of images rather than sequences. Consequently, our meta-learning strategy proves to be beneficial even if images in the sliding window are not time-ordered. However, Figure 13 showed that a sliding window of three items performs poorly with respect to using only two items, which may seem counterintuitive. Figure 13 shows one of those cases in which using a window of two items is beneficial with respect to using three items. Figure 13 represents the process adopted for the classification of a Diamond traffic sign (Category 7) when using a window of three images. All the three images taken from different viewpoints are individually classified by the base-level classifier AlexNet, which returns the probabilities PTS of belonging to all classes (see BaseClassifier output in the figure). These three probability vectors (which match the PTS_i in Section 4.1.1) are fed to the meta-level classifier to commit the final decision. We observe that PTS₁ and PTS₂ give almost a certain probability of belonging to class 7 (0.999), while PTS₃ gives a higher probability for class 1 (i.e., stop traffic sign). With those results, the SVM meta-learner decides that the traffic sign is a stop sign, ending up with a misclassification. Clearly, the third image is taken from a different angle, has some blurring, and makes the meta-learner lean towards a misclassification rather than helping.

Table 12: Time required for (left) feature extraction, (middle) exercising individual classifiers, and (right) different TSR strategies, either sequential or parallel execution.

Feature Extractor	Time in Seconds	Individual Classifier	Time in Seconds	TSR Strategy	Average Time in Seconds	
	avg \pm st.dev		avg \pm st.dev		(Sequential)	(Parallel)
HOG	0.0204 \pm 0.0024	SVM	0.1344 \pm 0.0364	Single-image (AFeat \cup RFeat + SVM)	0.1974	0.1756
LBP	0.0196 \pm 0.0023	<i>KNN</i>	0.1205 \pm 0.0302	Single-image (InceptionV3)	1.4205	1.4205
AFeat	0.0218 \pm 0.0023	LDA	0.1034 \pm 0.0256	Stacking with WS = 2	0.4036	0.3636
RFeat	0.0412 \pm 0.0034	InceptionV3	1.4205 \pm 0.6613	(AFeat \cup RFeat + SVM + <i>KNN</i>)		
		MobileNet-V2	0.6391 \pm 0.2180	Stacking with WS = 2 (AlexNet + <i>KNN</i>)		
		AlexNet	0.3749 \pm 0.1407	Stacking with WS = 2 (InceptionV3 + <i>KNN</i>)	1.6085	1.6085

Instead, Figure 14 shows the process to classify the same inputs using a window of two items. When $\{PTS_1, PTS_2\}$ are provided as meta-features to a meta-level classifier, the final output shows a high likelihood of being a category 7 which is indeed a correct classification. Meanwhile, providing $\{PTS_2, PTS_3\}$ or $\{PTS_1, PTS_3\}$ as meta-features lead the stacker to misclassify the set of images as a stop sign (category 1): the predicted final output is class 6 which is a wrong prediction. This enforces the conjecture that in this case using the third image constitutes noise that causes misclassification.

4.6 Timing Analysis

This section expands on the time required for classification using the different setups in this chapter. Table 12 reports the average and standard deviation of time required for (i) feature extraction, (ii) single-image classification, and (iii) stacking meta-learning across test images of three datasets.

Starting from feature extraction on the left of the table, it turns out that the extraction of handcrafted features takes slightly less time compared to deep features. However, even extracting deep features through ResNet-18 from a single image does not require on average more than 0.04 s (roughly 40 ms). Instead, the time required for exercising single-image TSR classifiers varies a lot: non-deep classifiers need at most 200 ms to classify a given input set, whereas DNNs need more than half a second to classify an image with our hardware setup,

depending on the number of layers of DNNs. Indeed, the reader should note that whereas DNNs embed feature extraction through convolutional layers, non-deep classifiers have the prerequisite of feature extraction. In fact, on the right of Table 12, we show that a TSR system that relies on AFeat \cup RFeat features (i.e., most useful ones according to Table 7) provided to an SVM classifier takes on average 0.1974 s to classify an image: this includes feature extraction and classification itself. A perfect parallelization of the feature extractors cuts down this timing to 0.1756 and will be easily achievable on basic multi-core systems.

Table 12 also shows the time needed to perform other TSR strategies we discussed in this chapter. Particularly, the third to sixth line on the right of the table show the time needed to classify an image using a sliding window of two or three items with different base-levels and meta-level learners. The time required for base-level learning equals single-image classification: only the most recent image in the window is processed, whereas probabilities assigned by classifiers to older images are stored, and therefore, do not need to be re-computed again. The table reports on different base-learners but always uses *KNN* as the meta-level learner, as this was the classifier that allowed reaching high scores in Section 4.4.2. *KNN* takes on average 0.188 to classify a sliding window of two items, (i.e., two PTS vectors of 8/9 numbers each), and only slightly more time to process a sliding window of three items.

Overall, we can observe how most TSR systems that embed sliding windows are able to classify a new image in less than a second, whereas heavier DNNs make classification time lean towards two seconds. We believe that such timing performance albeit slower than using single-image classifiers is still efficient enough to be installed on a vehicle, which only rarely samples more than an image per second for TSR tasks. Nevertheless, using more efficient hardware, especially GPUs, could help in reducing, even more, the time required for classification.

4.7 Comparison to the State of the Art TSR

Ultimately, we recap the accuracy scores achieved by studies we already referred to as related works in Section 2.3, to compare their scores with ours. Therefore, Table 13 summarizes those studies, the datasets they used, and the accuracy they achieved. At a first glance, those studies conclude that their single-image classifiers are often far from perfect classification. In fact, even in this study, we observed that single-image TSR in the BelgiumTSC and DITS datasets cannot reach perfect accuracy (i.e., second-last row in the table). Unfortunately, promising studies [64], [65], which describe multi-image classifiers, do

Table 13: Comparison with state of the art approaches. Gray area in the table shows our result approach using single frame and sliding windows result for 3 and 9 traffic sign categories.

Studies	Sequences of images	Achieved Accuracy (%)		
		GTSRB	BelgiumTSC	DITS
Stallkamp et al. [31]	No	98.98		
Agrawal et al. [45]	No	*77.43		
Youssef et al. [33]	No	95.00		98.20
Mathias et al. [1]	No		97.83	
Huang et al. [44]	No	*95.56		
Lin et al. [51]	No		*99.18	
Li et al. [28]	No	97.40	98.10	
Li and Wang [3]	No	99.66		
Zeng et al. [83]	No		95.40	
Single frame (3 categories)	No	100.00	99.80	99.31
Single frame (9 categories)	No	100.00	99.72	96.03
Sliding Windows (9 categories)	Yes	100.00	100.00	100.00

not rely on our datasets, and therefore, we cannot directly compare them. To summarize, our experiment ended up achieving perfect classification on all datasets thanks to sliding windows (see last row of Table 13), dramatically improving existing studies on those datasets, for which perfect accuracy was hardly achieved by existing studies.

4.8 Lessons Learned

This section highlights the main findings and lessons learned from this chapter of the thesis. We observed that classifying images in the DITS dataset is harder than classifying the BelgiumTSC and GTSRB datasets as both base-level non-deep classifiers and DNNs performances are low comparatively. This is mostly due to the amount of training images and their quality, which is higher in the GTSRB compared to the other two datasets. Furthermore, combining feature descriptors allows for improving classification performance. Particularly, we found that the $\{AFeat \cup RFeat\}$ descriptor allows non-deep classifiers to maximize accuracy. Single-image non-deep classifiers achieved perfect classification on the GTSRB dataset, while on the BelgiumTSC and DITS they show a non-zero amount of misclassifications. To the best of our knowledge, this result is due to the number of training samples, which is higher in the GTSRB with respect to the BelgiumTSC and DITS, and image quality, which again is better for the GTSRB.

On the other hand, we achieved 100% accuracy by adopting a sliding window based TSR strategy on all three considered datasets. There is no clear benefit in adopting DNNs over non-deep classifiers for single-image classification as they show similar accuracy scores.

Additionally, both are outperformed, when using sliding windows for TSR. LSTM networks often, but not always, outperform single-image classifiers but show lower accuracy than stacking meta-learners in orchestrating sliding windows. A stacking meta-learner with base-level DNNs and *KNN* as meta-level classifier can perfectly classify traffic signs on all three datasets with any window size $WS \geq 2$. For datasets that contain sequences (time-series) of images, enlarging the sliding window never decreases accuracy and, in most cases, raises the number of correct classifications. DNNs require more time compared to non-deep classifiers, especially because there are many layers, e.g., InceptionV3. Sliding windows-based classification takes more time compared to single-image classifiers but has remarkably higher classification performance across all three datasets. Overall, adopting classifiers that use a sliding window rather than a single-image classifier allows for reducing misclassifications, consequently raising accuracy. To wrap up this chapter, our experimental campaign showed how the adoption of a stacking meta-learner in conjunction with sliding windows allows for achieving perfect classification on the public GTSRB, BelgiumTSC, and DITS datasets.

5. ROBUST TRAFFIC SIGN RECOGNITION AGAINST VISUAL CAMERA FAILURES

Due to distraction, fatigue, or adverse operating conditions, human drivers can miss or misinterpret an important traffic sign [11], [86] potentially leading to dangerous situations. Unfortunately, the adverse environmental conditions, or the malfunctions of the visual camera, may produce low-quality images that may negatively impact the performance of classifiers. Examples include, but are not limited to: occlusions, shadows, defects of the visual camera lens, changes in environmental light, raindrops on the visual camera lens, out-of-focus, flare [88], [89]. Therefore, to guarantee safety of the driving task, it is necessary to study the robustness of TSR systems against the aforementioned threats, and develop solutions to tolerate them [90], [91]. This chapter accomplishes this task, first reviewing the challenges and the state of the art, and then proposing and evaluating alternative solutions.

The chapter develops as follow. In our study we corrupt images by applying visual camera failures that are successively fed to single-image classifiers and sliding windows classifiers described in detail in Chapter 4. Images are taken from three publicly available benchmark datasets described in Section 2.2.7. These datasets include sequences of images and consequently allow applying both classification strategies. We simulated 13 visual camera failures, each one actionable with different parameters, obtaining a total of 103 failures configuration to be individually injected into each image of the three datasets.

The produced data allows examining the robustness of three DNNs: AlexNet [34], InceptionV3 [40], and MobileNet-V2 [41]. We apply them independently to perform classification, and we also run them in parallel; in this second case, classification is performed by a stacking meta-learner which is fed with the outputs of the classifiers. Further, we apply two different classification strategies: single image classification and sliding window classification. Our results indicate that the occurrence of visual camera failures degrades classification performance, especially for single-image classifiers. Instead, approaches based on the sliding window are significantly more robust.

Further, we dig into the results using LIME [92], a toolbox for explainable Artificial Intelligence (AI). Explainable AI allows understanding how a classifier uses the input image to derive its output: we use LIME to explain why the injection of the visual camera failures

alters the behaviour of the classifiers, and why certain classifiers are more robust than others against visual camera failures.

5.1 TSR Strategies Under Consideration

This section explains in detail the TSR classification strategies we consider: single-image classification and sliding windows classification.

5.1.1 Single-image Classification

A single image is usually classified using one classifier. In addition, it is possible to combine multiple classifiers in a two-step process, relying on a stacking meta-learner to decide on the class [71] (Figure 15). In stacking, first, the image is fed to multiple base-level classifiers, then the individual predictions from the classifiers are processed by a classifier at a meta-level; in other words, the predictions of parallel classifiers are input meta-features to a meta-level classifier that provides the final prediction.

Figure 15 exemplifies this process considering as input an image of a STOP traffic sign and three base-level classifiers: AlexNet, InceptionV3, and MobileNet-V2. Each of the three classifiers outputs an array of probabilities $PTS_k = \{pts_{ik}, 1 \leq i \leq n\}$ where n is the number of possible classes, and k is the k -th base-level classifier (in Figure 15, we use $1 \leq k \leq 3$). Each pts_{ik} describes the probability that the input image contains a traffic sign of the i -th class, according to the k -th base-level classifier. All $pts_{ik} \in PTS_k$ sum up to 1. The three classifiers process the input image independently and provide their own probabilities PTS_1 , PTS_2 , and PTS_3 for that input. These probabilities are then aggregated into a unique meta-feature set of $3n$ features and delivered as input to the meta-level classifier, which provides its own PTS_{final}

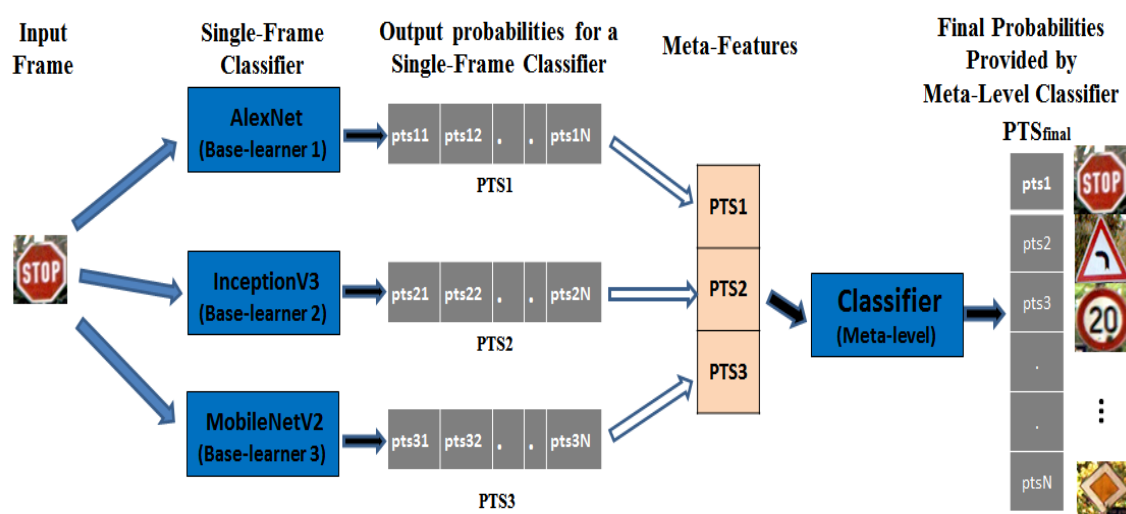


Figure 15: A TSR system that uses multiple (three) single-image classifiers with stacking.

probabilities that are used for TSR. Noticeably, the meta-level classifier plays the role of an adjudicator rather than working as a classifier itself.

In this chapter, we consider the following three classification strategies:

- *Single-image classifier on a single image (1SFC, One Single Image Classifier).* In this case, we are using just one classifier to perform classification.
- *Two single-image classifiers on a single image (2SFC, Two Single Image Classifiers).* This approach relies on stacking the predictions of the two base-level classifiers, relying on a stacking meta-level classifier.
- *Three single-image classifiers on a single image (3SFC, Three Single Image Classifiers).* This approach corresponds to Figure 15. It relies on stacking the predictions of the three base-level classifiers, using the stacking meta-level classifier.

5.1.2 Classification with a Sliding Window

In addition, we build classifiers that can process multiple images in a sliding window [87]. The structure of this classifier is described in Figure 16. Sliding windows contain multiple images captured at different time instants or from multiple visual cameras at the same time. Therefore, we need to slightly adapt the classifier from Section 5.1.1 to process multiple images in parallel.

For example, Figure 16 shows a TSR system that uses a sliding window of three images and that is composed of two single-image classifiers. First, each of the two single-image

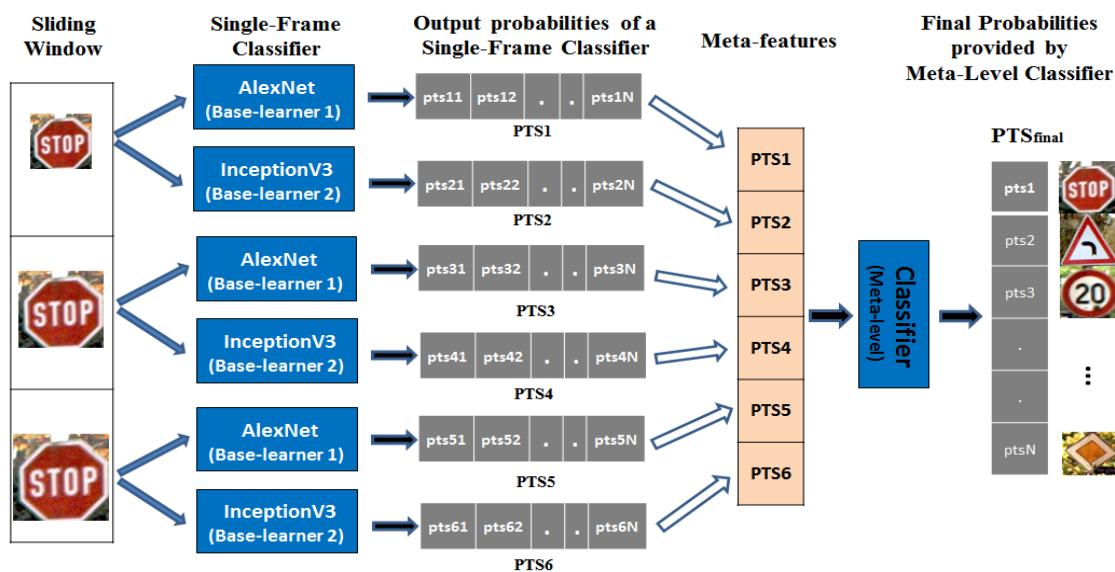


Figure 16: A TSR which uses sliding windows of three images (on the left) with two base-classifiers (AlexNet, InceptionV3) and a meta-level classifier that produces the final classification result PTS_{final} .

classifiers processes each of the three images individually, building six PTS_k arrays of probabilities $1 \leq k \leq 6$. These arrays constitute the meta-feature set that is provided to the stacking classifier. Finally, the stacking classifier produces the classification result PTS_{final} , which is the output of the TSR system.

In this chapter, we consider the following classification strategies based on a sliding window:

- Sliding window is applied using only one classifier (W1C, Sliding Window with One Classifier).
- Sliding window is applied using two classifiers (W2C, Sliding Window with Two Classifiers). This corresponds to Figure 16. In this case, all images in the sliding window are individually classified by two base-level classifiers; the predictions of each image are input meta-features to the meta-level classifier.
- Sliding window is applied using three classifiers (W3C, Sliding Window with Three Classifiers). All images in the sliding window are individually classified by three base-level classifiers; the predictions of each image are input meta-features to the meta-level classifier.

5.2 Approach to Robustness Evaluation

We detail the methodology we used to evaluate the robustness of the TSR strategies previously described, when visual camera failures occur. We describe the datasets of traffic signs, the failure injection strategies, and the single-image and sliding-window classifiers that perform TSR on clean and injected images.

5.2.1 Methodology

First, all classifiers described in Section 5.2.2, organized in groups 1SFC, 2SFC, 3SFC, W1C, W2C, W3C, are trained using the traffic sign datasets BelgiumTSC, GTSRB, and DITS described in Section 2.2.7. Then, the stacking meta-learners of Section 5.2.3 are trained as well. This creates all the classification strategies 1SFC, 2SFC, 3SFC, W1C, W2C, W3C trained on the traffic sign datasets.

In the next phase, the visual camera failures described in Section 5.2.4 are injected into each dataset individually, to create datasets of altered images. We will call these datasets as injected datasets, opposed to the clean datasets, and the altered images as injected images, opposed to the clean images. This creates a set of injected datasets, each with different failures. Noteworthy, some failures have multiple possible configuration (e.g., the amount of blur that

is applied on the images): we create injected datasets of GTSRB, BelgiumTSC and DITS for each configuration. This leads to a total of 103 configurations, each one repeated on the three datasets. These datasets are provided as test inputs to the 1SFC, 2SFC, 3SFC, W1C, W2C, W3C. Finally, we measure the robustness of different TSR approaches against each visual camera failure. The robustness is described using the metrics discussed in Section 5.2.5.

5.2.2 Base-level Classifiers

This study utilizes three DNN classifiers of Section 2.2.4. We adapt them for TSR using transfer learning from models trained on ImageNet such as MobileNet-V2 (MN2) [41], AlexNet (AN) [34], and InceptionV3 (IC3) [40]. Combination of classifiers (2SFC, 3SFC, W2C, W3C) includes all the possible pairs and triples from the three classifiers above.

5.2.3 Meta-level Classifiers

This section presents six non-deep classifiers of Section 2.2.3 that suit the role of meta-level classifiers for stacking such as kth-Nearest Neighbours (*KNN*) [13], Support Vector Machines (SVM) [14], Decision Tree (DT) [93], Linear Discriminant Analysis (LDA), AdaBoostM2 (ABM2) [39], and Random Forests (RF) [37].

5.2.4 Strategies to Inject Visual Camera Failures

This section provides a brief overview of the potential failures of a visual camera installed on vehicles. We considered a total of 13 failures, described in what follows. Failures may have multiple configurations; each failure and its configurations are applied on the three datasets. In total, this leads to 103 copies of each dataset, where all images are modified according to the specified configuration. The source code to reproduce the injection of failures is based on the library available at [94], which was customized where needed.

Banding. An image with banding has many horizontal and/or vertical lines in the background. Figure 17a shows the clean image, while Figure 17b shows the effect of applying banding failures on the image.

Blurred. Blurred occurs when the captured image is not properly focused by the visual camera lens (see Figure 17c). Overall, we used 11 different configurations to produce blurred images with different amounts of blurriness.

Brightness. This failure alters the brightness of the produced image (Figure 17d), ranging from no brightness (black image) to full brightness (white image) depending on malfunctions of the lens shutter, diaphragm, or iris. The same visual effect on the output image can happen when

the light enters the visual camera with a narrow angle (e.g., in case the sun is in front). We simulated 8 levels of brightness from a very dark image to an almost white one.

No Chromatic Aberration Correction (NCAC). This type of failure occurs when the ISP fails: halos appear on the edges and corners of the image. The resulting image (Figure 17e) shows the blurred effect on the outer edges.

Condensation (COND). Condensation on the lens happens when humidity levels are high, or because of rapid temperature changes. To inject a condensation failure, we overlapped condensation images of [95] to each of the images in all datasets, as in Figure 17f. We utilized 3 different overlaying images to produce three different effects of condensation failures.

No Bayer Filter (NOBF). Should the Bayer filter [96] do not work properly, the acquired image will result in a colorless image as in Figure 17g. In our experiments, we replicated this effect by changing the RGB (Red, Green, Blue) channels to convert an image to grayscale.

Dirt. This visual camera failure occurs when there is dirt on the internal or external lens. Figure 17h shows a dirty image generated after overlapping the dirt images of [95] to a traffic sign image. This study uses 36 different dirt images.

Ice. This type of failure occurs when the temperature of the environment drops below freezing, affecting image quality. Figure 17i shows the ice effect on an acquired image by using one out of the four different ice images of [95] that we overlapped to traffic sign images.

No Demosaicing (NODEMOS). The raw image may be not processed by ISP for demosaicing: this creates the output images by interpolating the mosaic of RGB colours produced by the image sensors. As a result, the image remains pixelated (i.e., each pixel contains either red, green, or blue color channel values) as shown in Figure 17j.

No Noise Reduction (NNR). When images are captured by a visual camera, the ISP is responsible for removing noise. However, this process may fail: to simulate this event, we used 10 different configurations of speckle noise: an example is shown in Figure 17k.

Rain. During rainy weather, the external lens of the visual camera may have small drops of water that degrade the quality of the acquired image. We simulate this event by overlapping 5 different rain images from [95] to traffic sign images; an example is in Figure 17l.

Dead Pixels. Certain failures of the image sensor may produce an output image with dark spots; visually, the effect is analogous to visualizing dead pixels. In this study we simulate different configurations, namely i) an array of vertical pixels is set to black, ii) horizontal and vertical

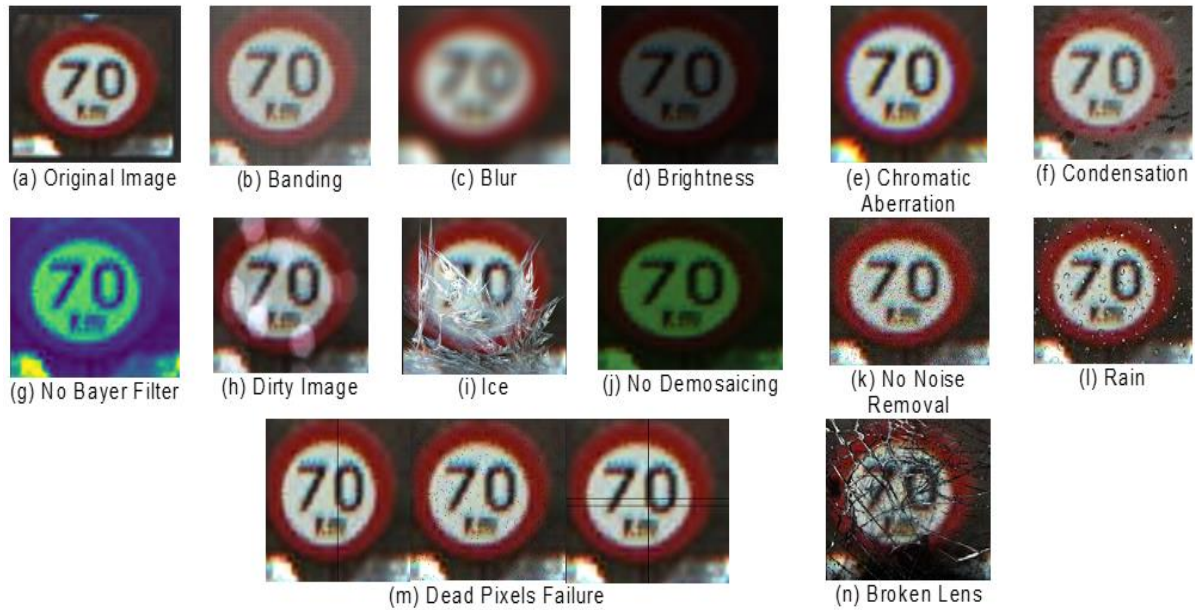


Figure 17: Injection of 13 different visual camera failures to a sample traffic sign. (best viewed in color)

arrays are set to black, or iii) sets of [50, 200, 500, 1000] pixels, randomly selected, are turned black. Figure 17m shows different failure effects when they are applied on the clean image.

Broken Lens. This failure occurs when one or more lenses of the visual camera break because of many reasons e.g., debris that crashes against the lens. This may make scratches visible in the produced image. We simulated a broken lens with 15 different overlaying images from [95]. An example is in Figure 17n.

5.2.5 Performance Metrics

To evaluate results, we rely on the performance metric accuracy [77], [78] as described in Section 2.2.8, which consider each misclassification as equally harmful. Further, we elaborate on accuracy by defining a metrics that measures the accuracy reduction. We define accuracy drop the difference between the accuracy obtained on the injected datasets and on the clean datasets. The accuracy drop is computed for each failure. Some of the failures have multiple configurations: in this case, for such failures we compute the minimum accuracy drop and the maximum accuracy drop.

5.3 Experimental Results

We organize results as follows. Section 5.3.1 discusses the results achieved using the clean datasets GTSRB, BelgiumTSC, and DITS for single image classifiers (1SFC, 2SFC, and 3SFC) and sliding window classifiers (W1C, W2C, and W3C). The successive sections instead

consider the injected datasets: Section 5.3.2 describes the results of 1SFC, Section 5.3.3 of 2SFC and 3SFC, Section 5.3.4 of W1C, and Section 5.3.5 of W2C and W3C.

5.3.1 Results on the Clean Datasets

Figure 18 highlights the accuracy achieved on three clean datasets, using the different classification strategies. With 1SFC, we reach 99.35%, 99.72%, and 96.03% accuracy on GTSRB, BelgiumTSC, and DITS, respectively. The usage of 2SFC improved the classification accuracy to 100%, 99.88%, and 99.48% on GTSRB, BelgiumTSC, and DITS, respectively. Classification of traffic signs based on sliding windows approach achieves an accuracy of 100% across GTSRB and DITS for W1C, W2C, and W3C, while 100% accuracy is achieved on BelgiumTSC using W1C and W2C.

W1C, W2C, or W3C achieves 100% accuracy on the GTSRB dataset. On DITS, strategies W1C, W2C and W3C achieves 100% accuracy, with the exception of W1C with AN. On BelgiumTSC, we get 100% classification accuracy using AN (W1C) and AN+MN2 classifiers (W2C). Figure 18 shows that 2SFC and 3SFC significantly reduce the misclassification across all datasets with respect to 1SFC, while W1C, W2C, or W3C achieves 100% classification accuracy across all three datasets. This proves and quantifies the efficacy of the sliding window approach for classification in nominal operating conditions (clean images).

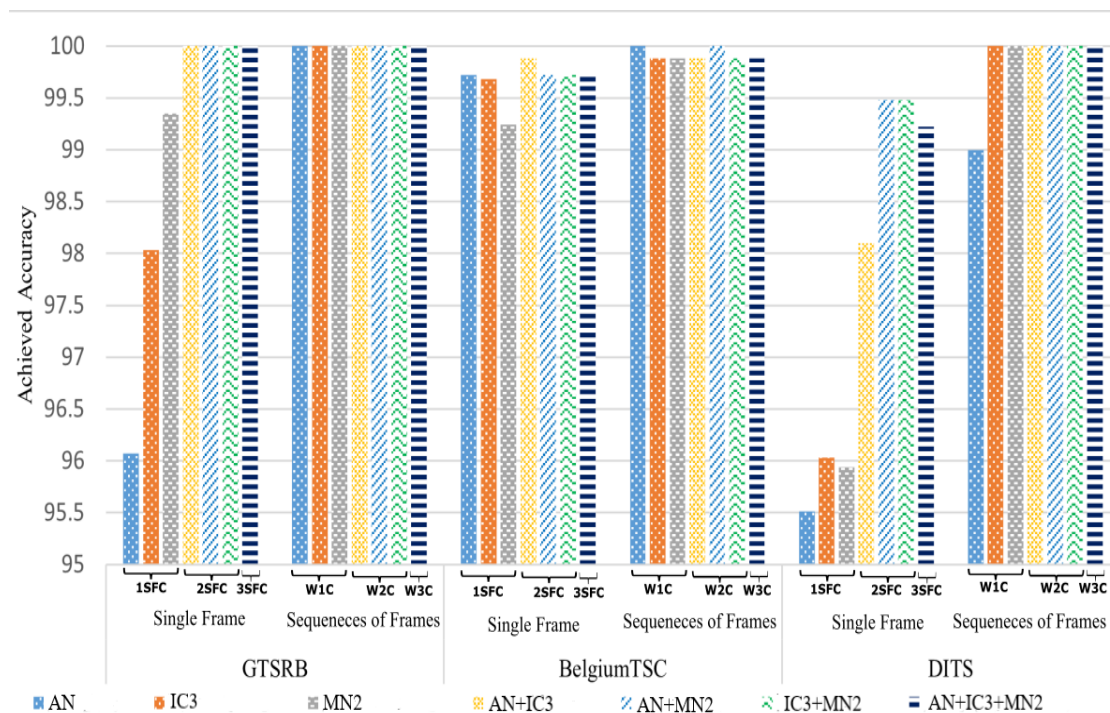


Figure 18: Highest accuracy achieved on the clean datasets GTSRB, BelgiumTSC, DITS.

5.3.2 1SFC on the Injected Datasets

We analyse the accuracy drop obtained by 1SFC on the clean datasets and the injected datasets (Table 14). The 1SFC that obtains the best accuracy is reported: for such classifier, we report the maximum and minimum difference in accuracy (accuracy drop) with respect to the case on the clean datasets.

Experimental results show that there is a significant rise in misclassifications. For example, with banding, under different configurations the maximum and minimum accuracy drop on GTSRB is [-3.09, -4.69], where AN performs better than MN2 and IC3. Experimental results indicate that 1SFC MN2 on GTSRB, and IC3 on DITS and BelgiumTSC, are overall robust against NCAC failure; all the other failures have a more relevant impact on the accuracy of 1SFC. The accuracy drop is more relevant for some failures, which are highlighted in bold in Table 14; these are broken lens, ice, brightness, and NNR. In particular, there are specific cases of brightness failure that lead to a significant accuracy drop. For example, when brightness is very low (images are very dark) or very high (images are almost white), even humans may not recognize traffic signs. Another important observation is that for brightness, broken lens, ice,

Table 14: Minimum and maximum accuracy drop of 1SFC when it is exercised on the injected datasets and the clean datasets.

		GTSRB		DITS		BelgiumTSC	
Original Dataset (Max Achieved Acc.)		99.35%		96.03%		99.72%	
Fault Type	Config.	Accuracy drop	Base Classifier	Accuracy drop	Base Classifier	Accuracy drop	Base Classifier
Banding	3	[-3.09, -4.69]	AN	[-0.86, -5.87]	IC3	[-0.12, -0.60]	AN
Blurred	11	[-1.00, -4.76]	MN2	[-12.34, -18.03]	IC3	[-2.75, -13.49]	AN
Brightness	8	[-3.41, -40.15]	AN	[-1.90, -60.14]	AN	[-0.16, -67.30]	AN
Broken Lens	15	[-15.72, -52.48]	AN	[-10.78, -35.37]	AN	[-1.60, -53.17]	AN
NCAC	1	-0.02	MN2	-1.12	IC3	-0.08	IC3
COND	3	[-0.06, -1.91]	MN2	[-0.52, -2.85]	IC3	[-0.20, -0.92]	AN
Dead Pixels	6	[-0.01, -11.82]	MN2	[-0.78, -11.13]	MN2	[-0.20, -6.03]	AN
Dirt	36	[0.05, -2.31]	MN2	[-0.09, -11.56]	IC3	[0.00, -1.92]	IC3
NOBF	1	-13.22	MN2	-14.84	IC3	-39.92	MN2
Ice	4	[-3.32, -26.45]	AN	[-2.24, -59.88]	AN	[-0.88, -39.40]	AN
NDEMOS	1	-3.08	AN	-18.64	IC3	-2.67	IC3
NNR	9	[-3.42, -57.40]	AN	[-5.69, -64.36]	AN	[-0.20, -76.97]	AN
Rain	5	[-3.53, 12.98]	AN	[-2.16, -17.00]	AN	[-0.56, -13.33]	AN

NNR, or rain failures, AN always achieves the best accuracy. IC3 performs worse compared to the other 1SFC i.e., AN and MN2 on GTSRB. Overall, MN2 and AN perform better on GTSRB, while AN and IC3 performance are better on the other two datasets (but still, the performance of 1SFC is far from perfect).

5.3.3 2SFC and 3SFC on the Injected Datasets

Table 15 discusses the accuracy drop achieved by 2SFC and 3SFC on the clean dataset and the injected datasets. Experimental results show that 2SFC and 3SFC reduce misclassifications against each failure except few cases, where accuracy is less than in the 1SFC cases. These last ones are in bold in Table 15, and they are brightness, broken lens, NNR, and ice failures.

With respect to 1SFC and 2SFC, on GTSRB and DITS, 3SFC achieved better accuracy on the majority of failures. Instead, on BelgiumTSC, the 2SFC AN+IC3 is the most robust against the majority of failures. Overall, utilizing 2SFC and 3SFC improves the accuracy. The meta-level classifier *KNN* performs better on all three datasets. However, the effect of failures is still not fully mitigated with the 2SFC and 3SFC: there is no case that achieves 100% accuracy against any failure, and in some cases the combination of classifiers is worse than using just a single classifier.

Table 15: Minimum and maximum accuracy drop of 2SFC and 3SFC on the injected datasets with respect to the clean datasets.

Original Dataset (Max Achieved Acc.)	GTSRB			DITS			BelgiumTSC		
	100%			100%			100%		
Fault Type (Config.)	Accuracy drop	Base Classifiers	Meta Classifier	Accuracy drop	Base Classifiers	Meta Classifier	Accuracy drop	Base Classifiers	Meta Classifier
Banding (3)	[-0.23, -2.85]	AN+MN2	<i>KNN</i> , DT	[-1.29, -4.31]	AN+IC3+MN2	<i>KNN</i>	[-0.04, -0.56]	AN+IC3	<i>KNN</i>
Blurred (11)	[-1.52, -5.92]	AN+IC3+MN2	LDA	[-7.76, -28.47]	MN2+IC3	RF	[-1.60, -14.57]	AN+IC3	<i>KNN</i>
Brightness (8)	[-0.10, -35.69]	AN+MN2	ABM2, RF	[-1.64, -63.33]	AN+IC3+MN2	<i>KNN</i>	[-0.28, -66.67]	AN+IC3+	RF, <i>KNN</i> , MN2 ABM2
Broken Lens (15)	[-2.20, -46.17]	AN+IC3+MN2	RF, <i>KNN</i>	[-4.31, -28.04]	AN+IC3+MN2	<i>KNN</i>	[-2.24, -51.34]	AN+IC3	<i>KNN</i> , RF
NCAC (1)	-0.08	AN+IC3+MN2	ABM2	-0.69	AN+IC3+MN2	<i>KNN</i>	-0.08	AN+IC3	<i>KNN</i>
COND (3)	[-0.10, -0.36]	AN+IC3+MN2	ABM2	[-0.86, -3.10]	AN+IC3+MN2	<i>KNN</i>	[-0.12, -1.44]	AN+IC3	<i>KNN</i>
Dead Pixels (6)	[-0.07, -1.99]	AN+MN2	ABM2, RF	[-2.41, -10.52]	AN+MN2	RF, <i>KNN</i>	[-0.36, -5.91]	AN+MN2	ABM2, <i>KNN</i>
Dirt (36)	[-0.02, -0.73]	AN+IC3+MN2	ABM2, RF	[-1.46, -9.75]	AN+IC3	<i>KNN</i>	[-0.04, -2.43]	AN+IC3	<i>KNN</i> , RF
NOBF (1)	-7.34	AN+IC3+MN2	RF	-11.13	MN2+IC3	<i>KNN</i>	-38.76	AN+MN2	<i>KNN</i>
Ice (4)	[-0.21, -22.43]	AN+MN2	ABM2, <i>KNN</i>	[-0.69, -59.88]	AN+IC3+MN2	<i>KNN</i>	[-0.68, -44.55]	AN+IC3	<i>KNN</i>
NDEMOS (1)	-1.01	AN+IC3+MN2	<i>KNN</i>	-18.03	MN2+IC3	<i>KNN</i>	-1.36	MN2+IC3	<i>KNN</i>
NNR (9)	[-0.96, -52.30]	AN+IC3+MN2	RF, DT	[-7.33, -67.64]	AN+IC3	<i>KNN</i> , RF	[-0.60, -75.45]	AN+MN2	<i>KNN</i>
Rain (5)	[-0.10, -4.80]	AN+MN2	ABM2, RF	[-0.43, -15.10]	AN+IC3+MN2	<i>KNN</i>	[-0.20, -20.96]	AN+IC3	<i>KNN</i>

5.3.4 W1C on the Injected Datasets

Table 16 reports the accuracy drop achieved by W1C on the clean datasets versus the injected datasets. Classification performed on subsequent images within a sliding window of size 3 improves the classification performance with respect to 1SFC. For example, W1C achieved 100% accuracy for all three datasets under the NCAC failure, where the accuracy difference is zero; this is marked in bold in Table 16. Furthermore, on the GTSRB dataset we achieved 100% accuracy against blurred, NCAC, COND, dead pixels, dirt and NDEMOS.

Another important observation is that, for the majority of failures, W1C (with AN as base classifier and *KNN* meta-level classifier) achieves the highest accuracy. W1C improves accuracy with respect to 1SFC, 2SFC, and 3SFC, with few exceptions, such as broken lens failure on DITS and NDEMOS failure on BelgiumTSC. The reason behind the inferior performance of W1C in this case is most likely due to an image which is misclassified by the single classifier: 2SFC and 3SFC may classify the image correctly, because there is more than one base classifier executing in parallel. Similarly, to the previous cases, there is no significant improvement against some failures such as broken lens, brightness, ice, and NNR failures.

5.3.5 W2C and W3C on the Injected Datasets

Table 17 highlights the degradation intervals of accuracy achieved by W2C and W3C on the clean dataset versus the injected datasets. Accuracy against failures blurred, NCAC, COND, dead pixels, dirt, and NDEMOS is 100% as already achieved for W1C, and it is not further reported in Table 17. The bold values in Table 17 represent the failures where the performance of W2C and W3C improves with respect to W1C. For the remaining failures, W1C performs better. The possible reason is that an image is classified correctly by a base classifier, but when the same image is classified by multiple base classifiers, there are chances that some misclassify the image. This way, when the various meta-features for W2C and W3C are provided to meta-level classifiers, this may result in a misclassification.

Overall, few failures such as ice, broken lens, brightness, and NNR are still very hard to be classified accurately. Overall, the meta-level classifier *KNN* is the one that performs better in the majority of cases.

Table 16: Minimum and maximum accuracy drop of WIC on the injected datasets with respect to the original datasets.

Original Dataset (Max Achieved Acc.)	GTSRB			DITS			BelgiumTSC		
	100%			100%			100%		
Fault Type (Config.)	Accuracy drop	Base Classifier	Meta Classifier	Accuracy drop	Base Classifier	Meta Classifier	Accuracy drop	Base Classifier	Meta Classifier
Banding (3)	[-0.24, -0.72]	AN	<i>KNN</i> , RF	-1.00	IC3	<i>KNN</i> , SVM, RF, DT, ABM2	[0.00, -0.36]	AN	<i>KNN</i>
Blurred (11)	0.00	MN2	<i>KNN</i> , LDA, SVM, ABM2	[-8.00,-21.00]	AN	<i>KNN</i> , DT, LDA	[-1.56,-12.10]	AN	<i>KNN</i>
Brightness (8)	[0.00, -6.93]	AN	<i>KNN</i> , DT, ABM2, RF	[-1.00,-51.00]	AN	SVM, LDA	[0.00, -62.04]	AN	<i>KNN</i>
Broken Lens (15)	[-1.20,-11.22]	AN	DT, LDA	[-8.00,-21.96]	AN	SVM, LDA, ABM2	[-0.72,-46.95]	AN	<i>KNN</i>
NCAC (1)	0.00	AN	<i>KNN</i> , DT, ABM2, RF	0.00	IC3	<i>KNN</i>	0.00	AN	RF
COND (3)	0.00	AN	<i>KNN</i> , DT, ABM2, RF	[0.00, -2.00]	IC3	<i>KNN</i> , ABM2	[-0.12, -0.60]	AN	<i>KNN</i>
Dead Pixels (6)	0.00	AN	<i>KNN</i> , DT, ABM2, RF	[0.00, -7.00]	MN2	<i>KNN</i>	[0.00, -3.24]	AN	<i>KNN</i>
Dirt (36)	0.00	AN	<i>KNN</i> , DT, ABM2, RF	[0.00, -5.00]	IC3	<i>KNN</i> , SVM, ABM2	[0.00, -0.96]	IC3	<i>KNN</i> , RF
NOBF (1)	-0.72	MN2	<i>KNN</i> , ABM2, DT	-3.00	IC3	<i>KNN</i>	-35.33	MN2	RF
Ice (4)	[0.00, -2.39]	AN	<i>KNN</i> , DT, ABM2, RF	[-1.00,-40.66]	AN	<i>KNN</i> , SVM, RF, DT, ABM2,	[-0.36,-37.49]	AN	<i>KNN</i>
NDEMOS (1)	0.00	AN	LDA	-3.00	IC3	<i>KNN</i>	-1.80	IC3	<i>KNN</i>
NNR (9)	[0.00, -37.71]	AN	<i>KNN</i> , ABM2, RF, DT	[-4.00,-60.00]	AN	<i>KNN</i> , SVM, DT	[-0.12,-77.25]	AN	<i>KNN</i>
Rain (5)	[0.00, -0.24]	AN	<i>KNN</i> , ABM2, RF, DT	[-1.00,-12.00]	AN	SVM, ABM2	[-0.24,-11.38]	AN	<i>KNN</i>

Table 17: Minimum and maximum accuracy drop of W2C and W3C on the injected datasets with respect to the clean datasets.

Original Dataset (Max Achieved Acc.)	GTSRB			DITS			BelgiumTSC		
	100%			100%			100%		
Fault Type (Config.)	Accuracy drop	Base Classifiers	Meta Classifier	Accuracy drop	Base Classifier	Meta Classifier	Accuracy drop	Base Classifier	Meta Classifier
Banding (3)	[0.00, -0.72]	AN+IC3	<i>KNN</i> , RF, DT	[0.00, -2.00]	AN+IC3+MN N2	<i>KNN</i> , SVM, ABM2	[-0.48, -0.44]	AN+MN2	DT
Blurred (11)				[-8.14, 20.00]	AN+IC3+MN N2	<i>KNN</i> , DT	[-0.72, -11.02]	AN+IC3	<i>KNN</i>
Brightness (8)	[0.00, -6.45]	AN+MN2	<i>KNN</i> , LDA, SVM	[0.00, -50.00]	AN+MN2	RF, <i>KNN</i>	[0.00, -65.15]	AN+IC3	<i>KNN</i>
Broken Lens (15)	[0.00, -14.32]	AN+IC3+MN 2	LDA, <i>KNN</i>	[-4.00, -21.00]	AN+IC3	<i>KNN</i> , ABM2	[-0.84, -49.47]	AN+IC3	<i>KNN</i> , RF
NCAC (1)									
COND (3)				0.00	MN2+IC3	<i>KNN</i> , ABM2	[0.00, -0.36]	AN+IC3	<i>KNN</i>
Dead Pixels (6)				[-1.00, -7.00]	AN+MN2	<i>KNN</i> , LDA, ABM2	[-0.24, -3.72]	AN+MN2	<i>KNN</i>
Dirt (36)				[-1.00, -4.00]	AN+IC3	<i>KNN</i>	[0.00, -1.80]	AN+IC3+MN 2	<i>KNN</i> , SVM
NOBF (1)	-0.48	AN+IC3+MN 2	ABM2	-4.00	MN+IC3	<i>KNN</i>	-33.78	AN+IC3+MN 2	ABM2
Ice (4)	[0.00, -2.15]	AN+IC3	<i>KNN</i> , SVM, DT, RF ABM2	[0.00, -37.00]	AN+MN2	<i>KNN</i>	[-0.12, -40.96]	AN+IC3	<i>KNN</i>
NDEMOS (1)				-2.00	AN+IC3	<i>KNN</i>	-0.72	MN2+IC3	<i>KNN</i>
NNR (9)	[0.00, -27.21]	AN+IC3	<i>KNN</i> , SVM, ABM2, DT	[-3.00, -57.00]	AN+IC3	<i>KNN</i> , SVM	[-0.36, -74.02]	AN+MN2	<i>KNN</i> , DT
Rain (5)	[0.00, -0.72]	AN+IC3+MN 2	<i>KNN</i> , SVM, DT, RF, ABM2	[0.00, -12.00]	AN+MN2	<i>KNN</i> , ABM2	[-0.12, -13.18]	AN+MN2	<i>KNN</i> , DT

5.4 Explanation Of DNNs Robustness

The LIME [92] tool aims to explain the predictions of a DNN. It shows how different models select the image's influential features that contribute the most to the final classification. We explore some images with the LIME toolbox to investigate the reason behind the prediction of the models, especially to understand the difference identified in our experiments between AN, IC3, and MN2.

Figure 19 shows the LIME output for various images that are classified by the three DNNs. Coloring shows which areas of the image contribute the most to the classification. The LIME output shows that AN selects stronger features compared to the other classifiers: this is particularly evident in the second and fourth rows. The LIME output of IC3 shows that its

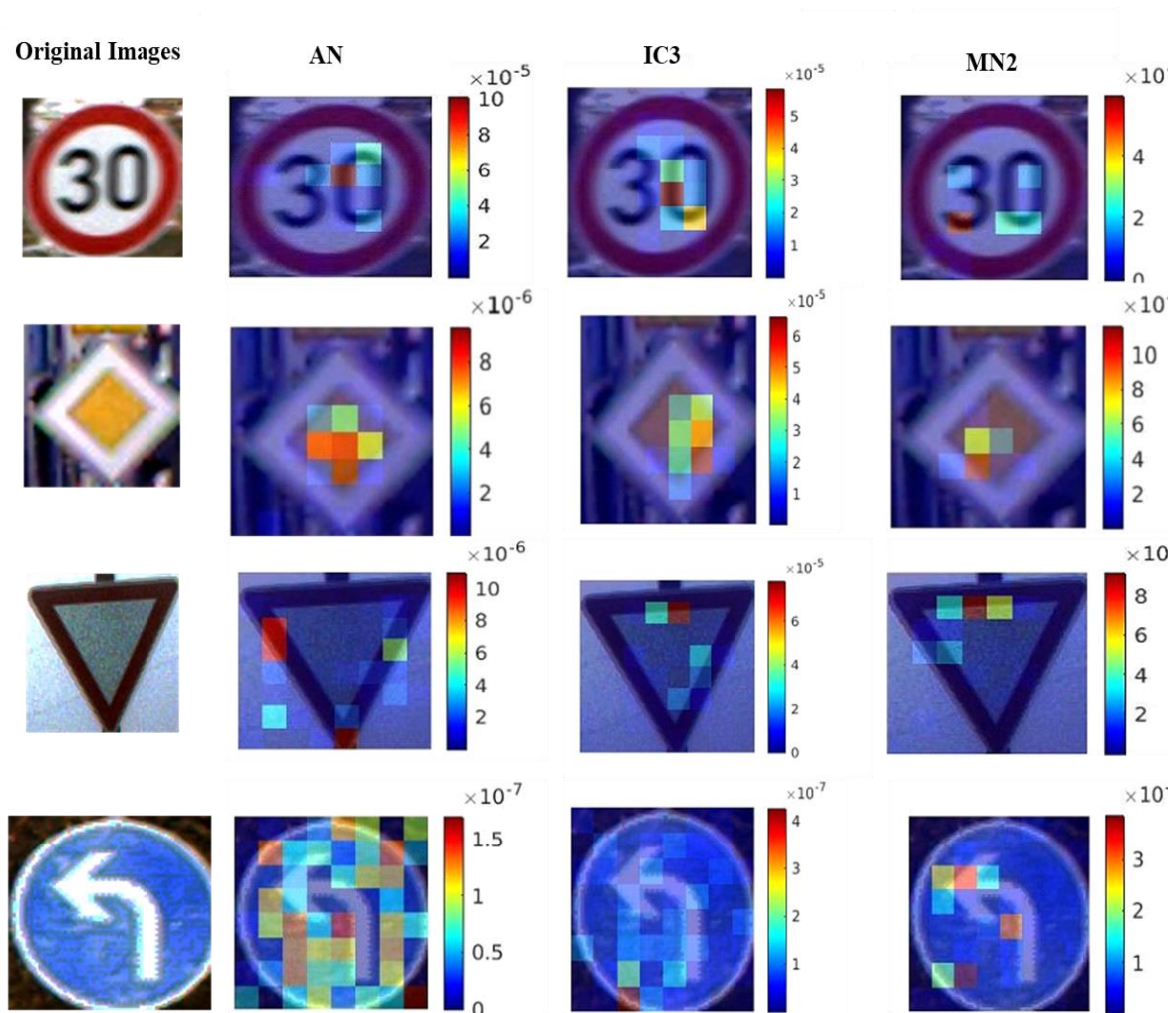


Figure 19: Explanation of AN, IC3, and MN2 models through LIME, highlighting the influential features of selected traffic sign images. (best viewed in color)

strongest features are mostly congested in an area. Instead, AN decides using features that are scattered through the image. This is most likely the reason IC3 is the less robust in our experiments: when a failure is injected in an image, as for example a scratch of the broken lens failure, it may overlap the areas that are relevant for feature classification. As shown in Figure 19, for the blue circular traffic sign, AN highlight many features, and the strongest features are located in various parts of the traffic sign; on the other hand, IC3 and MN2 have very few strong features and mostly in adjacent positions.

Summarizing, the main hurdle for classifiers robustness against different visual camera failures may depend on the DNNs: if the image alteration due to failure is on the image portion which plays the key role in classification, that failure will misguide the TSR to output an incorrect label for the traffic sign.

5.5 Lessons Learned

This chapter considers different strategies for TSR i.e., single-image and sliding window, to understand their robustness against visual camera failures. As reference, we use the datasets GTSRB, DITS, and BelgiumTSC, where no single-image classifier achieves 100% accuracy. Instead, stacking meta-level classifiers that work on sliding windows (W1C, W2C, W3C) significantly improve the accuracy of TSR: they achieve 100% accuracy on all three datasets.

Concerning instead the presence of visual camera failures, we observed that the implemented failures generally lead to a drop in accuracy. For example, Figure 20 highlights the highest accuracy achieved against each visual camera failure on the three datasets separately, i.e., GTSRB, DITS, and BelgiumTSC, considering the failures configurations which has the highest accuracy drop.

However, we also noticed that some TSR strategies are significantly more robust than others and can tolerate the majority of failures. While there is no individual TSR strategy that performs better than the other either against each failure or on all three datasets, the takeaway message of this chapter is that meta-level classifiers in conjunction with sliding windows of subsequent images improve the TSR performance and robustness. More concretely, we report the following considerations on robustness: The 1SFC strategy proved insufficiently robust against visual camera failures, with a severe degradation of accuracy with respect to the clean datasets. Instead, 2SFC and 3SFC strategies are more robust, and present a lower performance degradation; the most critical cases are highlighted in bold in Table 15. One important observation is that IC3 never achieves the highest accuracy on GTSRB in presence of visual

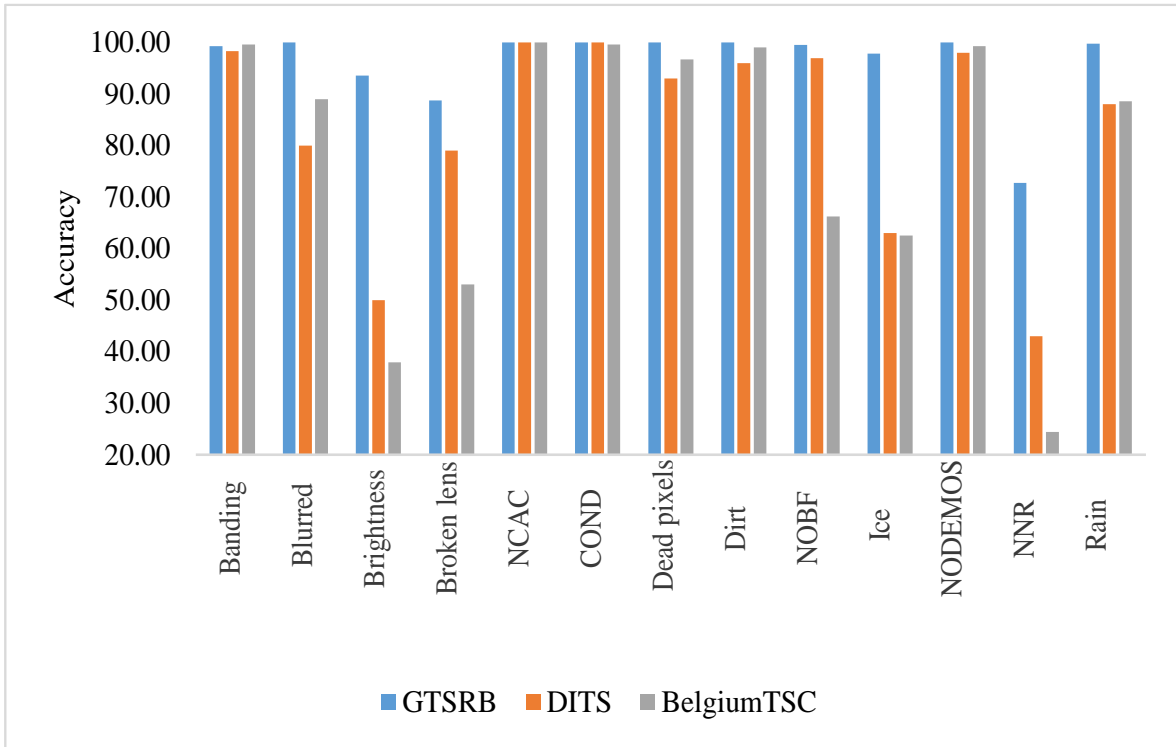


Figure 20: Accuracy achieved for each failure on the three datasets, when we consider the failure configurations which lead to the highest accuracy drop.

camera failures. This is motivated through Explainable AI in Section 5.4. Sliding windows strategies W1C, W2C, and W3C improve the robustness of TSR with respect to 1SFC, 2SFC, and 3SFC as shown in Table 16. Especially, we achieve 100% accuracy against 6 failures on the GTSRB dataset, while for DITS and BelgiumTSC 100% accuracy is achieved for NCAC failure only. DITS achieved 100% accuracy for COND failure using W2C (MN2+IC3).

Further, some failures such as NNR, ice, brightness, broken lens, and NOBF have a bad impact on the robustness of the TSR. Our experimental results indicate that visual camera failures degrade the TSR performance. Despite the different classification strategies, we are unable to minimize the drop in accuracy under all conditions and datasets, exceptions made for NCAC. This brings the necessity of possible strategies to overcome the effect of visual camera failures on the performance of a TSR classifier.

1. We can use data augmentation [97] in which the failed images are added to the training dataset to make the classifier more robust.
2. A detector can analyse each image to understand if the image is of acceptable quality or is degraded. If the image contains defects, an alarm can be raised, and the accuracy of the prediction can be suspected at system level. The detector can be created by training non-deep classifiers on the injected datasets.

The objective of this chapter was to analyse the effect of visual camera failures on the robustness of different TSR strategies. In this chapter we performed extensive experiments on three traffic sign datasets that are perturbed with 13 visual camera failures. Different classifiers are proposed, that are composed either to process a single image or a sliding window of images. We observe that approaches based on sliding windows perform better compared to approaches based on single images, both on the clean datasets, and when failures are considered. Furthermore, some failures impacted the TSR performance severely so that several misclassifications are measured. To wrap up this chapter, overall, there is no TSR strategy that is more robust than the others on all datasets and visual camera failures.

6. IMPROVING ROBUSTNESS OF DEEP IMAGE

CLASSIFIERS AGAINST VISUAL CAMERA FAILURES

The excellent performance of image classifiers based on Deep Neural Networks (DNNs) heavily depends on the training phase, through which the classifier learns how to classify images [36]. DNNs are trained iteratively on a training dataset [100], which contains images and the associated labels describing the class of each image. When the model learned from the DNN is not general, even small perturbations may cause classifiers with high accuracy to produce misclassifications of images [101]. It comes with no surprise that the amount of training data [102] and quantity of noise [103], contained in training data heavily affect the model generalization and consequently the classification task. A small training data set may result in underfitting [105] the model, which relies on poor knowledge and will not be accurate nor general. Overfitting [105], on the other hand, happens when a classifier learns a model that corresponds too closely or exactly to a particular set of data, and may therefore fail to generalize to a different, albeit similar, input set of images. Throughout years, DNNs were made more and more robust to overfitting through techniques such as early stopping [106], batch normalization [107], dropout regularization [108], conjugate gradient [109], and weight decay [110]. In addition, specific DNNs can adequately deal with uncertainty due to noisy labels [111] or in the train images themselves [112]. Altogether, those techniques allow for building DNN models which usually have satisfying generalization capabilities and perform well as image classifiers.

However, when DNNs are deployed into a cyber-physical system or an ICT system in general, the model they learned may not be general enough to face unexpected or anomalous operational conditions. Unexpected operational conditions may be due to a multitude of reasons, to name a few: i) weather may affect the sampling of images that are then fed into the DNN model, ii) physical devices that sense the environment (i.e., visual cameras) may not work properly, iii) adversaries may deliver adversarial images [113], [114] to the DNN through the visual camera, iv) the environment in which the system operates is evolving through time, making the DNN model outdated and thus not general anymore. Noticeably, most of those unexpected operational conditions are due to a problem in the image acquisition process [88], [115], which is usually delegated to one or more visual cameras that sense the environment and produce the image(s). Therefore, this study focuses on how to build image classifiers that are robust against visual camera failures, which generate altered images that are delivered to the image classifier. Examples include, but are not limited to: icing/raindrops on the visual camera lens, broken visual camera lens, sudden change of brightness, and blurring [88], [89].

After a careful review of the literature, we found that there are no studies that show how to build image classifiers that are robust against a wide range of visual camera failures. To this end, in this chapter, we discuss strategies that may help to deal with visual camera failures. Then, we consider a complete set of visual camera failures [88], and the associated library to inject their effects into images. This provides the data baseline upon which we will build robust image classifiers using datasets of traffic signs and renowned DNNs. Then, we show a practical methodology to build image classifiers whose classification performances do not (excessively) degrade when processing altered images. Our results have great applicability as they embrace most of the visual camera failures discovered to date. Research-wise, this chapter describes how to build classifiers that are robust to visual camera failures and the impact they have on image classification. From a practical standpoint, this chapter shows how to build more robust image classifiers to be deployed in autonomous vehicles, surveillance visual cameras, medical imaging, and other domains.

6.1 Robust TSR System Against Visual Camera Failures

This section presents two strategies to build robust DNN image classifiers. We start by defining building blocks that are relevant for all strategies. Then, we present the two strategies (Figure 21d and Figure 21b) and discuss the advantages and disadvantages of adopting each of them.

6.1.1 Main Building Blocks

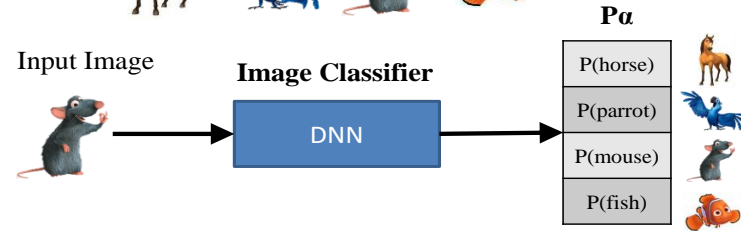
An image classifier, represented in Figure 21a, outputs a set $P\alpha$ of n probabilities, where each $P\alpha_i$ represents the probability of the input image belonging to class i , $0 \leq i < n$. The graphical example in Figure 21a sketches a 4-class classification problem (thus $n = 4$), where the image classifier processes the input, and it outputs an array of 4 probabilities, the highest of which represents the animal predicted by the classifier. Figure 21c depicts the Camera Failure Detector (CFD), which is itself a binary classifier that processes the input image and decides if the image is clean or altered. In the figure, there are 4 different types of visual camera failures, and the CFD solves a binary classification problem (i.e., is the image clean, or is it altered with any of the 4 failures?), which outputs the array $P\phi$ of two probabilities.

6.1.2 Strategies for Building Robust DNNs

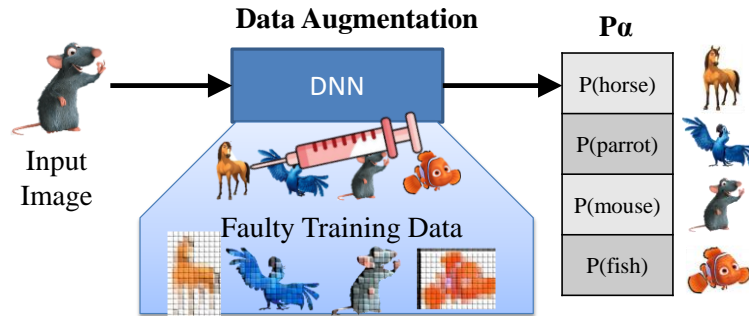
Two strategies can be used for building robust image classifiers as we discuss below.

- **Data Augmentation / Injection** (Figure 21b). Alternatively [117], [118], [119], it is possible to avoid the usage of the CFD and instead put all the efforts into improving the training phase of the DNN, aiming at a more generalized model. In such a scenario, the training set is enriched with perturbations of the clean images of the training set,

Classes for Image Classification: Horse, Parrot, Mouse, Fish

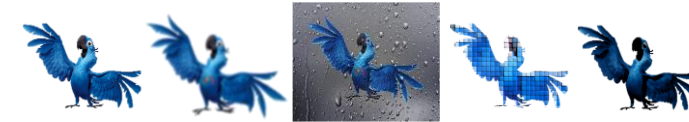


a) General structure of a DNN image classifier.

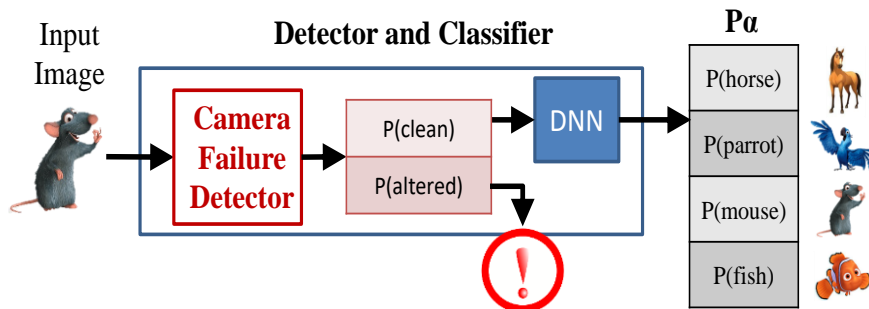


b) Data Augmentation / Injection strategy.

Failures: clean image, blurring, rain drops, banding, darkness



c) Structure of a Camera Failure Detector (CFD)



d) Strategy using a detector CFD and the DNN classifier.

Figure 21: General structure of an image classifier that embeds a DNN, plus two strategies for robustness.

providing a broader set of images to the DNN. For visual camera failures, perturbations are obtained by altering clean images through injection of the effects of visual camera failures.

Table 18: Comparison of different strategies to build robust Image Classifiers.

Strategy	Related Works	Requires CFD	Always Classifies	Impacts Training	Impacts Testing
Master-Slave	[116]	✓		✓	✓
Data Augmentation / Injection	[117], [118], [119]		✓	✓	

- **Master-Slave strategy** (Figure 21d). The CFD and the DNN operate in sequence [116]. Should the CFD predict that the input image is not altered (and thus $p(\text{clean})$ in the figure is the highest out of the two probabilities in $P\phi$), it will propagate it to the DNN that performs classification. Otherwise, the CFD will alert the user or trigger routines that handle altered images. With this strategy, the DNN executes only if the CFD classifies the input image as clean.

6.1.3 Comparison of Strategies to Improve Robustness

Each strategy holds its strengths and weaknesses and may be considered suitable for a given domain or system, and unfeasible to implement in a different system. To identify the pros and cons, in Table 18 we report different characteristics of the Master-Slave and Data Augmentation/Injection strategies. For each strategy, we analyze, from left to right in Table 18, if: i) a CFD is required, and it is an additional component that needs to be trained and exercised alongside the DNN, ii) the DNN always provides a classification result, or may stop the process should it detect altered images as it happens with Master-Slave strategy, iii) make the training phase more arduous or iv) make the testing phase of the image classifier more complicated.

Ideally, the process of building a robust classifier should have a minimal impact with respect to the training and testing phases of a regular image classifier. The strategy that requires a CFD has a noticeable impact on training due to the necessity of training the CFD alongside the DNN. Also, it adds overheads in testing as – again – it has to exercise both the CFD and the DNN. The Data Augmentation strategy has a huge impact on training but adds no overhead to the testing phase. Additionally, it does not open the problem of aborting the classification when altered images are detected. This happens with the Master-Slave approach, which will require setting up or triggering specific reaction strategies at the system-level.

But still, in the following we will build and experimentally evaluate both strategies such as image classifiers that are robust to visual camera failures following the approach of data augmentation (injection of failures into clean images) and Master-Slave strategy.

6.2 Experimental Campaign and Methodology

We implement image classifiers with data augmentation through experiments that require the following items: In Section 2.2.7, we gather our data baseline for classification: those are

images of Traffic Signs. Then, we describe how we altered the clean images contained in the datasets by injecting 13 different visual camera failures under different configurations. Section 6.2.1 describes how we altered each image with a total of 103 different failure types and their configurations. Once data preparation is complete, Section 6.2.3 lists three DNNs that we will be using to perform image classification (in our case TSR).

6.2.1 Injection of Visual Camera Failures

Building on the papers [85], [88], we injected 13 different visual camera failures of Section 5.2.4 on each image contained in each of the three datasets. We set up a Python script to inject each failure under different configurations.

6.2.2 Building Train and Test Sets for Classifiers

We collect three public dataset of Section 2.2.7 i.e., GTSRB, DITS, and BelgiumTSC datasets and then we process each image to inject the 103 failure configurations from Section 6.2.1. This creates, for each GTSRB, DITS, and BelgiumTSC, 103 variants in addition to the original dataset, for a total of 104 datasets. Each of these 104 datasets is split into a train and test set; the split is identical for all datasets.

We call *train_clean/test_clean* the sets containing the train/test split of the original dataset; both *train_clean* and *test_clean* contain only clean images. We call *train_altered/test_altered* the train/test set which contain *train_clean/test_clean* and the train/test splits of all the 103 variants of the original dataset. For the sake of our analysis, we will also need to understand how each of the six classifiers deals with images corrupted with different configurations of the same visual camera failure. As such, we group the 103 variants depending on the 13 failures we used to generate them. This will allow creating 13 groups of test splits of variants as follows:

- *Failures as NOBF, NCAC, NDEMOS were used to create a single variant as they have only a single configuration of failure;*
- *Failures as Banding and COND were used to create three variants;*
- *Ice, Rain, Black Pixels, Brightness, NNR, Blurring, Broken Lens, Dirt failures are responsible for generating 4, 5, 6, 8, 9, 11, 15, 36 variants each.*

Each of the 13 groups above contain all the variants generated using each of the 13 visual camera failures; different groups differ in the number of variants – and images - they contain, which depends on the number of failure configurations. Noticeably, testing a classifier on each group will provide a set of classification metric scores, one for each test split of a variant in the group. For example, testing a classifier with variants in the *Ice* group will output 4 metric scores, whereas testing a classifier with variants in the *Banding* group will generate only 3 metric scores, since our study considers 3 configurations of the Banding failure.

6.2.3 DNNs for Traffic Sign Recognition

We adapted three pre-trained image classifiers described in Section 2.2.4 such as AlexNet (AN, [34]), InceptionV3 (IC3, [40]), and MobileNet-V2 (MN2, [41]) to our TSR case study using transfer learning with a batch size of 32 and 10 epochs, utilizing the stochastic gradient descent with momentum (sgdm, [134]) optimizer and cross-entropy loss at the Softmax layer on each of the GTSRB, DITS, and BelgiumTSC datasets. We re-trained each DNN (AN, IC3, MN2) twice on each of the three datasets as follows:

- *regular classifier: we exercised transfer learning using the train_clean set of each of the three datasets; we employed classical techniques as image scaling and translation to limit overfitting.*
- *augmented classifier: created using the 50% of train_altered set as data baseline for transfer learning, without image scaling nor translation.*

This process creates a total of 18 classifiers: for each of the 3 datasets, we create a regular and augmented classifier for AN, MN2, and IC3 DNNs. We will then use the test_clean and test_altered sets for testing each classifier and quantifying their classification performance. Single augmented classifier (e.g., AN, MN2, and IC3) are trained for each of three datasets independently and that are dealing with the entire failure set.

6.2.4 Building Train and Test Sets for Binary Failure Detector

To train and test Camera Failure Detector (CFD) of Section 6.2.6, we utilized the test_clean and test_altered datasets of Section 6.2.2. First all test_clean and test_altered images are provided as input to the augmented classifier that may result in either correct classification or misclassification and based on the output of augmented classifiers, we assigned binary labels to each sample of test_clean and test_altered images as shown in Figure 22. As shown in Figure

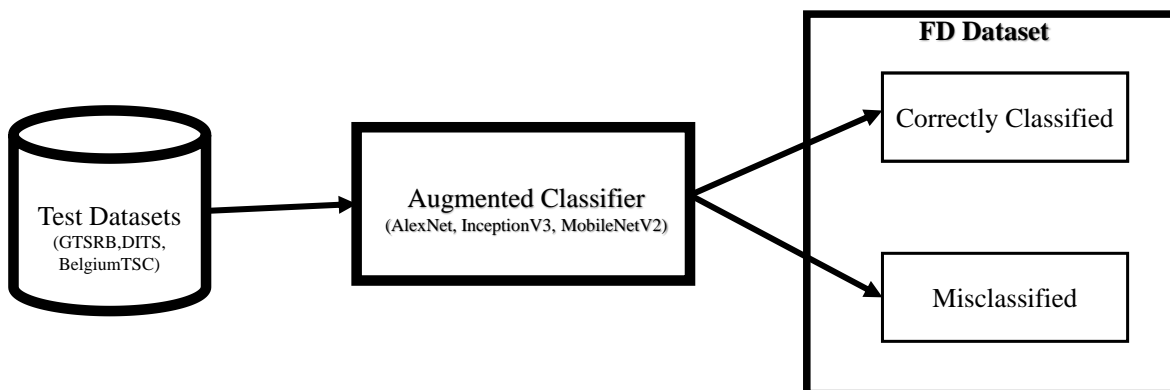


Figure 22: Process to assign binary labels to each test_clean and test_altered image of three datasets classified independently by each augmented classifier.

22, we have three different datasets (GTSRB, DITS and BelgiumTSC), and three independent augmented classifiers (AlexNet, InceptionV3 and MobileNet-V2) that result in 9 different datasets with binary labels i.e., correctly classified or misclassified and we call it Failure Detector (FD) dataset. Each FD dataset is divided into two parts i.e., 60% for training and 40% for testing of CFD.

6.2.5 Input Features for Binary Failure Detectors Training and Testing

We extract features from images by means of deep convolutional neural networks i.e., AlexNet and ResNet feature descriptors, as described in Section 2.2.2. We provide each feature vector independently for the training and testing of CFD. In addition, we also provide the combination of feature descriptors i.e., {AlexNet U ResNet}.

6.2.6 Classifiers Parameters for Binary Failure Detectors

Each non-deep classifier of Section 2.2.3 is trained as CFD with each input feature vector of Section 6.2.5. Each non-deep classifier has its own set of hyper-parameters. To such an extent, we identified the following parameter values to exercise grid searches.

- *KNN with different values of k , i.e., different odd values of k from 1 to 25.*
- *SVM: we used Linear SVM, and SVM with RBF kernel leaving other parameters (e.g., ν) as default.*
- *Decision Tree: we used the default configuration of MATLAB which assigns $MaxNumSplits = training\ sample\ size - 1$, with no depth limits on decision trees.*
- *Boosting: we created boosting ensembles with AdaBoostM1, GentleBoost, and LogitBoost by building 25, 50, 75, and 100 trees (decision stumps) independently.*
- *Random Forest: we build forests of 25, 50, 75, or 100 decision trees. Furthermore, we also exercised TreeBagger with the same number of decision trees.*
- *LDA: we Trained LDA using different discriminants, namely: linear, quadratic, pseudo-linear, diag-linear, diag-quadratic, and pseudo-quadratic.*

6.2.7 Performance Metrics

To evaluate the performance of regular and augmented classifiers, we use accuracy as performance metric, while for CFD performance is evaluated by accuracy, precision, and recall. Finally, to evaluate the performance of the TSR system with CFD, we use F-AA that is inspired by the formulation of F1-measure. All performance metrics are defined in Section 2.2.8.

6.3 Experimental Results of Augmented Classifiers

This section presents the results of our experimental campaign. Section 6.3.1 compares accuracy scores that regular and augmented classifiers achieve on the 14 test sets. Section 6.3.2 expands on brightness and NNR failures, which have a major impact on augmented classifiers. Section 6.3.3 investigates the contribution of visual camera failures in building robust augmented classifiers. Section 6.3.4 shows how augmented classifiers can correctly classify even clean images that are being misclassified by regular classifiers and have better overall classification performance.

6.3.1 Are we building Robust Image Classifiers?

First, we evaluate if augmented classifiers are robust to visual camera failures. We compare the classification performance of regular classifiers against augmented classifiers in Figure 23. The figure is composed of two series of box plots: on the left (Figure 23a) we have accuracy of regular classifiers. On the right side (Figure 23b) we have accuracy of augmented classifiers. Both figures are built using accuracy scores achieved by the three DNNs AN, MN2, and IC3 on the three datasets GTSRB, BelgiumTSC, and DITS when testing on clean images and on images altered using all the 103 possible configurations of visual camera failures. Each figure contains 14 box plots, one for each test set from Section 6.2.1: from left to right, we see results of the test_banding, test_blurring, test_brightness, test_brokenlens, test_NCAC, test_COND, test deadpixels, test_dirt, test_grayimages, test_ice, test_NDEMOS, test_NNR, test_rain, and test_clean. On the vertical axis we have the accuracy: the higher the accuracy, the better the classification performance.

The two plots in Figure 23 look very different from each other: boxes in Figure 23a clearly span across a wider range of accuracy scores with respect to Figure 23b. This means that regular

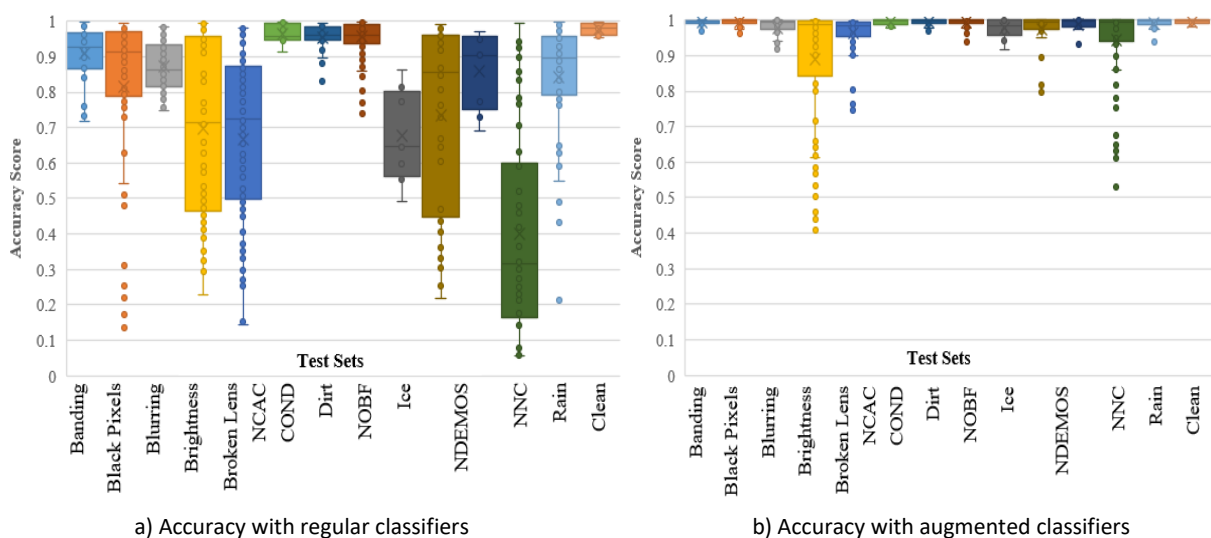


Figure 23: Box Plots showing accuracy of the three regular classifiers (Figure 23a) and the three augmented classifiers (Figure 23b) on the three datasets, for the test_clean and the 13 test_<failure>.

classifiers have more variability in their classification performance than augmented classifiers. Most importantly, augmented classifiers have almost maximum accuracy, which is desirable for any classification task.

There are two additional remarks we can extract from Figure 23. First, the accuracy improvement of the augmented against the regular classifiers is not constant across the different test sets (and visual camera failures). The accuracy is almost perfect (~ 1) when using augmented classifiers for the *test_banding*, *test_blackpixels*, *test_NDEMOS*, and *test_rain*, where instead regular classifiers struggle. Overall, boxes of Figure 23b are narrow and close to the top of the plot, which is clearly valuable. For most of the other test sets, there is still a clear improvement when using the augmented classifiers, whose accuracy gets close to 1.0. Second, the accuracy changes also when classifying the *test_clean* set, with no visual camera failures. The last box of Figure 23a (*test_clean*) is wider than the corresponding one in Figure 23b. All augmented classifiers are very good on clean images; instead, the accuracy scores of regular classifiers have more variance than augmented classifiers. We will detail this behavior in Section 6.3.4, which takes advantage of a tool for explainable AI.

We further explore the accuracy scores in Table 19 and Table 20. The table reports a row for each test set, and three groups of columns, one for each of the datasets. For each dataset, we report the highest accuracy achieved by DNNs. Accuracy scores in the table often report ranges: those happen when a test set embeds multiple configurations of visual camera failures, which may impact accuracy differently. Instead, *test_clean* contains no failures, and

Table 19: Accuracy achieved using regular classifiers on the three datasets.

Test Set or Group of Variants (# Config.)	GTSRB		DITS		BelgiumTSC	
	Accuracy	Best DNN	Accuracy	Best DNN	Accuracy	Best DNN
Clean	0.994	MN2	0.96	IC3	0.997	AN
Banding (3)	[0.947, 0.963]	AN	[0.902, 0.952]	IC3	[0.991, 0.996]	AN
Black Pixels (6)	[0.875, 0.993]	MN2	[0.849, 0.956]	MN2	[0.940, 0.995]	AN
Blurring (11)	[0.946, 0.983]	MN2	[0.780, 0.837]	IC3	[0.862, 0.970]	AN
Brightness (8)	[0.592, 0.959]	AN	[0.359, 0.941]	AN	[0.324, 0.996]	AN
Broken Lens (15)	[0.469, 0.836]	AN	[0.607, 0.852]	AN	[0.465, 0.981]	AN
COND (3)	[0.974, 0.993]	MN2	[0.932, 0.955]	IC3	[0.988, 0.995]	AN
Dirt (36)	[0.970, 0.994]	MN2	[0.845, 0.959]	IC3	[0.978, 0.997]	IC3
Ice (4)	[0.729, 0.960]	AN	[0.362, 0.938]	AN	[0.603, 0.988]	AN
NOBF (1)	0.861	MN2	0.812	IC3	0.598	MN2
NCAC (1)	0.993	MN2	0.949	IC3	0.996	IC3
NDEMOS (1)	0.963	AN	0.774	IC3	0.97	IC3
NNR (9)	[0.419, 0.959]	AN	[0.317, 0.903]	AN	[0.228, 0.995]	AN
Rain (5)	[0.864, 0.958]	AN	[0.790, 0.939]	AN	[0.864, 0.992]	AN

Table 20: Accuracy achieved using augmented classifiers on the three datasets.

Test Set or Group of Variants (# Config.)	GTSRB		DITS		BelgiumTSC	
	Accuracy	Best DNN	Accuracy	Best DNN	Accuracy	Best DNN
Clean	1.000	MN2	0.992	MN2	0.999	AN
Banding (3)	[0.999, 0.999]	IC3	[0.991, 0.993]	MN2	[0.998, 0.999]	IC3
Black Pixels (6)	1.000	MN2	[0.987, 0.994]	MN2	[0.997, 0.999]	IC3
Blurring (11)	[0.996, 0.999]	IC3	[0.975, 0.983]	MN2	[0.997, 0.999]	IC3
Brightness (8)	[0.990, 1.000]	MN2	[0.458, 0.992]	MN2	[0.668, 0.999]	IC3
Broken Lens (15)	[0.902, 0.997]	IC3	[0.948, 0.991]	MN2	[0.996, 0.998]	IC3
COND (3)	1.000	MN2	[0.988, 0.994]	MN2	[0.998, 0.999]	IC3
Dirt (36)	1.000	MN2	[0.972, 0.995]	MN2	[0.997, 0.999]	IC3
Ice (4)	1.000	MN2	[0.894, 0.993]	MN2	[0.976, 0.999]	IC3
NOBF (1)	1.000	MN2	0.982	MN2	0.986	MN2
NCAC (1)	1.000	MN2	0.992	MN2	0.998	IC3
NDEMOS (1)	1.000	MN2	0.983	MN2	0.998	AN
NNR (9)	[0.999, 1.000]	MN2	[0.649, 0.993]	MN2	[0.859, 0.998]	AN
Rain (5)	1.000	MN2	[0.988, 0.993]	MN2	[0.997, 0.998]	IC3

test_NCAC, *test_NOBF*, and *test_NDEMOS* contain a single configuration for a failure: they have a single accuracy value in Table 19 and Table 20.

The table confirms that regular classifiers always have inferior accuracy than augmented classifiers. Looking at the GTSRB columns, we see that in many cases, even in presence of visual camera failures, the classification accuracy is perfect (1.0). However, this trend does not repeat over the other two datasets, which are harder to classify: neither regular nor augmented classifiers reach 100% accuracy on *test_clean* (first row of Table 19 and Table 20). As such, we cannot expect perfect accuracy on test sets with altered images.

Also, there are test sets that result in low accuracy regardless of the training dataset and the DNN. This is the case of *test_brightness*, *test_NNR*, *test_ice*, and, to a lesser extent, *test_brokenlens*. The range of accuracy of *test_ice* on DITS and BelgiumTSC datasets is quite wide, even when adopting augmented classifiers. For some configurations of the ice visual camera failure, accuracy in DITS drops as low as 0.894 even when using the augmented classifiers. This is due to the way the ice image overlays with the image of the traffic sign image: in some cases, it covers significant parts of the image, whereas sometimes it does not. On BelgiumTSC accuracy goes down to 0.976, which is decent accuracy in general terms, but still lower than the one achieved in the other test sets, exception made for *test_brightness* and *test_NNR*.

In addition, Table 19 and Table 20 shows which DNN produces the highest accuracy value on each test set. There is no clear winner for regular classifiers: sometimes AlexNet (AN) performs better than MobileNet-V2 (MN2) and InceptionV3 (IC3), while in other cases it is the opposite. Considering the different DNNs, the most performing is typically MN2: MN2 is the preferred DNN for all test sets in the DITS dataset and 11 out of 14 test sets of GTSRB. For BelgiumTSC, instead, it looks more beneficial to adopt IC3. It is worth mentioning that AN gets better classification performance than MN2 and IC3 regular classifiers: for this DNN, the training with *train_altered* was not helpful as with MN2 and IC3 in raising their accuracy scores even in presence of failures.

Overall, we conclude that the augmented classifiers have significantly better classification performance than regular classifiers both when processing clean and altered images. Particularly, their accuracy remains high when dealing with images containing visual camera failures, making them robust to those altered images.

6.3.2 Effect of Brightness and Noise on Image Classifiers

We complement the general analysis in the previous section by exploring the impact of brightness and noise visual camera failures. Those are interesting cases for discussion because they have a major effect on the classification performance of both regular and augmented classifiers.

6.3.2.1 Brightness analysis

Figure 24 plots accuracy achieved by regular (Figure 24a) and augmented (Figure 24b) classifiers when varying brightness level in the range [0.1; 15], where 0.1 is an image almost entirely black, 1 is the clean image, and 15 is an image that is almost entirely white. Figure 24a shows how brightness levels in the range [0.3; 1.5] do not have a major impact on the accuracy

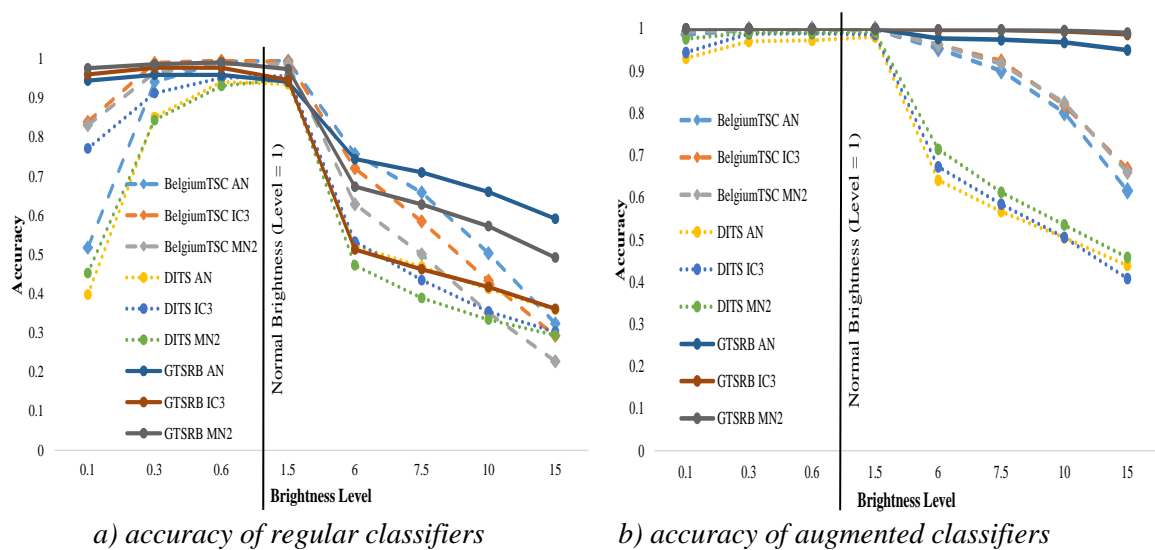


Figure 24: Accuracy achieved on test_brightness by regular classifiers (left) and augmented classifiers (right) using the 3 DNNs AN, MN2, IC3 on the 3 datasets DITS, BelgiumTSC, GTSRB and brightness levels in the range [0.1 to 15].

of regular classifiers, whereas brightness levels lower than 0.3 and greater than 1.5 dramatically reduce accuracy. In Figure 24b we can observe that augmented classifiers are more robust to brightness failures: their accuracy does not drop much with low brightness, and suffers serious degradation only with brightness level greater than 1.5 and only in DITS (dotted lines in the plot) and BelgiumTSC (dashed lines in the plot). Even in those cases, the degradation of accuracy is graceful whereas regular classifiers show a sudden drop of accuracy.

6.3.2.2 Noise analysis

Figure 25 depicts a plot similar to Figure 24 but considering the *test_NNR* test set. The clean image has noise level 0; the higher the noise level, the more the image is degraded. Figure 25a shows that regular classifiers struggle to classify noisy images. Specifically, with InceptionV3 (IC3) on GTSRB, accuracy drops below random guessing (accuracy of 12.5% for an 8-class classification problem) when the noise level reaches or exceeds 0.6. This is visible in the figure by looking at the brown solid line, which immediately falls at the bottom of the plot. On the opposite, Figure 25b shows that augmented classifiers are to some extent robust to NNR: when noise level is below 1, the accuracy still exceeds 90%. Then, it drops when noise is further incremented. Similarly, to the brightness failure, with the GTSRB dataset we observe only a very slight performance degradation: we can hypothesize that this is due to a high number of images in the training data and the high resolution of images.

In Figure 26, we show a traffic sign to which we apply brightness and noise values, for a visual reference of the different level of noise and brightness.

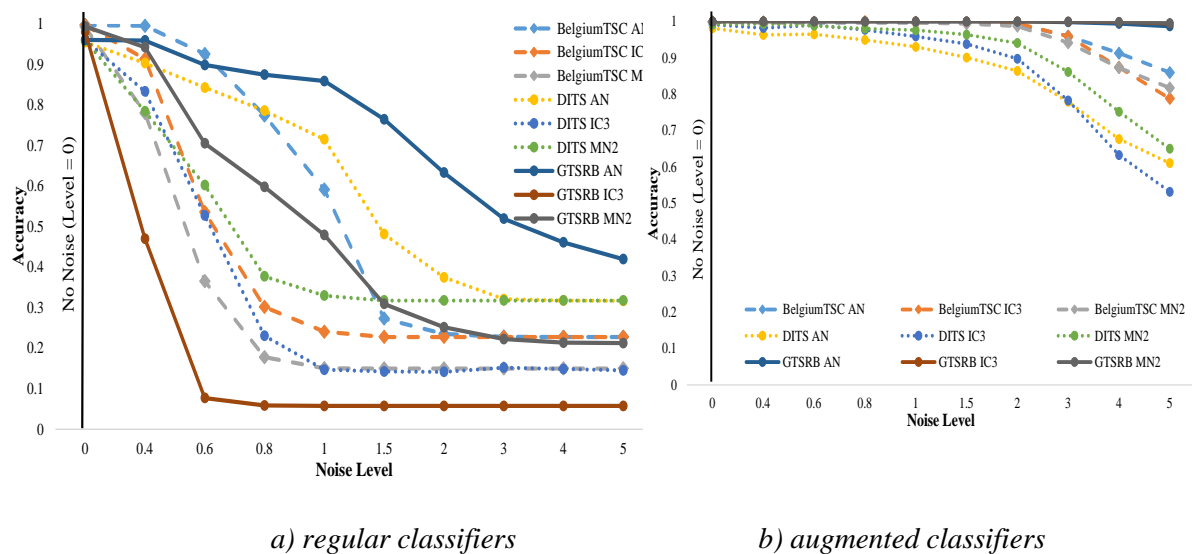


Figure 25: Accuracy achieved on test_NNR by regular classifiers (Figure 25a, on the left) and augmented classifiers (Figure 25b, on the right) using different DNNs on different datasets with different levels of noise (0 to 5).



a) Brightness levels 0.3, 1, 1.5, 6.

b) Noise levels 0, 0.6, 1.5, 3

Figure 26: Different brightness and noise levels applied to the same traffic sign.

6.3.3 On the Impact of Individual Failures on Training

We provided evidence that data augmentation with altered images improves classification accuracy under all the considered scenarios. We now investigate which visual camera failure contributes the most to this improvement. For this purpose, we train a DNN on 13 train variants, each composed of the *train_clean* set plus images altered with a single visual camera failure. This creates 13 ‘intermediate’ classifiers in between the regular and the augmented. For brevity, we explore the behavior of MN2 on DITS only. We deem it more interesting to explore the DITS dataset instead of GTSRB and BelgiumTSC as classification on DITS has the lowest accuracy scores: thus, we believe it is the most challenging. Also, we choose MN2 because (from Table 20) it is the DNN that always achieves the highest accuracy on DITS.

Table 21 contains a row for each test set. On the columns, we have 15 classifiers: the regular, the 13 obtained using the train variants above, and the augmented. The second column of the table reports the minimum accuracy achieved by the regular MN2 classifier on any

Table 21: Difference between the minimum Accuracy of a regular classifier and the minimum accuracy obtained by MN2 on 14 test sets of DITS. MN2 is trained using *train_clean* plus data of a single visual camera failure.

Test Set	Regular Classifier	Clean + Banding	Clean + Black Pixel	Clean + Blurring	Clean + Brightness	Clean + Broken Lens	Clean + COND	Clean + Dirt	Clean + Ice	Clean + NOBF	Clean + NCAC	Clean + NDEMOS	Clean + NNR	Clean + Rain	Augmented Classifier
	Accuracy	Difference from accuracy of regular classifier													
Clean	0.96	0.019	0.02	0.027	0.024	<u>0.032</u>	0.022	0.03	0.016	0.018	0.019	0.024	0.002	0.023	0.033
Banding	0.719	<u>0.254</u>	0.145		0.064	0.13			0.018			0.018	0.222	0.139	0.273
Black Pixels	0.849		<u>0.118</u>			0.067								0.009	0.135
Blurring	0.747			<u>0.211</u>		0.014									0.228
Brightness	0.294		0.046	0.046	<u>0.16</u>	0.049	0.067		0.04	0.06	0.037	0.01	0.027	0.055	0.164
Broken Lens	0.569					<u>0.356</u>			0.019						0.38
COND	0.883		0.047			<u>0.074</u>	0.072	0.039	0.066				0.038	0.06	0.105
Dirt	0.74	0.009	0.055	0.104	0.07	0.121	0.037	<u>0.22</u>	0.021			0.05		0.06	0.232
Ice	0.304		0.028	0.027	0.082			0.027	<u>0.541</u>				0.107	0.124	0.59
NOBF	0.79	0.091	0.091	0.017	0.047	0.079	0.089	0.063		<u>0.181</u>	0.093	0.072	0.086		0.192
NCAC	0.947		0.015	0.009	0.022	0.016	0.018	0.027	0.009	0.011	<u>0.028</u>	0.02	0.009	0.016	0.046
NDEMOS	0.728	0.062	0.148	0.111	0.065	0.059	0.1	0.167	0.053	0.126	0.084	<u>0.229</u>	0.12	0.106	0.255
NNR	0.318												<u>0.355</u>		0.332
Rain	0.763		0.108			0.078							0.076	<u>0.187</u>	0.225

configuration of a specific failure to be used as a reference. The quantities shown in the rest of the table are the difference between the minimum accuracy achieved with the 13 new classifiers and the one achieved by the regular classifier. Only positive differences are reported; these are the cases in which training with a specific train variant creates a better classifier than the regular one. The biggest accuracy gains for each test set (each row in the table) are underlined: this shows the train variant which improves classification the most, with respect to training on *train_clean*. Finally, the last column of the table reports the accuracy difference between the *augmented* MN2 and the *regular* MN2: as expected, all differences are positive and always outperform any accuracy improvement we obtained with any train variant except for *test_NNR*, where we achieve highest accuracy by *train_clean* and *train_NNR* dataset.

For all test sets but *test_clean* and *test_COND* the biggest improvement in accuracy is obtained adding images altered with the corresponding failure in the train set. For example, training with clean images and those altered with the dirt failure increases accuracy for *test_dirt*. This provides the DNN with information about the effects of the specific visual camera failure, letting MN2 learn how to classify even in the presence of that specific failure. Interestingly, the biggest improvement in accuracy for *test_clean* and *test_COND* is when adding images altered with the Broken Lens failure to the train set.

Another interesting observation regards the accuracy on clean images (first row of Table 21). The augmented classifier scores 0.993 of accuracy with an increase of 0.033 with respect to the regular classifier, which scores 0.960. Further, training with broken lens failure allows a gain in accuracy of 0.032, and training with dirt failure provides an accuracy improvement of 0.030: almost as good as those of the augmented classifier. This may suggest that adding scratches and dirt effects on train images could improve image classification in general, regardless of the purpose of protecting against visual camera failures.

6.3.4 Explaining Performance of Regular and Augmented Classifiers on Clean Images

While it is intuitive that the augmented classifiers have better classification accuracy than the regular ones on altered images, it is worth understanding why classification accuracy on clean images is also improved.

Figure 27 shows several images from DITS that are misclassified by the regular MN2 classifier (on the left of the figure) but are correctly classified by the augmented MN2 classifier (on the right of the figure). Some of the images in the figure, especially those in the third and fourth row, already have blurred effect. This is not due to failure injection, but instead is a characteristic of the DITS dataset, which also contains images taken in non-ideal sampling conditions. Anyways, for the purpose of the study, those are considered clean images as they are contained in the input dataset.

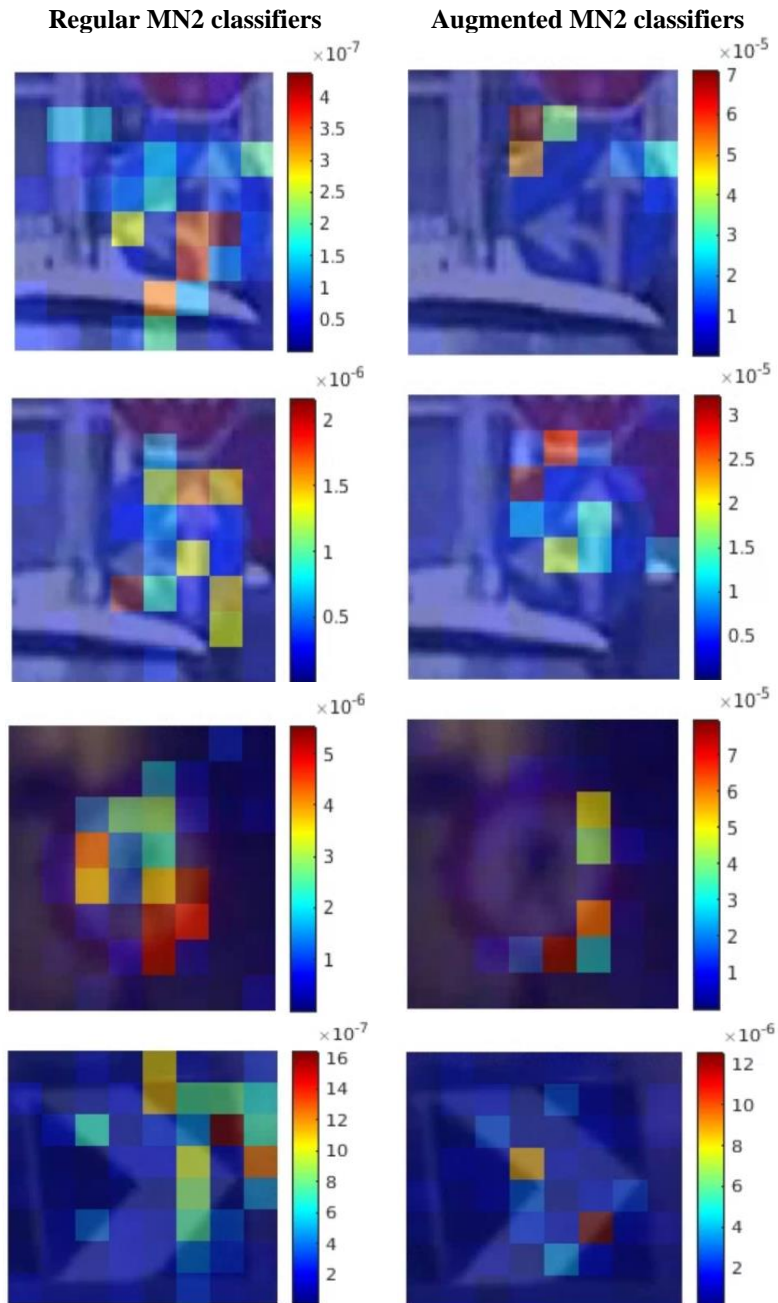


Figure 27: Explanation of predictions of regular (on the left) and augmented (on the right) classifiers using LIME. Figure reads better when viewed in colors.

Frameworks for eXplainable AI (XAI) such as LIME [92] allow for examining how a classifier builds its prediction and ultimately explains the process behind the outputs of image classifiers. Particularly, LIME provides a graphical interface that shows which areas of the input image have the most relevance when calculating the output prediction. Each image in Figure 27 was processed through LIME, which visualize a heatmap of the most relevant features used for the prediction. Red areas correspond to features that contribute the most to classification, while blue ones have negligible to no impact.

The regular MN2 classifiers on the left of Figure 27 select relevant features from many areas of the image; whereas the augmented MN2 classifiers select only a few. Moreover, the color scales on the right of each image show a contribution of each feature in the order of 10^{-6} or even 10^{-7} for regular classifiers, whereas features for the augmented classifier have bigger absolute weight, in the magnitude of 10^{-5} . Those two observations pair well together: the augmented classifier selects a few strong features, whereas the regular classifier selects many weak features from different parts of the image. This difference in the models learned from regular and augmented classifiers turns out to improve image classification accuracy: relying on less features provides a clear advantage in our experiments.

6.4 Experimental Results of Camera Failure Detector

This section compares different binary CFDs trained on the dataset generated in Section 6.2.4. We did not consider the GTSRB dataset for CFD classifier analysis as augmented classifier achieves better accuracy against failure datasets compare to other two datasets. Results of binary CFD classifiers is shown in Figure 28, where blue, orange and gray solid bars represent accuracy, precision and recall respectively. We applied different configurations for each CFD classifier that are independently trained using different feature sets, whereas Figure 28 shows the records against each CFD classifier based on highest F1-measure. On a first

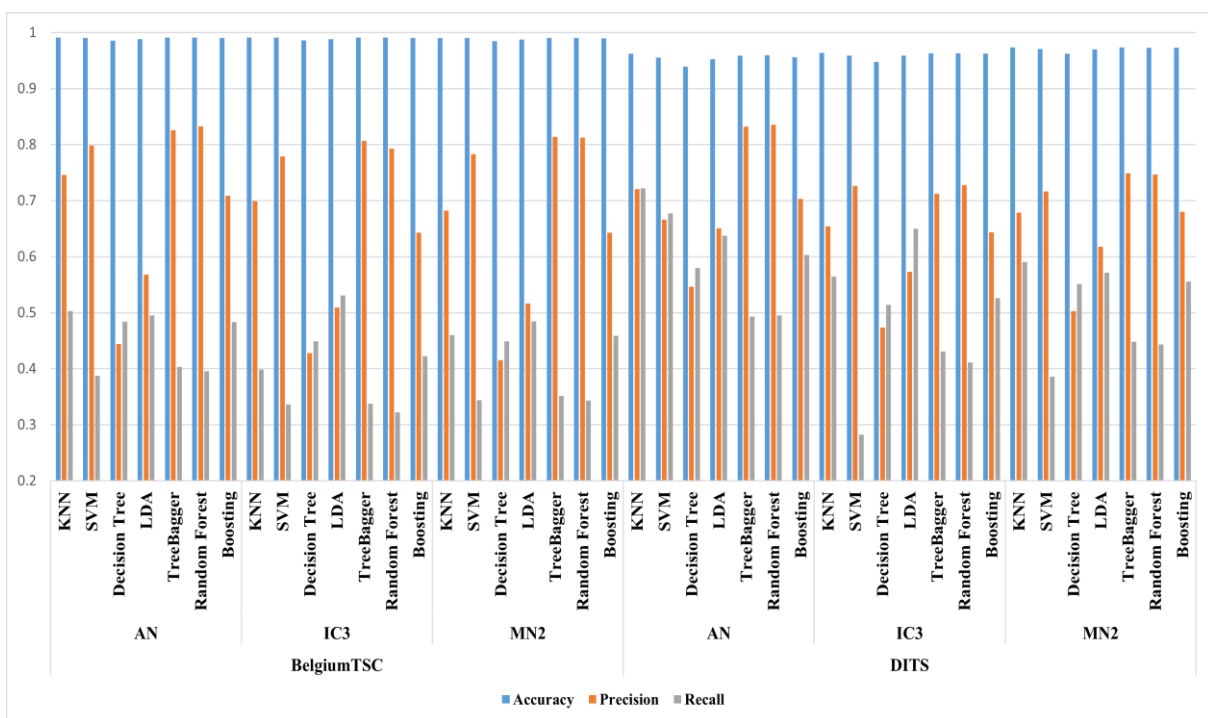


Figure 28: Performance of different binary CFD classifiers on two datasets i.e., BelgiumTSC and DITS by independently considering each Augmented Classifier

glance, we can observe that CFD classifiers achieves better accuracy on BelgiumTSC dataset compared to DITS, the main reason behind better performance is the quality and number of the images in the training dataset. Augmented classifier AN with CFD classifier KNN achieves the highest accuracy 0.9913, which is better compared to all other combinations of Augmented and CFD classifiers on BelgiumTSC dataset. Similarly, the highest accuracy (0.9738) on DITS dataset is achieved by MN2 with KNN classifier. We observed that, CFD classifier KNN perform better on DITS dataset, while on BelgiumTSC dataset Tree Bagger performance is better with all augmented classifiers, despite in one case AN and KNN achieves slightly better performance. Figure 28 shows that on BelgiumTSC dataset IC3 with CFD classifier LDA achieves higher recall, which means that the combination of IC3 and LDA produces lower number of false negatives. On the other hand, higher specificity score is achieved by AN and random forest that shows that this combination results in lower number of false positives.

Similarly, on DITS dataset AN and random forest achieves highest specificity score 0.8359. The highest recall score (0.7223) is achieved by the combination of augmented classifier AN and CFD classifier KNN, while all other combinations of augmented and CFD classifiers achieves recall score less than 0.677. Results suggest that CFD classifier can recognize the samples that will mislead the classifier component of the TSR system. To such extent, Section 6.5 discuss the impact of embedding CFD component in TSR system.

6.5 Experimental Results of TSR System using CFD

This section builds TSR systems by considering two different strategies: (i) augmented classifiers and (ii) augmented classifiers in conjunction with CFD component. Both TSR systems are tested independently on test data generated in Section 6.2.4.

Table 22 shows the accuracy achieved by augmented classifiers on 40% test samples of Section 6.2.4 that contain the failure injected samples from all 103 different visual camera failure configurations. We observed that AN and MN2 achieves the same accuracy i.e., 0.9869

Table 22: Accuracy achieved by augmented classifiers on 40% test set containing all failure samples with same proportion

Classifiers/Dataset	BelgiumTSC	DITS
AN	0.9869	0.9324
IC3	0.9881	0.9504
MN2	0.9869	0.9619

on BelgiumTSC dataset, while IC3 achieves the highest accuracy 0.9881 that is still far from perfect. While on the DITS dataset MN2 achieves the highest accuracy 0.9619 on DITS dataset.

Experimental results in Table 23 and Table 24 shows the highest performance of the TSR system that combines the augmented classifier and the CFD on BelgiumTSC and DITS dataset, where the records are selected based on highest F-AA that is defined in Section 2.2.8. F-AA is giving equal importance to the accuracy and availability of TSR system. We considered only BelgiumTSC and DITS dataset for the experiments of CFD as we already achieved perfect classification against majority of camera failures by exercising only augmented classifiers. datasetsTo ensure the safety, there is always a trade-off between the performance and availability of the system, and they are dependent on each other as in Table 23 and Table 24, where the highest values of availability and accuracies are highlighted with bold. Table 23 and Table 24 shows the performance of TSR system selected by highest F-AA values shown in 7th column, where 5th and 6th columns shows the accuracy and availability of the TSR system

Table 23: Overall TSR system performance on BelgiumTSC dataset, for each augmented and CFD classifiers records are selected by highest F-AA calculated between accuracy and availability.

Augmented Classifier	CFD Classifiers	CFD Accuracy	Discarded Samples by CFD	TSR Accuracy	TSR Availability	F-AA
AN	KNN	0.9908	763	0.9925	0.9927	0.9926
	SVM	0.9895	343	0.9898	0.9967	0.9933
	LDA	0.9885	1187	0.9933	0.9886	0.991
	Decision Tree	0.9854	1481	0.9932	0.9858	0.9895
	Boosting	0.9904	878	0.9928	0.9916	0.9922
	Random Forest	0.9907	588	0.9916	0.9944	0.9930
	Tree Bagger	0.9906	584	0.9915	0.9944	0.9930
IC3	KNN	0.9907	627	0.9924	0.994	0.9932
	SVM	0.9903	315	0.9907	0.997	0.9938
	LDA	0.9884	1289	0.9944	0.9876	0.9910
	Decision Tree	0.9863	1264	0.9932	0.9879	0.9905
	Boosting	0.9900	496	0.9914	0.9952	0.9933
	Random Forest	0.9908	453	0.9916	0.9957	0.9936
	Tree Bagger	0.9907	457	0.9916	0.9956	0.9936
MN2	KNN	0.9901	669	0.9916	0.9936	0.9926
	SVM	0.9893	349	0.9897	0.9967	0.9932
	LDA	0.9873	1283	0.9932	0.9877	0.9904
	Decision Tree	0.9845	1475	0.9927	0.9858	0.9892
	Boosting	0.9893	670	0.9913	0.9936	0.9924
	Random Forest	0.9900	523	0.9909	0.9950	0.9929
	Tree Bagger	0.9901	541	0.9910	0.9948	0.9929

Table 24: Overall TSR system performance on DITS dataset, for each augmented and CFD classifiers records are selected by highest F-AA calculated between accuracy and availability

Augmented Classifier	CFD Classifiers	CFD Accuracy	Discarded Samples by CFD	TSR Accuracy	TSR Availability	F-AA
AN	KNN	0.9511	1763	0.9585	0.9634	0.961
	SVM	0.9383	584	0.9407	0.9879	0.9637
	LDA	0.9396	2488	0.9597	0.9484	0.954
	Decision Tree	0.9391	3459	0.9694	0.9283	0.9484
	Boosting	0.9487	1803	0.9577	0.9626	0.9601
	Random Forest	0.9523	1506	0.9566	0.9688	0.9626
	Tree Bagger	0.952	1520	0.9566	0.9685	0.9625
IC3	KNN	0.9564	1177	0.9648	0.9756	0.9701
	SVM	0.9564	578	0.9589	0.988	0.9733
	LDA	0.9519	2364	0.9744	0.951	0.9626
	Decision Tree	0.9475	2597	0.9745	0.9461	0.9601
	Boosting	0.9567	1291	0.9661	0.9732	0.9696
	Random Forest	0.9611	1128	0.9667	0.9766	0.9716
	Tree Bagger	0.9613	1152	0.967	0.9761	0.9715
MN2	KNN	0.9658	638	0.9701	0.9868	0.9784
	SVM	0.9677	585	0.9705	0.9879	0.9791
	LDA	0.9663	1354	0.9776	0.9719	0.9747
	Decision Tree	0.9622	2013	0.9822	0.9582	0.9701
	Boosting	0.9682	1072	0.9756	0.9778	0.9767
	Random Forest	0.9711	895	0.9753	0.9814	0.9784
	Tree Bagger	0.971	899	0.9753	0.9814	0.9783

respectively and highest values are highlighted with bold fonts. The experimental results indicate that the higher availability of the system results in lower accuracy of TSR, and similarly when TSR accuracy is higher it ultimately lowers the availability of system.

We observed that highest accuracy is always achieved by LDA classifier considering three different augmented classifiers as shown with bold fonts in 5th column of Table 23, while in 6th column of Table 23, we highlighted with bold fonts the maximum availability of the TSR system that is always achieved by SVM. Combining the augmented classifier IC3 and CFD LDA provides the highest accuracy 0.9944, but it discards 1289 images, reducing the availability of the TSR system. Experimental results shows that augmented classifier IC3 achieves 0.9881 accuracy on BelgiumTSC dataset as shown in the 2nd column of Table 22, while the addition of CFD component, always achieve higher accuracy for any combination of augmented and CFD classifier with reduction in the availability of the TSR system as shown in Table 23.

Similarly, Table 24 shows the comparison of different CFD classifier in combination with each augmented classifier on DITS dataset. Experimental results shows that decision tree always Achieve highest accuracy for all three augmented classifiers. We achieved 0.9822 highest accuracy by using augmented classifier MN2 and CFD classifier decision tree that is higher compared to only exercising augmented classifier, where we achieve highest accuracy 0.9619 by MN2 as shown in Table 22. Interestingly, like BelgiumTSC dataset SVM always ensure maximum availability of the TSR system on DITS dataset. Decision tree discards 2013 images of DITS dataset, where MN2 ensure the maximum accuracy 0.9822, on the other side it reduce the availability of the TSR system 0.9582 that is low.

Experimental results show that for both BelgiumTSC and DITS dataset, exercising augmented classifier with CFD component increase the accuracy of the TSR system. Relying only on augmented classifiers ensures 100% availability, while in case of augmented classifier with CFD component the availability of the system is lower as shown in Table 23 and Table 24 for BelgiumTSC and DITS dataset respectively. Furthermore, a TSR system that embeds a CFD component may be computationally expensive, as the CFD adds workload to the TSR system.

Accuracy and availability are important that must be considered while implementing a TSR for a safety critical system [153]. Relying only on augmented classifiers (without failure detectors) for TSR is dangerous, a single misclassification may result in catastrophic situation. On the other hand, the addition of a failure detector increases the accuracy but reduces the availability of TSR. Depending on the requirements of system, one can choose among two strategies, if the availability of the system is more important one can implement TSR using only augmented classifiers that ensure 100% availability. If the requirement of the system is to ensure lower number of misclassifications that ultimately increase the accuracy of the TSR they can implement augmented classifiers in conjunction with CFD component that discard the corrupted images that result in high accuracy of TSR and ultimately reduce the availability of system.

6.6 Lessons Learned

In this chapter, we investigated how to improve the robustness of TSR system against visual camera failures. We presented two strategies such as data augmentation and the adoption of a camera failure detector that are effective to tolerate most failures of the visual camera.

Data augmentation improves the robustness of DNNs against visual camera failures, and also improving classification accuracy: especially, including dirt or scratched lens effects in the training set is raising accuracy up to 0.03 i.e., from 0.96 to 0.99 and 0.992 respectively with respect to regular classifiers on DITS original dataset. We showed that images altered using specific failures contribute more than others to improving the accuracy and robustness of classifiers. We observed that few visual camera failures such as brightness, and no noise reduction have more negative impact even using augmented classifiers. Still, to ensure robustness with respect to the entire failure set, it has been necessary to perform data augmentation enriching the training set with altered images due to multiple visual camera failures.

Furthermore, we exercised the second strategy i.e., master-slave strategy where CFD is to detect the images that are “too corrupted”, assuming that minor corrections are tolerated by the augmented classifier. In other words, we implemented an architectural approach where the images are either discarded by the CFD or are deemed of adequate quality for being processed by a robust classifier. Augmented classifiers in conjunction with CFD component improve the accuracy of TSR system compared to only relying on augmented classifier, but the higher the accuracy, the lower the availability of TSR system. We found that CFD classifier SVM always ensure maximum availability of the TSR system on both datasets, while LDA and decision tree always achieves maximum accuracy on BelgiumTSC and DITS dataset respectively.

7. CONCLUSIONS AND FUTURE DIRECTIONS

To conclude the thesis, we summarize the conclusion of the thesis and ultimately discuss future works.

7.1 *Conclusions*

In this thesis, we worked on a safety critical traffic sign recognition system. There are two main parts of this thesis, where in the first part we performed quantitative comparison of the state-of-the-art approaches and proposed a sliding window based approach for perfect traffic sign recognition. While in the second phase, we studied the behaviour of different TSR approaches against visual camera failures and different approaches are adopted for robust TSR against visual camera failures.

First, we conducted an extensive experimental comparison on available feature extractors and non-deep classifiers, accounting also for DNNs that can efficiently perform TSR through transfer learning. Results showed that K-NN, SVM and NN are the most adequate non-deep classifiers, and that HOG features provided actionable information to be processed by those algorithms. Overall, non-deep classifiers achieved the highest accuracy of 100% on GTSRB, 96.52% on BelgiumTSC and 93.61% on DITS. While the combination of feature descriptors such as HOG+SURF+LBP achieves highest accuracy 97.58% on BelgiumTSC and 95.71% on DITS dataset. We also compared those results with DNNs that were adapted to TSR through transfer learning and achieved higher accuracy than non-deep classifiers on DITS and BelgiumTSC i.e., 99.80% and 99.31% respectively. Next, we adopted sliding window approach for classification based on sequences of traffic sign images. Our study showed how the adoption of a stacking meta-learner in conjunction with sliding windows allows achieving perfect classification (100% accuracy) on the public GTSRB, BelgiumTSC and DITS datasets. Those datasets contain images taken in different parts of the world and mostly taken in semi-ideal lighting and environmental conditions. Therefore, they may not completely represent what a real TSR system installed on a vehicle will face during its life.

In the next phase, we analysed the effect of visual camera failures on the robustness of different TSR strategies. This study performs extensive experiments on three traffic sign datasets that are perturbed with 13 visual camera failures. Different classifiers are proposed, that are composed either to process a single image or a sliding window of images. We observe

that approaches based on sliding windows perform better compared to approaches based on single images, both on the clean datasets, and when failures are considered. Despite many classification strategies that we adopted for TSR, there is no single classification strategy that is more robust against the different visual camera failures. To such extent, we adopted different approaches such as data augmentation and camera failure detector to increase the robustness of DNNs against visual camera failures. We investigated how to improve the robustness of image classifiers against visual camera failures. We show that data augmentation is effective to tolerate most failures of the visual camera. We show that the same data augmentation approach is not only improving robustness, but is also improving classification accuracy: especially, including dirt or scratched lens effects in the training set is raising accuracy up to 0.03 i.e., from 0.96 to 0.99 and 0.992 respectively with respect to regular classifiers on DITS original dataset. We showed that images altered using specific failures contribute more than others to improving the accuracy and robustness of classifiers. Still, to ensure robustness with respect to the entire failure set, it has been necessary to perform data augmentation enriching the training set with altered images due to multiple visual camera failures.

To further improve the robustness of TSR, we trained a binary camera failure detector to exclude the images that are corrupted beyond the extent that our augmented classifier can classify properly. Our experimental results indicate that the addition of camera failure detector component with augmented classifiers increase the robustness of TSR compared to augmented classifiers at the cost of availability of system. As camera failure detector exclude the images that are corrupted beyond the extent that our augmented classifier can classify properly increase the accuracy of augmented classifier for TSR but reduce the availability of the system.

7.2 Future Works

In our future work, we will study the robustness of DNNs against adversarial patches and different adversarial attacks. We will explore other strategies for robust classification such as feature aggregation or architectural changes in DNNs. We are using benchmarks datasets; in future we are looking for companies that are working on (semi-)autonomous vehicles, where we can deploy our TSR system in real time to observe the real time performance. we will explore if our composite approaches that merges a failure detector and data augmentation is general enough to build safer image classifiers in safety critical domains other than Traffic Sign Recognition.

References

- [1] Mathias, M.; Timofte, R.; Benenson, R.; Van Gool, L. Traffic Sign Recognition—How Far Are We from the Solution? In Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–8.
- [2] Mammeri, A.; Boukerche, A.; Almulla, M. Design of Traffic Sign Detection, Recognition, and Transmission Systems for Smart Vehicles. *IEEE Wirel. Commun.* 2013, 20, 36–43. <https://doi.org/10.1109/MWC.2013.6704472>.
- [3] Li, J.; Wang, Z. Real-Time Traffic Sign Recognition Based on Efficient CNNs in the Wild. *IEEE Trans. Intell. Transp. Syst.* 2019, 20, 975–984. <https://doi.org/10.1109/TITS.2018.2843815>.
- [4] Avizienis, A.; Laprie, J.C.; Randell, B.; Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.* 2004, 1, 11–33. <https://doi.org/10.1109/TDSC.2004.2>.
- [5] Carvalho Barbosa, R.; Shoaib Ayub, M.; Lopes Rosa, R.; Zegarra Rodríguez, D.; Wuttisittikulkij, L. Lightweight PVIDNet: A Priority Vehicles Detection Network Model Based on Deep Learning for Intelligent Traffic Lights. *Sensors* 2020, 20, 6218. <https://doi.org/10.3390/s20216218>.
- [6] Application of Machine Learning Algorithms in Lane-Changing Model for Intelligent Vehicles Exiting to off-Ramp: *Transportmetrica A: Transport Science: Volume 17, No 1*. Available online: <https://www.tandfonline.com/doi/abs/10.1080/23249935.2020.1746861> (accessed on 18 March 2022).
- [7] Sasikala, G.; Ramesh Kumar, V. Development of Advanced Driver Assistance System Using Intelligent Surveillance. In *International Conference on Computer Networks and Communication Technologies*; Springer: Singapore, 2019; pp. 991–1003. https://doi.org/10.1007/978-981-10-8681-6_91.
- [8] Jose, A.; Thodupunoori, H.; Nair, B.B. A Novel Traffic Sign Recognition System Combining Viola–Jones Framework and Deep Learning. In *Soft Computing and Signal Processing*; Springer: Singapore, 2019; pp. 507–517. https://doi.org/10.1007/978-981-13-3600-3_48.

- [9] Kuo, C.Y.; Lu, Y.R.; Yang, S.M. On the Image Sensor Processing for Lane Detection and Control in Vehicle Lane Keeping Systems. *Sensors* 2019, 19, 1665. <https://doi.org/10.3390/s19071665>.
- [10] McDonald, A.; Carney, C.; McGehee, D.V. *Vehicle Owners' Experiences with and Reactions to Advanced Driver Assistance Systems*; 2018.
- [11] Hurtado, S.; Chiasson, S. An Eye-Tracking Evaluation of Driver Distraction and Unfamiliar Road Signs. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 24 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 153–160.
- [12] Wali, S. Comparative Survey on Traffic Sign Detection and Recognition: A Review. *Przegląd Elektrotechniczny* 2015, 1, 40–44. <https://doi.org/10.15199/48.2015.12.08>.
- [13] Guo, G.; Wang, H.; Bell, D.; Bi, Y.; Greer, K. *KNN Model-Based Approach in Classification*. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 986–996.
- [14] Hsu, C.W.; Lin, C.J. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Trans. Neural Netw.* 2002, 13, 415–425. <https://doi.org/10.1109/72.991427>.
- [15] Wahyono; Jo, K.H. A Comparative Study of Classification Methods for Traffic Signs Recognition. In *Proceedings of the 2014 IEEE International Conference on Industrial Technology (ICIT)*, Busan, Korea, 26 February 2014; pp. 614–619.
- [16] Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* 2002, 24, 971–987. <https://doi.org/10.1109/TPAMI.2002.1017623>.
- [17] Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.

- [18] Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Netw.* 2012, 32, 323–332. <https://doi.org/10.1016/j.neunet.2012.02.016>.
- [19] Yang, X.; Qu, Y.; Fang, S. Color Fused Multiple Features for Traffic Sign Recognition. In *Proceedings of the 4th International Conference on Internet Multimedia Computing and Service*, 9 September 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 84–87.
- [20] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
- [21] Manisha, U.K.D.N.; Liyanage, S.R. An Online Traffic Sign Recognition System for Intelligent Driver Assistance. In *Proceedings of the 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Colombo, Sri Lanka, 6–9 September 2017; pp. 1–6.
- [22] Matoš, I.; Krpić, Z.; Romić, K. The Speed Limit Road Signs Recognition Using Hough Transformation and Multi-Class Svm. In *Proceedings of the 2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Osijek, Croatia, 5–7 June 2019; pp. 89–94.
- [23] Liu, C.; Li, S.; Chang, F.; Wang, Y. Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives. *IEEE Access* 2019, 7, 86578–86596. <https://doi.org/10.1109/ACCESS.2019.2924947>.
- [24] Soni, D.; Chaurasiya, R.K.; Agrawal, S. Improving the Classification Accuracy of Accurate Traffic Sign Detection and Recognition System Using HOG and LBP Features and PCA-Based Dimension Reduction; Social Science Research Network: Rochester, NY, USA, 2019.
- [25] Hasan, N.; Anzum, T.; Jahan, N. Traffic Sign Recognition System (TSRS): SVM and Convolutional Neural Network. In *Inventive Communication and Computational*

Technologies; Springer: Singapore, 2021; pp. 69–79. https://doi.org/10.1007/978-981-15-7345-3_6.

- [26] Rahmad, C.; Rahmah, I.F.; Asmara, R.A.; Adhisuwignjo, S. Indonesian Traffic Sign Detection and Recognition Using Color and Texture Feature Extraction and SVM Classifier. In Proceedings of the 2018 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 6–7 March 2018; pp. 50–55.
- [27] Cao, J.; Song, C.; Peng, S.; Xiao, F.; Song, S. Improved Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicles. *Sensors* 2019, 19, 4021. <https://doi.org/10.3390/s19184021>.
- [28] Li, W.; Li, D.; Zeng, S. Traffic Sign Recognition with a Small Convolutional Neural Network. *IOP Conf. Ser. Mater. Sci. Eng.* 2019, 688, 044034. <https://doi.org/10.1088/1757-899X/688/4/044034>.
- [29] Atif, M.; Zoppi, T.; Gharib, M.; Bondavalli, A. Quantitative Comparison of Supervised Algorithms and Feature Sets for Traffic Sign Recognition. In Proceedings of the 36th Annual ACM Symposium on Applied Computing; Association for Computing Machinery: Gwangju, South Korea, 2021; March 22-26, 2021, pp. 174–177. <https://doi.org/10.1145/3412841.3442072>
- [30] Vilalta, R.; Drissi, Y. A Perspective View and Survey of Meta-Learning. *Artif. Intell. Rev.* 2002, 18, 77–95. <https://doi.org/10.1023/A:1019956318069>.
- [31] Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 1453–1460.
- [32] Timofte, R.; Zimmermann, K.; Van Gool, L. Multi-View Traffic Sign Detection, Recognition, and 3D Localisation. *Mach. Vis. Appl.* 2014, 25, 633–647. <https://doi.org/10.1007/s00138-011-0391-3>.

- [33] Youssef, A.; Albani, D.; Nardi, D.; Bloisi, D.D. Fast Traffic Sign Recognition Using Color Segmentation and Deep Convolutional Networks. In *Advanced Concepts for Intelligent Vision Systems*; Springer: Cham, 2016; pp. 205–216.
- [34] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2012; Volume 25.
- [35] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*; pp. 770–778.
- [36] Carbonell, J.G.; Michalski, R.S.; Mitchell, T.M. 1—An Overview of Machine Learning. In *Machine Learning*; Michalski, R.S., Carbonell, J.G., Mitchell, T.M., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1983; pp. 3–23; ISBN 978-0-08-051054-5.
- [37] Breiman, L. Random Forests. *Mach. Learn.* 2001, 45, 5–32. <https://doi.org/10.1023/A:1010933404324>.
- [38] Fisher, R.A. The Use of Multiple Measurements in Taxonomic Problems. *Ann. Eugen.* 1936, 7, 179–188. <https://doi.org/10.1111/j.1469-1809.1936.tb02137.x>.
- [39] Freund, Y. A More Robust Boosting Algorithm. arXiv 2009, arXiv:0905.2138.
- [40] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016*; pp. 2818–2826.
- [41] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018*; pp. 4510–4520.
- [42] Lam, L.; Suen, S.Y. Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Hum.* 1997, 27, 553–568. <https://doi.org/10.1109/3468.618255>.

- [43] Yasuda, H.; Takahashi, K.; Matsumoto, T. A Discrete HMM for Online Handwriting Recognition. *Int. J. Pattern Recognit. Artif. Intell.* 2000, 14, 675–688. <https://doi.org/10.1142/S021800140000043X>.
- [44] Huang, Z.; Yu, Y.; Gu, J. A Novel Method for Traffic Sign Recognition Based on Extreme Learning Machine. In *Proceedings of the 11th World Congress on Intelligent Control and Automation, Shenyang, China, 29 June–4 July 2014*; pp. 1451–1456.
- [45] Agrawal, S.; Chaurasiya, R.K. Ensemble of SVM for Accurate Traffic Sign Detection and Recognition. In *Proceedings of the International Conference on Graphics and Signal Processing, 24 June 2017*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 10–15.
- [46] Myint, T.; Thida, L. Real-Time Myanmar Traffic Sign Recognition System Using HOG and SVM. *Int. J. Trend Sci. Res. Dev.* 2019, 3.
- [47] Abedin, M.Z.; Dhar, P.; Deb, K. Traffic Sign Recognition Using SURF: Speeded up Robust Feature Descriptor and Artificial Neural Network Classifier. In *Proceedings of the 2016 9th International Conference on Electrical and Computer Engineering (ICECE), Dhaka, Bangladesh, 20–22 December 2016*; pp. 198–201.
- [48] Farag, W. Traffic Signs Classification by Deep Learning for Advanced Driving Assistance Systems. *Intell. Decis. Technol.* 2019, 13, 305–314. <https://doi.org/10.3233/IDT-180064>.
- [49] Kamal, U.; Tonmoy, T.I.; Das, S.; Hasan, M.K. Automatic Traffic Sign Detection and Recognition Using SegU-Net and a Modified Tversky Loss Function With L1-Constraint. *IEEE Trans. Intell. Transp. Syst.* 2020, 21, 1467–1479. <https://doi.org/10.1109/TITS.2019.2911727>.
- [50] Liu, C.; Chang, F.; Chen, Z.; Liu, D. Fast Traffic Sign Recognition via High-Contrast Region Extraction and Extended Sparse Representation. *IEEE Trans. Intell. Transp. Syst.* 2016, 17, 79–92. <https://doi.org/10.1109/TITS.2015.2459594>.
- [51] Lin, C.; Li, L.; Luo, W.; Wang, K.C.P.; Guo, J. Transfer Learning Based Traffic Sign Recognition Using Inception-v3 Model. *Period. Polytech. Transp. Eng.* 2019, 47, 242–250. <https://doi.org/10.3311/PPtr.11480>.

- [52] Zaki, P.S.; William, M.M.; Soliman, B.K.; Alexsan, K.G.; Khalil, K.; El-Moursy, M. Traffic Signs Detection and Recognition System Using Deep Learning. arXiv 2020, arXiv:2003.03256.
- [53] Abdel-Salam, R.; Mostafa, R.; Abdel-Gawad, A.H. RIECNN: Real-Time Image Enhanced CNN for Traffic Sign Recognition. *Neural Comput. Appl.* 2022, 34, 6085–6096. <https://doi.org/10.1007/s00521-021-06762-5>.
- [54] Mangshor, N.N.A.; Paudzi, N.P.A.M.; Ibrahim, S.; Sabri, N. A Real-Time Malaysian Traffic Sign Recognition Using YOLO Algorithm. In *Proceedings of the 12th National Technical Seminar on Unmanned System Technology 2020*; Isa, K., Zain, M.Z., Mohd-Mokhtar, R., Mat Noh, M., Ismail, Z.H., Yusof, A.A., Mohamad Ayob, A.F., Azhar Ali, S.S., Abdul Kadir, H., Eds.; Springer: Singapore, 2022; pp. 283–293.
- [55] Naim, S.; Moumquine, N. LiteNet: A Novel Approach for Traffic Sign Classification Using a Light Architecture. In *Proceedings of the WITS 2020*; Bennani, S., Lakhrissi, Y., Khaissidi, G., Mansouri, A., Khamlichi, Y., Eds.; Springer: Singapore, 2022; pp. 37–47.
- [56] Nartey, O.T.; Yang, G.; Asare, S.K.; Wu, J.; Frempong, L.N. Robust Semi-Supervised Traffic Sign Recognition via Self-Training and Weakly-Supervised Learning. *Sensors* 2020, 20, 2684. <https://doi.org/10.3390/s20092684>.
- [57] Vennelakanti, A.; Shreya, S.; Rajendran, R.; Sarkar, D.; Muddegowda, D.; Hanagal, P. Traffic Sign Detection and Recognition Using a CNN Ensemble. In *Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 11–13 January 2019; pp. 1–4.
- [58] Lu, X.; Wang, Y.; Zhou, X.; Zhang, Z.; Ling, Z. Traffic Sign Recognition via Multi-Modal Tree-Structure Embedded Multi-Task Learning. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 960–972. <https://doi.org/10.1109/TITS.2016.2598356>.
- [59] Bi, Z.; Yu, L.; Gao, H.; Zhou, P.; Yao, H. Improved VGG Model-Based Efficient Traffic Sign Recognition for Safe Driving in 5G Scenarios. *Int. J. Mach. Learn. Cybern.* 2021, 12, 3069–3080. <https://doi.org/10.1007/s13042-020-01185-5>.
- [60] Bayouhd, K.; Hamdaoui, F.; Mtibaa, A. Transfer Learning Based Hybrid 2D–3D CNN for Traffic Sign Recognition and Semantic Road Detection Applied in Advanced Driver

Assistance Systems. *Appl. Intell.* 2021, 51, 124–142. <https://doi.org/10.1007/s10489-020-01801-5>.

- [61] Sermanet, P.; LeCun, Y. Traffic Sign Recognition with Multi-Scale Convolutional Networks. In Proceedings of the The 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 2809–2813.
- [62] Yang, Y.; Luo, H.; Xu, H.; Wu, F. Towards Real-Time Traffic Sign Detection and Classification. *IEEE Trans. Intell. Transp. Syst.* 2016, 17, 2022–2031. <https://doi.org/10.1109/TITS.2015.2482461>.
- [63] Islam, K.T.; Raj, R.G. Real-Time (Vision-Based) Road Sign Recognition Using an Artificial Neural Network. *Sensors* 2017, 17, 853. <https://doi.org/10.3390/s17040853>.
- [64] Luo, H.; Yang, Y.; Tong, B.; Wu, F.; Fan, B. Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network. *IEEE Trans. Intell. Transp. Syst.* 2018, 19, 1100–1111. <https://doi.org/10.1109/TITS.2017.2714691>.
- [65] Yuan, Y.; Xiong, Z.; Wang, Q. An Incremental Framework for Video-Based Traffic Sign Detection, Tracking, and Recognition. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 1918–1929. <https://doi.org/10.1109/TITS.2016.2614548>.
- [66] Park, J.; Lee, K.; Kim, H.Y. Recognition Assistant Framework Based on Deep Learning for Autonomous Driving: Restoring Damaged Road Sign Information; Social Science Research Network: Rochester, NY, USA, 2022.
- [67] Zakir Hussain, K.M.; Kattigenahally, K.N.; Nikitha, S.; Jena, P.P.; Harshalatha, Y. Traffic Symbol Detection and Recognition System. In Proceedings of the Emerging Research in Computing, Information, Communication and Applications; Shetty, N.R., Patnaik, L.M., Nagaraj, H.C., Hamsavath, P.N., Nalini, N., Eds.; Springer: Singapore, 2022; pp. 885–897.
- [68] Lahmyed, R.; Ansari, M.E.; Kerkaou, Z. Automatic Road Sign Detection and Recognition Based on Neural Network. *Soft Comput.* 2022, 26, 1743–1764. <https://doi.org/10.1007/s00500-021-06726-w>.

- [69] Gautam, S.; Kumar, A. Automatic Traffic Light Detection for Self-Driving Cars Using Transfer Learning. In Proceedings of the Intelligent Sustainable Systems; Nagar, A.K., Jat, D.S., Marín-Raventós, G., Mishra, D.K., Eds.; Springer: Singapore, 2022; pp. 597–606.
- [70] Schuszter, I.C. A Comparative Study of Machine Learning Methods for Traffic Sign Recognition. In Proceedings of the 2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 21–24 September 2017; pp. 389–392.
- [71] Brazdil, P.; Carrier, C.G.; Soares, C.; Vilalta, R. *Metalearning: Applications to Data Mining*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008; ISBN 978-3-540-73262-4.
- [72] Vanschoren, J. *Understanding Machine Learning Performance with Experiment Databases (Het Verwerven van Inzichten in Leerperformantie Met Experiment Databanken)*; 2010.
- [73] Wolpert, D.H. Stacked Generalization. *Neural Netw.* 1992, 5, 241–259. [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1).
- [74] Breiman, L. Bagging Predictors. *Mach. Learn.* 1996, 24, 123–140. <https://doi.org/10.1007/BF00058655>.
- [75] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* 1997, 9, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [76] Li, Y.; Zhu, Z.; Kong, D.; Han, H.; Zhao, Y. EA-LSTM: Evolutionary Attention-Based LSTM for Time Series Prediction. *Knowl.-Based Syst.* 2019, 181, 104785. <https://doi.org/10.1016/j.knosys.2019.05.028>.
- [77] Sokolova, M.; Lapalme, G. A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manag.* 2009, 45, 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>.

- [78] Mortaz, E. Imbalance Accuracy Metric for Model Selection in Multi-Class Imbalance Classification Problems. *Knowl.-Based Syst.* 2020, 210, 106490. <https://doi.org/10.1016/j.knosys.2020.106490>.
- [79] Chamasemani, F.F.; Singh, Y.P. Multi-Class Support Vector Machine (SVM) Classifiers—An Application in Hypothyroid Detection and Classification. In *Proceedings of the 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, Penang, Malaysia, 27–29 September 2011; pp. 351–356.
- [80] Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
- [81] Popov, G.; Raynova, K. Diversity in Nature and Technology—Tool for Increase the Reliability of Systems. In *Proceedings of the 2017 15th International Conference on Electrical Machines, Drives and Power Systems (ELMA)*, Sofia, Bulgaria, 1–3 June 2017; pp. 464–466.
- [82] Shang, R.; Xu, K.; Shang, F.; Jiao, L. Sparse and Low-Redundant Subspace Learning-Based Dual-Graph Regularized Robust Feature Selection. *Knowl.-Based Syst.* 2020, 187, 104830. <https://doi.org/10.1016/j.knosys.2019.07.001>.
- [83] Zeng, Y.; Xu, X.; Shen, D.; Fang, Y.; Xiao, Z. Traffic Sign Recognition Using Kernel Extreme Learning Machines With Deep Perceptual Features. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 1647–1653. <https://doi.org/10.1109/TITS.2016.2614916>.
- [84] Jiménez, Á.B.; Lázaro, J.L.; Dorronsoro, J.R. Finding Optimal Model Parameters by Discrete Grid Search. In *Innovations in Hybrid Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 120–127. https://doi.org/10.1007/978-3-540-74972-1_17.
- [85] Secci, F.; Ceccarelli, A. On Failures of RGB Cameras and Their Effects in Autonomous Driving Applications. In *Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, Coimbra, Portugal, 12–15 October 2020; pp. 13–24.

- [86] J. Levinson et al., “Towards fully autonomous driving: Systems and algorithms,” in 2011 IEEE Intelligent Vehicles Symposium (IV), Jun. 2011, pp. 163–168. doi: 10.1109/IVS.2011.5940562.
- [87] Atif, M., Zoppi, T., Gharib, M., & Bondavalli, A. (2022). Towards enhancing traffic sign recognition through sliding windows. *Sensors*, 22(7), 2683.
- [88] Ceccarelli, A., & Secci, F. (2022). RGB cameras failures and their effects in autonomous driving applications. *IEEE Transactions on Dependable and Secure Computing*.
- [89] J. H. Chung, D. W. Kim, T. K. Kang, and M. T. Lim, “Traffic Sign Recognition in Harsh Environment Using Attention Based Convolutional Pooling Neural Network,” *Neural Process. Lett.*, vol. 51, no. 3, pp. 2551–2573, Jun. 2020, doi: 10.1007/s11063-020-10211-0.
- [90] C. Collet and O. Musicant, “Associating Vehicles Automation With Drivers Functional State Assessment Systems: A Challenge for Road Safety in the Future,” *Front. Hum. Neurosci.*, vol. 13, 2019, Accessed: Feb. 16, 2022. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fnhum.2019.00131>
- [91] R. S. Ferreira, J. Arlat, J. Guiochet, and H. Waeselynck, “Benchmarking Safety Monitors for Image Classifiers with Machine Learning,” in 2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC), Dec. 2021, pp. 7–16. doi: 10.1109/PRDC53464.2021.00012.
- [92] M. T. Ribeiro, S. Singh, and C. Guestrin, ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, Aug. 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.
- [93] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, “1 - AN OVERVIEW OF MACHINE LEARNING,” in *Machine Learning*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. San Francisco (CA): Morgan Kaufmann, 1983, pp. 3–23. doi: 10.1016/B978-0-08-051054-5.50005-4.

- [94] “Realistic Lens Blur/Chromatic Aberration Filter, <https://github.com/yoonsikp/kromo>.”
- [95] “ActionVFX, <https://www.actionvfx.com/>.”
- [96] D. Baumgartner, P. Roessler, W. Kubinger, C. Zinner, and K. Ambrosch, “Benchmarks of Low-Level Vision Algorithms for DSP, FPGA, and Mobile PC Processors,” *Embed. Comput. Vis.*, pp. 101–120, 2009, doi: 10.1007/978-1-84800-304-0_5.
- [97] D. A. van Dyk and X.-L. Meng, “The Art of Data Augmentation,” *J. Comput. Graph. Stat.*, vol. 10, no. 1, pp. 1–50, Mar. 2001, doi: 10.1198/10618600152418584.
- [98] Bani Hashem, B. H., & Ozeki, T. (2015, October). Pedestrian Detection by Using FAST-HOG Features. In *Proceedings of the 3rd International Conference on Human-Agent Interaction* (pp. 277-278).
- [99] Hossin, Mohammad, and M. N. Sulaiman. "A review on evaluation metrics for data classification evaluations." *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015): 1.
- [100] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
- [101] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, “Adversarially Robust Generalization Requires More Data,” in *Advances in Neural Information Processing Systems*, 2018, vol. 31. Accessed: Apr. 07, 2022.
- [102] Chen, X. W., & Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE access*, 2, 514-525.
- [103] Song, Hwanjun, et al. "Learning from noisy labels with deep neural networks: A survey." *IEEE Transactions on Neural Networks and Learning Systems* (2022).
- [104] Sietsma, J., & Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural networks*, 4(1), 67-79.
- [105] Lawrence, S., & Giles, C. L. (2000, July). Overfitting and neural networks: conjugate gradient and backpropagation. In *Proceedings of the IEEE-INNS-ENNS International*

- Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 1, pp. 114-119). IEEE.
- [106] Prechelt, L. (1998). Early stopping-but when?. In *Neural Networks: Tricks of the trade* (pp. 55-69). Springer, Berlin, Heidelberg.
- [107] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting Batch Normalization For Practical Domain Adaptation," *ArXiv160304779 Cs*, Nov. 2016, Accessed: Apr. 07, 2022. [Online]. Available: <http://arxiv.org/abs/1603.04779>
- [108] I. Jindal, M. Nokleby, and X. Chen, "Learning Deep Networks from Noisy Labels with Dropout Regularization," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, Dec. 2016, pp. 967–972. doi: 10.1109/ICDM.2016.0121.
- [109] Caruana, R., Lawrence, S., & Giles, C. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 13.
- [110] Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4.
- [111] Song, Hwanjun, et al. "Learning from noisy labels with deep neural networks: A survey." *IEEE Transactions on Neural Networks and Learning Systems* (2022).
- [112] Sietsma, J., & Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural networks*, 4(1), 67-79.
- [113] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018, doi: 10.1109/ACCESS.2018.2807385.
- [114] Gopinath, D., Zhang, M., Wang, K., Kadron, I. B., Pasareanu, C., & Khurshid, S. (2019, October). Symbolic execution for importance analysis and adversarial generation in neural networks. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 313-322). IEEE.

- [115] Premebida, Cristiano, Gledson Melotti, and Alireza Asvadi. "RGB-D Object Classification for Autonomous Driving Perception." *RGB-D Image Analysis and Processing*. Springer, Cham, 2019. 377-395.
- [116] S. Ghosh, R. Shet, P. Amon, A. Hutter, and A. Kaup, "Robustness of Deep Convolutional Neural Networks for Image Degradations," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Apr. 2018, pp. 2916–2920. doi: 10.1109/ICASSP.2018.8461907.
- [117] C. Gong, T. Ren, M. Ye, and Q. Liu, "MaxUp: Lightweight Adversarial Training With Data Augmentation Improves Neural Network Training," 2021, pp. 2474–2483.
- [118] A. Agarwal, M. Vatsa, R. Singh, and N. Ratha, "Cognitive data augmentation for adversarial defense via pixel masking," *Pattern Recognit. Lett.*, vol. 146, pp. 244–251, Jun. 2021, doi: 10.1016/j.patrec.2021.01.032.
- [119] Z. Fabian, R. Heckel, and M. Soltanolkotabi, "Data augmentation for deep learning based accelerated MRI reconstruction with limited data," in *Proceedings of the 38th International Conference on Machine Learning*, Jul. 2021, pp. 3057–3067.
- [120] A. Morozov, E. Valiev, M. Beyer, K. Ding, L. Gauerhof, and C. Schorn, "Bayesian Model for Trustworthiness Analysis of Deep Learning Classifiers," 2020, p. 7
- [121] Matas, J. and Chum, O. and Urban, M. and Pajdla, T. 2002. «Robust Wide Baseline Stereo from Maximally Stable Extremal Regions.» *Proceedings of the British Machine Vision Conference*. BMVA Press. 36.1-36.10.
- [122] Harris, C., and M. J. Stephens. 1988. «A Combined Corner and Edge Detector.» *Proceedings of the 4th Alvey Vision Conference*. 147–152.
- [123] Shi, J., and C. Tomasi. 1994. «Good Features to Track.» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 593–600.
- [124] Rosten, E., and T. Drummond. 2006. «Machine Learning for High-Speed Corner Detection.» *9th European Conference on Computer Vision*. 430–443.

- [125] S. Leutenegger, M. Chli and R. Y. Siegwart. 2011. «BRISK: Binary Robust invariant scalable keypoints.» International Conference on Computer Vision. Barcelona. 2548-2555.
- [126] Alahi, Alexandre and Ortiz, Raphael and Vandergheynst, Pierre. 2012. «Freak: Fast retina keypoint.» 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 510--517.
- [127] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. 2008. «Speeded-Up Robust Features (SURF).» Computer vision and image understanding 346-359.
- [128] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Proceedings of the 2001 IEEE conference on computer vision and pattern recognition. CVPR 2001. Vol. 1. IEEE, 2001.
- [129] Alcantarilla P.F., Bartoli A., Davison A.J. 2012. «KAZE Features.» European Conference on Computer Vision. Springer, Berlin, Heidelberg. 214-227.
- [130] Anderson, Dave and McNeill, George. 1992. «Artificial neural networks technology.» Kaman Sciences Corporation 1--83.
- [131] Warsito, Budi and Santoso, Rukun and Yasin, Hasbi and others. 2018. «Cascade Forward Neural Network for Time Series Prediction.» Journal of Physics: Conference Series. IOP Publishing. 012097.
- [132] Ali, Shawkat and Smith, Kate A. 2006. «On learning algorithm selection for classification.» Applied Soft Computing 119–138.
- [133] Lei Ba, J.; Kingma, D.P. Adam: A method for stochastic gradient descent. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
- [134] Liu, Y.; Yuan, G.; Wotao, Y. An improved analysis of stochastic gradient descent with momentum. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020.
- [135] Dogo, E.M.; Afolabi, O.J.; Nwulu, N.I.; Twala, B.; Aigbavboa, C.O. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural

- networks. In Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Belgaum, India, 21–22 December 2018; pp. 92–99.
- [136] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. on Dependable and Secure Computing*, 1(1):11–33, 2004.
- [137] Phillips, J. B., & Eliasson, H. (2018). *Camera image quality benchmarking*. John Wiley & Sons.
- [138] Hogan, K. (2014, October). Options for Camera Raw in the Digital Workflow. In *SMPTE 2014 Annual Technical Conference & Exhibition* (pp. 1-18).
- [139] Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.
- [140] Gu, J., Ramamoorthi, R., Belhumeur, P., & Nayar, S. (2009). Removing image artifacts due to dirty camera lenses and thin occluders. In *ACM SIGGRAPH Asia 2009* (pp. 1-10).
- [141] Hossain, S., Fayjie, A. R., Doukhi, O., & Lee, D. J. (2018, October). CAIAS simulator: self-driving vehicle simulator for AI research. In *International Conference on Intelligent Computing & Optimization* (pp. 187-195). Springer.
- [142] Drenkow, N., Sani, N., Shpitser, I., & Unberath, M. (2021). Robustness in Deep Learning for Computer Vision: Mind the gap?. *arXiv preprint arXiv:2112.00639*.
- [143] Naveed, H. (2021). Survey: Image mixing and deleting for data augmentation. *arXiv preprint arXiv:2106.07085*.
- [144] Stocco, A., Weiss, M., Calzana, M., & Tonella, P. (2020, June). Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering* (pp. 359-371).
- [145] Henriksson, J., Berger, C., Borg, M., Tornberg, L., Sathyamoorthy, S. R., & Englund, C. (2019, August). Performance analysis of out-of-distribution detection on various trained neural networks. In *SEAA* (pp. 113-120). IEEE.

- [146] Hsu, Y. C., Shen, Y., Jin, H., & Kira, Z. (2020). Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10951-10960).
- [147] Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., & Stanley, M. (2017, August). A brief survey of machine learning methods and their sensor and IoT applications. In 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
- [148] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
- [149] Bradski, G., & Kaehler, A. (2000). OpenCV. Dr. Dobb's journal of software tools, 3, 120
- [150] CV2 filtering, <https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>
- [151] PIL 3.0, <https://pillow.readthedocs.io/en/3.0.x/> [online].
- [152] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.
- [153] Rausand, M. (2014). Reliability of safety-critical systems: theory and applications. John Wiley & Son