Towards Trustworthy AI:

Understanding the Impact of AI Threats and Countermeasures

Giulio Rossolini

SOCINT Press

Società Italiana di Intelligence
c/o Università della Calabria,
Cubo 18-b, 7° piano
via Pietro Bucci – 87036
Arcavacata di Rende (CS) -Italia
https://www.socint.org
ISBN 979-12-80111-73-9

Sant'Anna
Scuola Universitaria Superiore Pisa

Ph.D. in Emerging Digital Technologies

Towards Trustworthy AI: Understanding the Impact of Threats and Countermeasures

Author:
Giulio Rossolini

Supervisor:
Prof. Giorgio Carlo Buttazzo

Tutor:
Prof. Alessandro Biondi

Anno accademico 2022/2023

Towards Trustworthy AI: Understanding the Impact of Threats and Countermeasures

Giulio Rossolini

"Any technological advance can be dangerous. Fire was dangerous from the start, and so (even more so) was speech - and both are still dangerous to this day - but human beings would not be human without them."

- Isaac Asimov

Abstract

The rapid advancements in AI, particularly in deep neural networks (DNNs), have prompted the research community to face complex safety and security challenges, which must be carefully addressed to ensure the correct integration of AI algorithms into human-centric systems. AI threats can range from intentionally-crafted samples, such as adversarial perturbations or real-world adversarial objects, to unexpected out-of-distribution samples. The presence of these threats raises numerous questions and considerations about the security vulnerabilities and safety requirements of the models and applications under analysis. Accordingly, it is crucial to thoroughly understand and design testing methodologies and mitigation strategies, taking into account specific aspects and requirements of each application scenario.

This thesis delves into the domain of AI threats and countermeasures, with a specific focus on computer vision applications in safety-critical environments like cyber-physical systems, autonomous robots, and self-driving cars. The main research areas explored in the thesis, within the context of trustworthy AI, include DNN testing and the design of novel real-world attacks and defenses in complex outdoor scenarios.

Firstly, the thesis critically examines the landscape of DNN testing, with a particular focus on the application of coverage criteria, a concept adapted from the field of software engineering. In this context, we introduce a framework designed to utilize coverage criteria for monitoring the behavior of neural networks at run-time. This offers a novel methodological perspective on leveraging testing techniques to assess model behavior. Through an analysis of state-of-the-art approaches in coverage testing and the results obtained, the thesis dedicates significant effort to paving the way for future research directions.

Then, in the realm of real-world adversarial attacks, the thesis reviews the literature on attack and defense strategies, highlighting the gaps in analysis for certain computer vision tasks and applications. Additionally, concerning the understanding of state-of-the-art defense mechanisms, the review underscores an insufficient awareness of the practical implications of current defense mechanisms when applied in safety-critical scenarios. Following these observations, the work focuses on developing real-world attacks against semantic segmentation tasks, providing a clear interpretation of the spatial robustness of DNNs. The proposed attack methodology is achieved through novel optimization approaches and the utilization of driving simulators. Subsequently, the thesis presents an in-depth study of novel interpretable defense mechanisms, founded on provable and robust analysis.

In conclusion, this thesis offers an examination of AI threats from different perspectives, merging theoretical discussions with practical applications. It aims at expanding and reviewing the existing literature, stimulating further research, and enhancing the understanding of AI safety and security threats.

Contents

Chapter 1

Introduction

The rapid advancements in artificial intelligence (AI) have demonstrated superhuman performance in solving vision and language problems, such as complex scene understanding [1, 2, 3], question answering [4, 5], control [6, 7], and content generation [8]. Supported by the continuous growth in performance and research efforts, the wave of AI has also captured the interest of the tech industry, pushing towards the integration of learning-enabled algorithms, such as deep neural networks (DNNs), into complex cyber-physical systems.

However, integrating advanced AI algorithms into human-centric systems presents significant safety and security challenges. For instance, the deployment of complex DNNs in autonomous systems, such as robots or self-driving vehicles, must meet rigorous safety and reliability standards. In this context, the poor interpretability of DNNs makes the development of sophisticated methods and analysis crucial for promoting a safe AI integration where human safety is paramount.

Indeed, a critical ongoing challenge in incorporating AI into these systems lies in proving and enhancing the trustworthiness of large and complex models. This raises several research questions: Can we fully trust machine learning algorithms? How can we effectively test and evaluate the robustness and accuracy of AI predictions? Are these systems vulnerable to cyber attacks and what actions should we take in case of an attack? Furthermore, given potential threats, how feasible is it to develop supporting algorithms that improve model trustwortiness, while being computationally efficient? Addressing these concerns require facing a range of issues across different software and hardware domains, including security, safety, explainability, and predictability.

This Ph.D. thesis explores these questions from multiple perspectives, focusing on AI threats in the realm of computer vision tasks for safety-critical applications.

This section provides a top-level overview of the main threats addressed and outlines the structure and scope of the thesis.

AI Threats in Computer Vision

To improve the trustworthiness and robustness of recent AI models, various threats have been identified and studied in recent years. These threats are often depicted as input samples that produce unexpected outcomes, which need to be exhaustively identified and analyzed.

Many samples that challenge the trustworthiness and robustness of AI systems can

be categorized into *adversarial samples* (also known as adversarial examples) and *out-of-distribution (OOD)* samples. Figure 1.1 illustrates these concepts, with further explanations provided below.

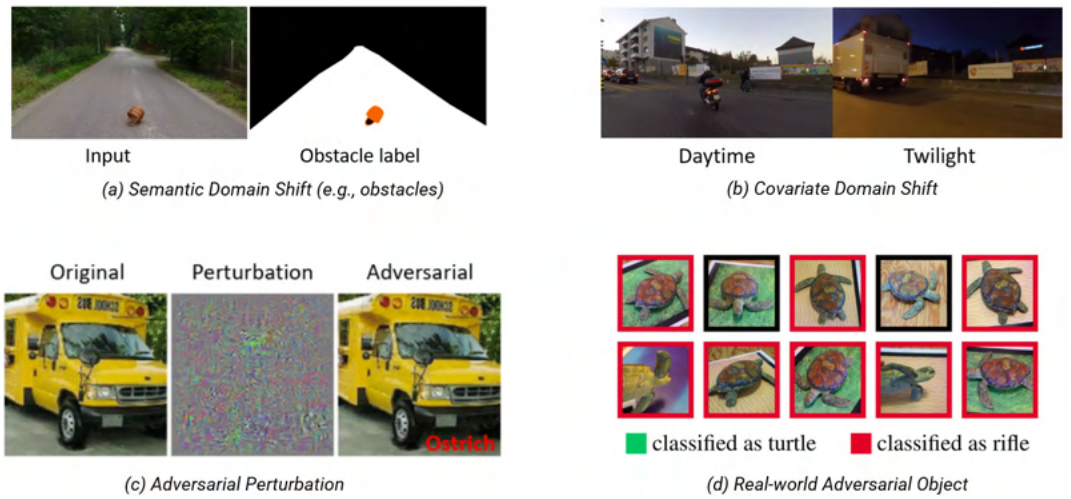


Figure 1.1: Illustration of highly studied computer vision threats in the AI era: Figure (a) and (b) present examples of out-of-distribution samples in the case of semantic shift (image from [9]) and covariate domain shift ([10]), respectively. Figure (c) illustrates adversarial perturbations [11, 12], and (d) shows real-world adversarial objects [13].

Out-of-distribution (OOD) samples, in the field of computer vision, refer to input data that semantically or contextually differ from those used in model training, meaning they fall outside the statistical distribution of the training set. For example, if a neural network is trained on images of urban scenarios, an OOD sample might be an image set in a nighttime landscape. This represents a domain shift in the appearance of the objects, even though they retain a known semantic meaning. Conversely, another form of out-of-distribution samples could refer to a semantic drift, such as depicting an object never encountered during training (e.g., an obstacle in a driving scenario). Although there is a vast and growing literature on out-of-distribution detection in computer vision [14, 15], this thesis primarily focuses on adversarial examples.

Adversarial examples are intentionally crafted inputs, designed through specific optimization strategies, to mislead the model predictions. For the objective of this thesis, it is important to differentiate between two types of adversarial examples: *adversarial perturbations* and *real-world adversarial objects*.

- *Adversarial perturbations* have been extensively explored in literature [16, 11, 17]. These are subtle, human-imperceptible, modifications of the input sample that alter model outputs by activating specific patterns.
- *Real-world adversarial objects*, in contrast, deceive the model in a different manner. Unlike digital pixel-level perturbations, these are physical objects, such as patches or 3D meshed objects, that fool the model outcome when captured by the camera sensor.

While both types fall under the field of adversarial attacks, their distinct natures lead to different studies and attack scenarios.

Distinguishing Safety and Security in Adversarial Attacks

While adversarial attacks undeniably present a tangible threat to DNNs, there is an ongoing debate about their practical significance in terms of safety and security [18, 19].

In the realm of cyber-physical systems, such as self-driving vehicles and robots, it is not entirely realistic to assume threat models where attackers have direct access to the digital representation of frames captured by a vision system [20, 21, 22, 23]. Furthemore, assuming the attacker can easily control the digital representation of the input, it overlooks other potential attacks that could be more effective than complex and time-consuming adversarial perturbations. For example, an attacker could introduce NaN pixels or unbounded perturbations of the image.

Nonetheless, it is important to note that adversarial perturbations constrained under a magnitude ϵ represent a well-defined approximation of a worst-case noise scenario. This implies that proving the model robustness against ϵ-norm perturbations can lead to a certified robustness of the model against any forms of ϵ-noise. Therefore, in the realm of vision tasks, adversarial perturbations represent a more intriguing and useful tool for understanding and improving the *safety* of AI systems than a concrete *security* threat.

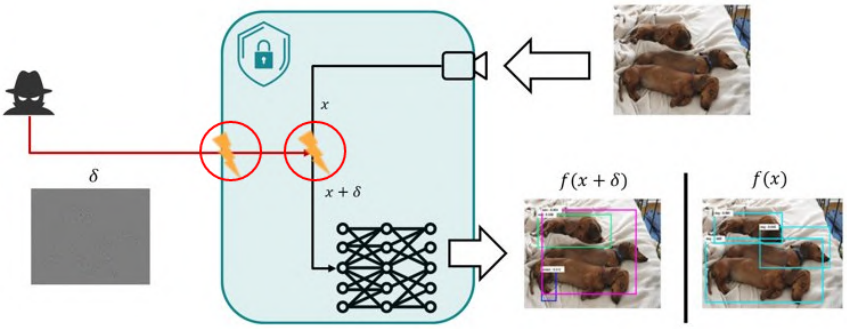
In response to these observations, research efforts have increasingly focused on *real-world adversarial attacks* [13]. These attacks leveraged *physical* objects, such as printable billboards and patches, designed to deceive DNNs from an external environment [24, 19]. This approach effectively circumvents software-based intrusion detection modules, as illustrated in Figure 1.2, thereby representing a more realistic security threat. From a safety perspective, these attacks also offer practical insights into model robustness, acting as a proxy for worst-case scenarios involving unexpected objects. This poses real-world attacks as also a valuable tool for evaluating the safety of DNNs in different threat scenarios.

Figure 1.2 illustrates the differences between real-world adversarial objects and adversarial perturbations from an attacker perspective. Such distinctions are particularly relevant in the context cyber-physical systems, such as autonomous vehicles or robots. In contrast, applications like social networks, which fall outside the scope of this thesis, present different challenges and attack scenarios, for instance, these platforms may facilitate the creation of black-box adversarial perturbations, leading to different discussions and considerations.

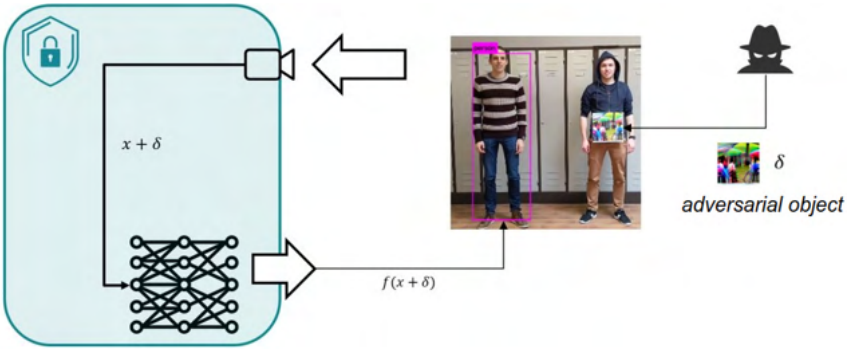
The Need for Testing, Evaluation, and Defenses in AI Systems

The increasing awareness of potential AI threats has highlighted the importance of developing comprehensive testing criteria and additional strategies to enhance the reliability and robustness of AI models. Traditional testing methods, which often rely on predefined datasets, generally fall in considering a wide range of unexpected inputs and threats.

Close to the topic addressed in this thesis, the aforementioned challenges have prompted the research community to explore more comprehensive testing strategies. These strategies are designed to assess model responses by exploring as many *internal behaviors* as possible. Taking inspiration from classic coverage testing in software engineering, coverage



(a) injecting an adversarial perturbation



(b) injecting a real-world advrsarial attacks

Figure 1.2: Differences between real-world adversarial objects and adversarial perturbations from an attacker perspective.

testing for DNNs focuses on systematically testing models by abstracting their behaviors into operational states (e.g., combinations of neurons activations). This approach provides a measure of the adequacy of a given test set as its level of coverage with respect to all possible states of the network. Although these techniques have sparked several concerns and discussions (as detailed in Chapters 2 and 3), they constitute an intriguing area of interdisciplinary research, which has received particular interest, especially from software engineering and cyber-physical systems communities.

Concerning real-world adversarial attacks instead, different considerations arise. The nature of real-world adversarial objects presents unique challenges compared to classic adversarial perturbations. Despite the already extensive literature [19], the study of real-world attacks remains in its early stages, with in-depth investigations across various scenarios and settings still unexplored. In this context, it is crucial to accurately define and evaluate new attack methods to understand their practical effectiveness in each real-world application and task.

Furthermore, with the increasing interest in real-world adversarial attacks, several defense strategies have been developed to enhance model robustness and reduce the attack effectiveness. However, in this field, much more attention needs to be directed towards design requirements that could lead to more suitable defenses for cyber-physical and real-

time system domains. As highlighted in Section 2.4.2, it is of utmost importance to develop defense mechanisms that not only provide high mitigation against real-world attacks but are also robust, reliable, and computationally efficient.

1.1 Thesis Statement

Scope of the thesis. This thesis aims at presenting the results and contributions from publications conducted during my Ph.D. studies. In addition, it poses strong effort for extending and enriching the discussion of the current literature, hoping to stimulate further research studies in all the addressed fields. The primary objectives of the thesis are: *(i)* To integrate and discuss the main works listed below in a cohesive and unified way, thus providing a broader perspective on how these studies fit together; *(ii)* To critically examine the existing literature in each field, including the contributions presented in this thesis; and *(iii)* To introduce novel ideas and perspectives that might establish initial points for future investigations.

My personal goal with this thesis is to enhance both my own understanding and that of the readers, encouraging an ongoing dialogue about these topics and setting the stage for future work. Indeed, each chapter concludes with a discussion on potential future research directions, inviting myself, colleagues, and readers to explore and contribute to these emerging areas of study.

Topics Addressed

The thesis faces and discusses various challenges in the field of testing, evaluation, and defense mechanisms for DNNs.

[Part A - Coverage Testing] The first part addresses the literature of coverage testing for DNNs and then proposes the use of coverage testing for the purpose of real-time DNN monitoring. Then, recalling the state-of-the-art landscape, we discuss future advancements and directions to be explored in DNN testing.

[Part B - Real-World Adversarial Attacks] The second part of the thesis focuses on real-world adversarial objects, specifically in outdoor scenarios related to autonomous driving. It focuses on presenting a novel attack method to generate real-world adversarial objects for semantic segmentation tasks. It also aims at extending the knowledge of the spatial robustness for computer vision models and addressing future research directions.

[Part C - Real-World Adversarial Defenses] Finally, to explore novel strategies for mitigating the effects of real-world adversarial attacks in computer vision tasks, we present the benefits of the over-activation analysis to better understand and counteract real-world objects. Various strategies are presented, supported by theoretical foundations, which allow implementing efficient and computationally affordable defense mechanisms.

1.2 Structure of the Thesis

The structure of the thesis is as follows:

- Chapter 2 introduces preliminaries on the formalisms and notations used. Then, it provides a unified discussion of the literature on testing, real-world attacks, and defense mechanisms;
- Chapter 3 introduces a new paradigm for enabling coverage testing for run-time monitoring. Then, it provides a detailed discussion that poses insights for future directions in DNN coverage testing;
- Chapter 4 presents formulations of the first real-world attacks for semantic segmentation, focusing on crafting adversarial patches to maximize the spatial adversarial effect in output predictions. An exhaustive set of experiments is performed under various settings and scenarios.
- Chapter 5 presents several defense mechanisms to mitigate real-world adversarial attacks, and assesses their robustness and efficiency through an over-activation analysis reformuled by a novel perspective.
- Chapter 6 - Concludes the thesis with a recap of the future directions and the contributions.

The contributions of this thesis are based on the following publications:

- [A] - **G.Rossolini**, A.Biondi and G.Buttazzo - Increasing the Confidence of Deep Neural Networks by Coverage Analysis - IEEE Transactions on Software Engineering (TSE), 2023
- [B,C] - **G.Rossolini**, F.Nesti, G.D'Amico, S.Nair, A.Biondi and G.Buttazzo - On the Real-World Adversarial Robustness of Real-Time Semantic Segmentation Models for Autonomous Driving - IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2023
- [B] - F.Nesti* **G.Rossolini***, G.D'Amico, A.Biondi and G.Buttazzo - Carla-gear: a dataset generator for a systematic evaluation of adversarial robustness of vision models - IEEE Transactions on Intelligent Transportation Systems (T-ITS), 2023
- [C] - **G.Rossolini**, F.Nesti, F.Brau, A.Biondi and G.Buttazzo - Defending from physically-realizable adversarial attacks through internal over-activation analysis - AAAI Conference on Artificial Intelligence 2023
- [C] - **G.Rossolini**, A.Biondi and G.Buttazzo -Attention-Based Real-Time Defenses for Physical Adversarial Attacks in Vision Applications - International Conference on Cyber-Physical Systems (ICCPS) 2024 - CPS-IoT Week 2024

For completeness, other works carried out during my Ph.D. that are not included in this thesis are:

- F. Brau, **G.Rossolini**, A.Biondi and G.Buttazzo - On the Minimal Adversarial Perturbation for Deep Neural Networks With Provable Estimation Error - IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) - 2022
- G.D'Amico, M.Marinoni, F.Nesti, **G.Rossolini**, G.Buttazzo, S.Sabina, G.Lauro - TrainSim: A Railway Simulation Framework for LiDAR and Camera Dataset Generation - IEEE Transactions on Intelligent Transportation Systems (T-ITS), 2023

- F.Nesti* **G.Rossolini***, S.Nair, A.Biondi and G.Buttazzo - Evaluating the Robustness of Semantic Segmentation for Autonomous Driving Against Real-World Adversarial Patch Attacks - Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) - 2022

- F. Brau, **G.Rossolini**, A.Biondi and G.Buttazzo - Robust-by-design classification via unitary-gradient neural networks - in AAAI Conference on Artificial Intelligence, 2023

Chapter 2

Background and Related Work

2.1 Preliminary Definitions

This section provides preliminary definitions and concepts adopted across the rest of the thesis. We consider neural networks for computer vision tasks that take as input an image with dimensions $H \times W$ pixels and C channels, denoted by $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$. Without loss of generality, the model output, denoted as $f(\mathbf{x})$, depends on the specific vision task under consideration. For examples,

- For a semantic segmentation model with N_y classes, the output $f(\mathbf{x}) \in [0,1]^{N_y \times H \times W}$ is a tensor that encodes the semantic context of each pixel (i,j), i.e., the probability distribution that each pixel as been classified with respect to each of the N_y classes. The predicted semantic segmentation (SS) of the pixel (i,j), denoted by $\hat{y}_{(i,j)}$, is then computed by extracting the class with the highest probability score, i.e., $\hat{y}_{(i,j)} = \text{argmax}_{c \in \{1,\dots,N_y\}} f_{(i,j)}(\mathbf{x})$. The complete semantic segmentation prediction $SS(\mathbf{x})$ is then the collection $\{\hat{y}_{(i,j)},\ \forall (i,j)\}$;

- For an image classification model, the output $f(\mathbf{x})$ is a vector in $\mathbb{R}^{N_y}$. The predicted class, as for semantic segmentation, is obtained by selecting the class with the highest probability score, i.e., $\hat{y} = \text{argmax}_{c \in \{1,\dots,N_y\}} f(\mathbf{x})$;

- For an object detection (OD) model, the output $f(\mathbf{x})$ is typically a set of tuples, each representing an object detected in the image. Each tuple generally consists of a class label, a probability score, and a bounding box;

Operative Notation. For simplicity, the notation is introduced by referring to a simple feed-forward deep neural network, with a list of layers $\{L_1, \dots, L_{N_L}\}$, where the input is forwarded sequentially through these layers. To this end, we use the following notation $f^{i \to j}$ to denote the processing flow of features from layer L_i to layer L_j. The layer index 0 is used to refer to the input ($\mathbf{x}$). For instance, $f(\mathbf{x})$ is equivalent to $f^{0 \to N_L}(\mathbf{x})$ or $f^{j \to N_L}(f^{0 \to j}(\mathbf{x}))$ for any j such that $1 \leq j \leq N_L$.

Features notation. We denote by $\boldsymbol{h}_l \in \mathbb{R}^{C^l \times H^l \times W^l}$ the features produced by any layer L_l, where C^l, H^l, and W^l are the corresponding tensor dimensions, i.e., $\boldsymbol{h}_l = f^{0 \to l}(\mathbf{x})$.

The notation $(*)_{(c,i,j)}$ is used to denote a single element of any 3D tensor $*$, where c, i, and j are the indices for the channel, height, and width dimensions, respectively. If one or more indices are omitted between (c, i, j), it refers to all the elements across the not specified dimensions. For example, $\mathbf{h}_{l,(i,j)}$ is a vector in $\mathbb{R}^{C^l}$ that selects all the C^l neurons of layer L_l at the spatial position (i, j), while $h_{l,(c,i,j)}$ is a specific activation in layer L_l, at the channel c and spatial position (i, j).

Time notation. Part of the research conducted in this thesis leverages iterative optimization strategies and algorithms in multi-frame scenarios (e.g., Section 5.5), thus we introduce a discrete-time notation with the index t to refer to symbols when related to the t-th frame, where $t \in \{0, \dots, T\}$.

Note about the use of specific symbols. Due to the diversity of the algorithms presented in the thesis, other symbols, as λ, γ, β, etc., will redefined intercenchably in different sections to indicate particular propretiers and parameters of each algorithm under consideration.

Training and Testing of Models

Without loss of generality, a task-specific loss function $\mathcal{L}(f(\mathbf{x}), \mathbf{y})$ is used to measure the quality of a prediction $f(\mathbf{x})$ with respect to a ground-truth output $\mathbf{y}$. The training process aims at minimizing the average loss over the training dataset $\mathcal{X}$, which can be formalized as:

$$\min_{\theta} \frac{1}{|\mathcal{X}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}} \mathcal{L}(f(\mathbf{x}), \mathbf{y})$$

where θ represents the parameters of the model, $|\mathcal{X}|$ denotes the size of the training dataset, and $f(\mathbf{x})$ depends from its trainable parameters θ.

Various loss functions are adopted in computer vision tasks. For instance, the cross-entropy loss is commonly used for image classification, while pixel-wise cross-entropy is utilized for semantic segmentation. Often, a combination of multiple loss functions is used [25].

Concerning the testing, a test set $\mathcal{T}$ is defined as a set of samples, which differ from the training data, used to evaluate the model performance and other important metrics. This set allows assessing how the model responds to data that has not encountered during training, thereby providing a measure of the generalization capabilities, robustness, and other properties. Some classic performance metrics include mean Average Precision (mAP) for object detection; mean Intersection over Union (mIoU) or pixel accuracy for semantic segmentation; for image classification. Beyond these standard metrics, this thesis will explore in Section 2.2 and Section 3 a more detailed understanding of testing metrics (criteria) for DNNs.

Perturbed Samples

A sample $\mathbf{x}$ can be perturbed to get $\tilde{\mathbf{x}}$. The type of perturbation can vary, for instance, it can be achieved via data augmentation (e.g., rotation, scaling, masking, etc.), through adversarial ϵ-bounded perturbations, or by adding real-world adversarial objects.

Concerning adversarial perturbations, we define $\tilde{\mathbf{x}} = \mathbf{x} + \delta$, where the perturbation δ is usually crafted to be adversarial under a magnitude ϵ, such that $\|\mathbf{x} - \tilde{\mathbf{x}}\|_p \leq \epsilon$ for a specific norm p.

In the case of real-world adversarial objects (better discussed in Section 2.3), for a 2D image, an object can be generalized as an adversarial patch [26], as commonly addressed in the literature. Formally, an adversarial patch can be expressed as an unconstrained perturbation $\delta \in \mathbb{R}^{C \times \tilde{H} \times \tilde{W}}$, where $\tilde{H} \leq H$ and $\tilde{W} \leq W$. The patch is applied to the image $\mathbf{x}$ to cover a specific area, which is indicated by a binary mask $\mathbf{M} \in \{0,1\}^{H \times W}$, such that the patch application is formulated as follows:

$$\tilde{\mathbf{x}} = \mathbf{M} \odot \mathbf{x} + (1 - \mathbf{M}) \odot \delta$$

where δ is in practice mapped into the image dimension through the use of zero padding, and $\odot$ denotes the Hadamard product operator on the spatial dimensions. Essentially, this formula replaces the pixels of $\mathbf{x}$ with those of δ in the region indicated by $\mathbf{M}$.

For simplicity, in sections where the use of real-world attacks or adversarial patch are discussed, the application of δ is sometimes indicated by $\tilde{\mathbf{x}} = \mathbf{x} + \delta$ or accomplished by specific *application function*, with the exact position given by $\mathbf{M}$.

2.2 DNN Testing and Coverage Criteria

DNN Testing is a crucial step in the MLOps pipeline, aimed at validating the performance, generalizability, robustness, and safety of trained models. Unlike traditional software algorithms, DNNs present unique challenges for testing due to their complex architectures, data-driven nature, and low interpretable decision-making processes.

2.2.1 Coverage Testing for DNNs

Coverage testing in DNNs [27, 28, 29, 30, 31, 32, 33], a concept adapted from the software testing field, focuses on measuring how much different 'behaviors' (e.g., layers, neurons, etc.) of the neural network have been stimulated (covered) during testing.

In particular, coverage testing refers to *test objectives*, as distinct elements of the model, that require to be tested by one or more test samples. Under these considerations, the coverage achieved by a test set $\mathcal{T}$ (i.e., *test coverage*) can be quantified as follows:

$$\textit{Test coverage} = \frac{\textit{Number of test objectives covered by } \mathcal{T}}{\textit{Total number of test objectives}} \tag{2.1}$$

where the 'number of covered test objectives' refers to the sum of unique test objectives activated (or effectively tested) by one or more test samples.

Formally, in accordance with the notation used in [34], given a DNN characterized by a function f, a coverage criterion cov, which provides a definition of a set of test objectives R, and a test suite $\mathcal{T}$, the covered test objectives can be expressed as:

$$\{\alpha \in R \,|\, \exists (x_1, x_2, \ldots, x_k) \in T : \text{cov}(\alpha, (x_1, x_2, \ldots, x_k)) = \textit{True}\}$$

where cov( ) is an auxiliary function that returns 'True' if the test objective α has been stimulated by one or more test samples in $(x_1, x_2, \ldots, x_k)$. This implies that within $\mathcal{T}$, there exists a subset of samples that cover a test objective α from the set R.

Lastly, the 'total number of test objectives' refers to the complete set of test objectives within the network architecture that are pertinent to the chosen coverage criterion, denoted as $|R|$. Therefore, we can write the previous formula (2.1) as:

$$M(R, \mathcal{T}) = \frac{|\{\alpha \in R \,|\, \exists (x_1, x_2, \ldots, x_k) \in T : \text{cov}(\alpha, (x_1, x_2, \ldots, x_k)) = \textit{True}\}|}{|R|}. \tag{2.2}$$

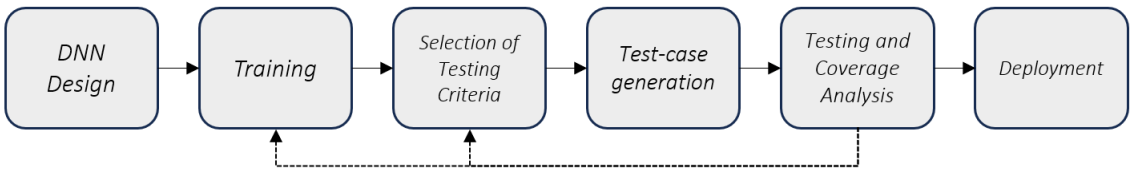


Figure 2.1: MLOps pipeline integrating coverage testing.

Similar to the steps conducted in software engineering, the pipeline for utilizing coverage criteria is illustrated in Figure 2.1 and is described in more detail in the following items:

1. *Design and Implementation of DNN Models*: Initially, a DNN model is carefully developed or selected to meet specific tasks and requirements. This stage involves choosing the appropriate architecture, training methods, and datasets;

2. *Training*: The DNN model is trained with a dataset $\mathcal{X}$, based on a specific training strategy in order to reach satisfactory performance;

3. *Selection of Coverage Criteria*: After the training phase, one or more coverage criteria for DNNs are selected based on the specific testing requirements;

4. *Test Case Generation*: Test cases are generated according to the selected coverage criteria. These are designed to maximize coverage of the network under test, potentially using optimization methods, to challenge the network and reveal possible unsafe inputs. Formally, test case generation can be defined as an optimization problem aimed at producing a test suite $\mathcal{T}$ represented as:

$$\mathcal{T} = \begin{cases} \max_{\mathcal{T}} & M_{\text{cov}}(R, \mathcal{T}) \\ & C_1(\mathcal{T}),\ C_2(\mathcal{T}),\ \ldots \end{cases}$$

   where: $M_{\text{cov}}(R, \mathcal{T})$ is the test coverage based on the criterion cov, quantifying the coverage of test objectives in R by the test suite $\mathcal{T}$. While $C_i(\mathcal{T})$ represents the i-th constraint on the test suite $\mathcal{T}$. A constraint could include a bounded perturbation magnitude (e.g., ϵ-normed adversarial perturbations), specific augmentation techniques, and so on.

   Common techniques in literature for test case generation include: adversarial attacks [35], fuzz testing [31, 36, 30, 37, 32], symbolic execution and testing [38, 39], testing via generative adversarial networks [40] and mutation testing [41, 42, 43]. Since the test-case generation process is not particularly related with the scope of this thesis, a comprehensive review can be found in [34];

5. *Testing and Coverage Analysis*: The generated test cases are forwarded on the DNN, and both the network output and the test objectives are observed and examined. After executing the test cases, the coverage is analyzed to determine how much of the DNN has been covered by the test set, i.e., $M_{\text{cov}}(R, \mathcal{T})$. This analysis helps in understanding the degree of which the DNN has been evaluated and in identifying any parts of the network that may require more thorough testing. In fact, if the testing and coverage analysis reveals that certain areas of the network are undertested or exhibit unexpected behaviors, additional test cases can be generated to target these areas or a new training phase should be run to improve the model robustness and performance;

6. *Final Evaluation and Deployment*: Once satisfactory coverage is reached, and the DNN behaves as expected across the test cases, it can be considered ready for deployment. However, as it is known, continuous monitoring and testing might be necessary as the DNN is exposed to real-world data and scenarios.

2.2.2 Coverage Criteria

The following paragraphs explore specific coverage criteria for DNNs (properly listed in [34]) and introduce critical viewpoints from the literature, questioning the practical use and effectiveness of these criteria in DNN domains. This discussion sets the stage for Chapter 3, where a novel use of coverage criteria for run-time monitoring is introduced, and future research directions regarding coverage testing are more thoroughly presented and schematized.

Classic Structural Coverage Criteria

Structural coverage criteria establish test objectives based on the internal components of a given DNN, assessing the extent to which these components (such as neurons, layers, etc.) have been activated during testing. Important works about structural coverage critiera, better discussed in the following are: neuron coverage [33], layer coverage [44], and path coverage [28].

Neuron Coverage (NC) [33] One of the first structural coverage criteria is *neuron coverage*. This criterion evaluates whether a specific neuron n of the network f serves as a critical test objective. The underlying assumption is that the activation of a single neuron represents a specific behavior of the model. This is achieved by determining whether the neuron is activated or not, based on a predefined threshold. The aim is to ensure that a substantial portion of the neurons in the network are stimulated during the testing phase, which could potentially reveal hidden vulnerabilities in the DNN.

Formally speaking, for a neuron n in a DNN model f, it is considered as an activated if, given an input $\mathbf{x}$, the output of the neuron n, denoted with $\phi(\mathbf{x}, n)$, is larger than a threshold τ. Given a set of inputs $\mathcal{T}$, the NC is defined as the ratio of the number of unique activated neurons for all test inputs and the total number N of neurons in a DNN model.

$$NC(T, t) = \frac{|\{n, \mid \exists \mathbf{x} \in \mathcal{T}, \phi(\mathbf{x}, n) > \tau\}|}{|N|}$$

Inspired by neuron coverage, the following are more sophisticated coverage criteria.

k-multisection Neuron Coverage (KMNC) [32] For a neuron n, the upper and lower boundary of its input values on training data can be denoted as up_n and low_n respectively. The boundary $[low_n, up_n]$ is divided into k equal sections, where S_{mn} denotes the m-th section of the neuron n. Then $\phi(\mathbf{x}, n) \in S_{mn}$ means the m-th section of n is covered by at the input $\mathbf{x}$. Hence, the *KMNC* test criterion is defined as

$$KMNC(\mathcal{T}) = \frac{\sum_{n \in N} |\{S_{mn} \mid \exists \mathbf{x} \in \mathcal{T} : \phi(\mathbf{x}, n) \in S_{mn}\}|}{k \times |N|}$$

Neuron Boundary Coverage (NBC) [32] For test inputs $\mathcal{T}$, the output values may fall into $(-\infty, low_n)$ or $(up_n, +\infty)$ referred as corner-case regions instead of the (low_n, up_n)

boundary derived from the training data. The NBC measures to what extent the corner-case regions are covered outside the boundary derived from training data.

$$NBC(T) = \frac{|UCN(\mathcal{T})| + |LCN(\mathcal{T})|}{2 \times |N|}$$

where $UCN(\mathcal{T})$ and $LCN(\mathcal{T})$ denote the sets of unique neurons with output values in the upper and lower corner-case regions, respectively.

Strong Neuron Activation Coverage (SNAC) [32] This criterion semplifies NBC by only measuring how many upper-corner neurons have been covered by the given test inputs $\mathcal{T}$.

$$SNAC(\mathcal{T}) = \frac{|UCN(\mathcal{T})|}{|N|}$$

This metrics have then been extended to also calculate coverage from the layer perspective:

Top-k Neuron Coverage (TKNC) [32] Given a test suite $\mathcal{T}$ and a layer L_l of a DNN, TKNC is defined as the ratio of the neurons that have been between the most active k neurons of each layer on the given test data $\mathcal{T}$.

$$TKNC(\mathcal{T}) = \frac{|\bigcup_{x \in \mathcal{T}} \bigcup_{1 \leq l \leq L} topk(\mathbf{x}, l)|}{|N|}$$

Top-k Neuron Patterns (TKNP) [32] For a test input $\mathbf{x}$, the top-k neurons of each layer form a specific pattern. TKNP measures how many patterns of the top-k neurons are covered by the given test data $\mathcal{T}$.

$$TKNP(T) = |\{(topk(x, 1), \ldots, topk(x, L)) \mid x \in T\}|$$

Despite several concerns raised in recent years about the use of the aforementioned structural coverage criteria in DNN testing (points remarked also in the next paragraphs and Chapter 3), these criteria are still under investigation and utilized for test case generation strategies [34].

Figure 2.2, from [45], provides an illustrative example of the structural coverage criteria mentioned above, to demonstrate their differences, in terms neurons activation within test objectives.

More Recent Coverage Criteria.

For completeness, it is important to mention other coverage techniques that, as discussed in Section 3.2, could pave the ways for future research in this field.

Modified Condition Decision Coverage. One of the first works to explore the inter-layer relationships as test objectives in the domain of DNN structural coverage testing is [28]. This study introduces the concept of path coverage by examining connections between neurons in adiacent layers, drawing from the Modified Condition Decision Coverage (MC/DC) concept traditionally used in software testing [46]. Specifically, in software

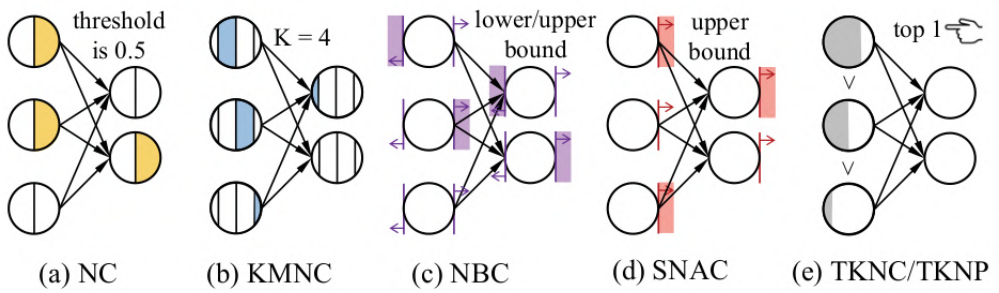


Figure 2.2: Illustration of the neurons parts addressed by above mentioned structural coverage criteria. Figure from [45].

engineering, MC/DC (Modified Condition Decision Coverage) is designed to verify that each condition in an earlier part of an algorithm has a distinct influence on the decision outcomes in subsequent instructions. Adapting this concept to DNNs, Sun et al. [28] proposed a methodology aimed at determining if the activation of a neuron in layer L_l is a primary cause for the activation of another neuron in the subsequent layer $L_l + 1$.

While this approach expands the range of coverable states beyond those addressed by previous structural metrics, it encounters possible limitations in terms of scalability. The large number of interconnections between neurons in consecutive layers poses a significant challenge to the practical application of this method.

Despite this limitation, the adaptation of MC/DC to the realm of DNN testing represents an interesting and intriguing advancement for the development of more sophisticated criteria.

Neuron Path Coverage. Xie et al. [47] proposed one of the first criteria linked with the explainable AI field. The authors defined the concept of a Critical Decision Path (CDP) from a set of critical neurons identified through the Layer-Wise Relevance Propagation (LRP) algorithm [48]. These specific neurons act as the main components for the test objectives used in the testing strategy. The strong integation of DNN testing with the use of XAI methods poses this work as a pivotal idea for future research in the field.

Surprise Coverage. Another interesting study is the concept of surprise coverage, proposed by Jinhan et al. [35]. The main idea behind surprise coverage crtieria is on measuring the "surprise" (novelty or difference) of test inputs with respect to the distribution of features in the training data. To achieve this, the surprise criteria employ kernel density functions, which are utilized to compute the distance of features from a given test input from the known training distribution. The goal is to develop test-generation algorithms capable of exploring novel areas in the DNN's feature space.

Other recent works. Recent advancements have introduced novel concepts of coverage criteria that go beyond traditional structural coverage metrics. In [45, 44], the authors posed interesting observations for a correct design o coverage criteria for DNNs. These approaches will be thoroughly discussed in Chapter 3.2 to address the current limitations identified in existing literature while also exploring potential directions for future research.

2.2.3 On the Properness of Structural Coverage Criteria

Despite the interest in classic structural coverage criteria across from related literature, recent studies have raised questions about their adequacy.

In [49], the authors conducted a comprehensive comparison of structural coverage metrics, focusing on functional diversity and defect detection capabilities. Specifically: (i) *functional diversity* focuses on analyzing the testing benefits of test samples coming from a diverse range of output classes. A test set that is sensitive to functional diversity should report increased coverage when inputs from multiple classes are included. Ideally, the coverage is expected to grow linearly as the variety of classes in the test set expands. (ii) *Defect detection ability* evaluates the effectiveness of a coverage metric in identifying malicious inputs within neural network models.

The study highlighted a general lack of sensitivity to both functional diversity and defect detection, indicating a need for new coverage metrics.

Another interesting study that question the use of structural coverage criteria is [50], which converged to similar results. The authors in [50] confirmed that increased neuron coverage does not strongly correlate with enhanced defect detection capabilities. In fact, only 2 out of 64 experimental outcomes supported this hypothesis, while 33 results indicated a negative correlation. This suggests that higher neuron coverage could potentially impair defect detection, when it is used to enhance datasets for retraining, as in adversarial training scenarios. Furthermore, the study noted a poor naturalness of test-generated inputs: increasing neuron coverage tends to produce more unnatural inputs. Finally, concerning testing bias, the study revealed that certain class labels inherently have higher neuron coverage, showing that some criteria could uncorrectly balanced with respect to the set of output classes addressed.

The work presented in Chapter 3 also explores structural coverage criteria, adapting them for a distinct application, namely run-time monitoring. However, beyond this different use case, the results obtained are in align with the observations made in the aforementioned works ([49] and [50]).

The results achieved in the first part of Chapter 3, combined with the concerns raised by these studies, have inspired a detailed discussion in Section 3.2, posing potential future directions in the field of coverage testing.

2.3 Real-World Adversarial Attacks

The existence of adversarial examples have proved the poor robustness of deep learning models, showing how imperceptible perturbations can easily deceive their outcomes [20, 51, 52, 12, 53, 16, 54, 55].

However, from the perspective of secure AI, in cyber-physical domains, like autonomous cars and robots, special attention has been directed towards real-world attacks. As discussed in the introduction, real-world attacks pose a more intriguing and concrete security threat due to their ability to inject adversarial effects through physically realizable objects.

2.3.1 Crafting Real-World Adversarial Attacks

Inspired by the categorization adopted by Wang et al. [19], we can define the taxonomy of real-world adversarial attacks by addressing the attack settings and the attack implementations.

Attack Settings. With the attack settings, the attacker specifies:

- *Reference vision task*, for instance, image classification or semantic segmentation;
- *Attacker knowledge*, i.e., white-box or black-box attacks. Note that due to the difficulties on crafting real-world attacks in black-box settings (especially for dense prediction tasks), we will focus our analysi on white-box attacks;
- *Adversarial objective*, i.e., targeted or untargeted attacks. This indicates whether the intention is to guide the attack towards a specific targeted output or just reduce the performance of the model.

In addition to these considerations, specifying the *reference testing scenarios* is extremely important. For instance, some attacks have been evaluated solely as digital patches, i.e., the images are modifed at pixel-level by attaching the patch on top [26, 56, 57]. In contrast, more realistic scenarios involve creating printable objects that are then placed and tested within the external environment [24, 58, 59, 60].

While the latter represents a more correct testing strategy, the challenges in evaluating attacks in real-world scenarios pose practical obstacles. As a result, many studies focus primarily on digital evaluations. Nevertheless, real-world evaluations are crucial, particularly when facing target outdoor scenarios that might be influenced by varying shadows, light conditions, and other natural transformations. As such, a more realistic evaluation provides a deeper understanding of the robustness constraints required during the optimization of the attack.

Attack implementation. Once the attack objective is defined, a crafting strategy should consider:

- *Optimization Strategy:* many approaches focus on iterative gradient-based attacks, while other works explore generative model-based approaches [61, 62]. In the following, we focus on the iterative gradient-based approach, aligning with the contributions proposed in this thesis.

- *Robustness Constraints:* These are constraints that must be integrated into the optimization strategy to account for possible transformations that can occur in the real world. Further details on this point will be discussed later.

- *Application Surface:* This represents the object or surface used to apply adversarial features. Many works refer to patch-based attacks where the adversarial attack is a 2D surface depicted as a billboard or a digital patch [26, 24, 26]. In contrast, other works (e.g., [63, 64, 65]) tackle more complex yet realistic use cases in a physical environment by utilizing 3D surfaces. In such cases, the attack is depicted as a texture attached to 3D objects. Some works additionally aim at crafting also the shape of the surface, introducing an additional degree of freedom for performing the attack [66]. Finally, certain works do not focus on texture-level attacks but instead address optical attacks, by the means of laser [67] or flashlights [68].

Figure 2.3, from [19], presents a list of works that characterized the ongoing research on real-world adversarial attacks.

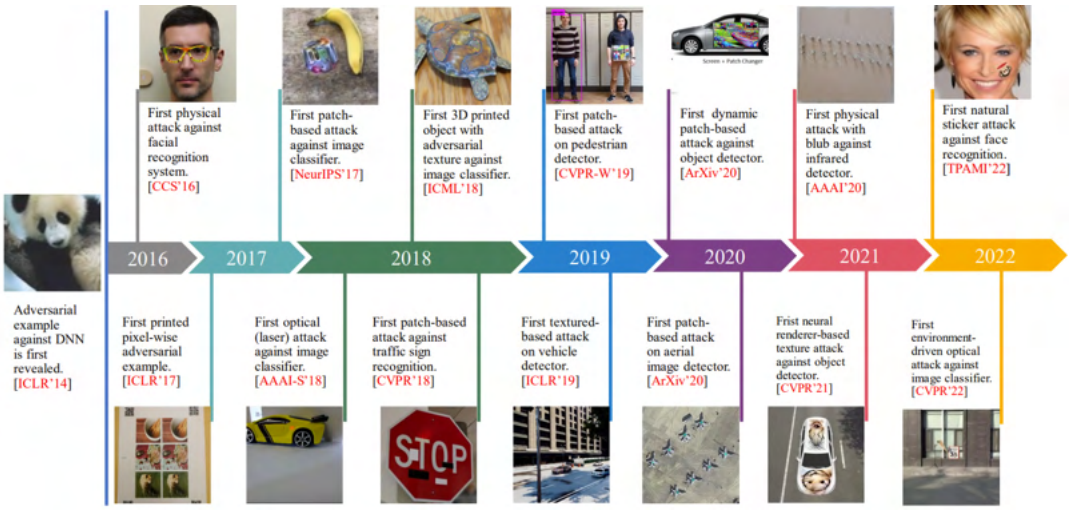


Figure 2.3: Illustration of different types of freal-world attacks. Image from [19]

Adversarial Optimization Problem. A generalized formulation of a real-world attack attack's objective can be written through the following optmization problem:

$$\delta = \underset{\delta}{\operatorname{argmin}} \ \mathbb{E}_{\mathbf{x}\in\mathbf{X},\zeta_a\in\mathbf{\Gamma_a},\eta} \ \mathcal{L}_{Att}(f(\tilde{\mathbf{x}}), y_{adv}) \tag{2.3}$$

where the pertubed image $\tilde{\mathbf{x}}$ is obtained through a *patch application function*

$$\tilde{\mathbf{x}} = g(\mathbf{x}, \delta, \Gamma_a, \eta),$$

which allows to control possible transformation of the patch and its position within the image. In particular, the patch application function replaces the area of the image $\mathbf{x}$ specified by a *patch placement function* η with an modified version of the patch δ obtained

by a transformation randomly selected from Γ_a, hence returning the patched image $\tilde{\mathbf{x}}$. More in details:

- A *set of appearance-changing transformations* Γ_a: Each element of Γ_a is a composition of different transformations, as illumination changes (brightness and contrast) and noise addition (uniform or Gaussian). This set of transformations is randomly sampled, and the selected transformation is directly applied to the patch δ. The typical parameters of these transformations are randomized during the optimization to make the adversarial features more robust across real-world environments.
- A *patch placement function* η: This function defines which portion of the original image $\mathbf{x}$ is occupied by the adversarial features. The function η can have different definitions depending on the chosen attack. For instance, it can address a simple random 2D position or it can account for projective placement based on 3D transformations.

Finally, the attack *loss function* $\mathcal{L}_{Att}$ depicts the objective function to be optimized. $\mathcal{L}_{Att}$ consists of a weighted sum of multiple loss functions, including the *adversarial attack loss* $\mathcal{L}_{Adv}$. To further ensure that the patch transfers well to the real world, additional losses are considered for *physical realizability* of the patch. For examples, a *a smoothness loss* $\mathcal{L}_S$ and a *non-printability score* $\mathcal{L}_N$. These additional losses are both discussed more in details in Section 2.3.1.

Note that the optimization problem in (2.3), also known as Expectation over Transformation [13], has been formulated here in a completely generalized manner to fit with different attack settings (e.g., targeted or untargeted). In the following subsections, we better address all the differences across various settings and implementations.

Adversarial objective

As mentioned previously, we can consider two different attack objectives: *untargeted attacks* and *targeted attacks*. Untargeted attacks aim at foolin a target model into outputting predictions that deviate from the correct ground truth label (y). Conversely, targeted attacks aim at returning a specific outcome chosen by the attacker.

Technically speaking, the attack objective is usually specified in the adversarial loss function $\mathcal{L}_{Adv}$. For instance, in the case of untargeted attacks, in equation (2.3) y_{Adv} should resemble the ground truth label ($y_{Adv} = y$), where the adversarial loss function included in $\mathcal{L}_{Att}$ should be defined such that the optimization process causes the output to deviate from having y_{Adv} (e.g., negative cross-entropy loss for image classification). Conversely, in the case of targeted attacks, y_{Adv} could represent a desired class ($y_{Adv} \neq y$) and the adversarial loss should push the prediction towards y_{Adv} (e.g., cross-entropy loss).

It is important and also interesting to note that targeted attacks encounter more obstacles. This is especially true when the relationship between the original and target classes is semantically hard to get within the high-dimensional decision space [69]. This implies that targeted attacks are often more complex to perform than untargeted ones [70, 52]. This assertion is also supported by our findings in Chapter 4, which explores semantic segmentation tasks in driving scenarios. For instance, crafting adversarial patches to misclassify 'road' as 'tree' proved to be challenging, whereas misleading the model to misclassify 'road' with 'sidewalk' was notably easier.

Transformations and Robustness Constraints.

Equation (2.3) aims at generalazing the attack to be robust against different transformations, which represents a crucial aspect in the optimization of real-world adversarial attacks. To this end, the objective is to optimize the attacks across a broad set of transformations, addressing changes in both appearance and position (in the 2D or 3D world).

However, the choice of transformations to consider is problem-specific and should be made with careful consideration. While a larger set of transformations can enhance generalizability, it may also likely reduce the power of adversarial effectiveness. This is due to the introduction of more constraints in the optimization problem.

Common approaches involve classic transformations such as rotation, scaling, translation, perspective distortion, reflection, brightness, blur, and so on. The specific set of transformations, and also how they are weighted during the optimization, should be decided accounting the tasks and the attack scenarios. For instance, in the case of small indoor scenario some transformations could more important than outdoor driving environments.

Projective 3D Transformations. In some works, e.g., [71, 66], and the work presented in Section 4.3, 3D transformations are considered to improve the adversarial transferability of the object in the 3D world. However, to allow the use of 3D projections, additional information should be provided, such as the 3D rotation-translation matrix. Specifically, using 3D rotation-translation in homogeneous formulation [72], it is possible to express the position of a 3D point on the attackable surface p^S (initially expressed in the surface reference frame) from the perspective of the camera reference frame (e.g., the camera on top of the autonomous veichle). Given T_a^b, as the rotation-translation matrix from the generic reference frame a to b, which is expressed as a composition of a rotation matrix R_a^b and translation vector t_a^b,

$$T_a^b = \begin{bmatrix} R_a^b & t_a^b \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{2.4}$$

the expression for a point on the attackable surface in the camera reference frame is

$$p^C = T_W^C T_S^W p^S \tag{2.5}$$

where T_W^C is the World-to-Camera rototranslation, and T_S^W is the Surface-to-World rototranslation.

The resulting point can be projected with a pinhole camera model, which is the one used by RGB camera sensor. The projection, which is expressed in pixels, and identify the position of the point on the image, can be obtained as

$$p_{pixels} = K p^C \tag{2.6}$$

where K is the intrinsic matrix of the RGB camera sensor. Its general form is

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

where f is the focal, and (c_x, c_y) is the center of the image expressed in pixels (and still in homogeneous coordinate).

Retrieving these parameters can be difficult, to this end important tools are required. For instance, as described in [73], to craft adversarial t-shirts, the authors used a checkerboard pattern to map pixels from one frame to another, providing parametric information about the t-shirt deformation and movement.

In the context of driving scenarios, as those discussed in [24], many studies leverage virtual environments, like Carla [74]. These simulations automatically provide 3D matrices, which can then be integrated into the optimization processes with 3D projection operations (see approach proposed in Chapter 4).

Non-printability score and smoothness loss. The transformations mentioned previously are typically applied as a preprocessing step. However, there are also other important constraints, which need to be integrated into the loss function of the adversarial attack.

In particular, physical realizability is a crucial factor to consider in the optimization process of digital adversarial features. This is because the printer used to print the adversarial textures likely cannot replicate the entire continuous spectrum of colors (i), and may not match the resolution of the digital representation of the features (ii). Additionally, the level of detail and noise in images collected in specific environments can significantly differ from those considered during the digital optimization (iii).

To addressed the first issue, the non-printability score [75] is introduced to take into account this effect. Assuming that a pixel p of the patch δ is composed of an RGB triplet, the *non-printability score* $\mathcal{L}_N$ is defined as

$$\mathcal{L}_N(p) = \prod_{c \in Colors} \|p - c\| \tag{2.8}$$

where *Colors* is the set of RGB triplets that compose the printable color palette of the printer. The non-printability score is then averaged across all the pixels p of the patch δ.

Concerning the second and third issues, typical adversarial perturbations are generally very noisy, since there is very little correlation between adjacent pixels. These patterns in the real world are smoothed out by both the printing process, and the camera acquisition process (which introduces noise and blurring). This effect can be taken into account by considering a *smoothness loss*, defined as

$$\mathcal{L}_S = \sum_i \sum_j [(\delta_{i+1,j} - \delta_{i,j})^2 + (\delta_{i,j+1} - \delta_{i,j})^2] \tag{2.9}$$

where (i, j) are the 2D index of the patch. This additional loss function introduces correlation between adjacent pixels, and should help craft a more "smooth" perturbation, without noisy patterns.

Overall adversarial attack loss. Considering only the previously mentioned losses[1], we can compute the final adversarial attack loss for physically-realizable real-world adversarial attacks as follows:

$$\mathcal{L}_{Att} = w_{adv}\mathcal{L}_{adv} + w_N\mathcal{L}_N + w_S\mathcal{L}_S \tag{2.10}$$

[1] additional losses may be required depending on the specific problem addressed

where w_{adv}, w_N, w_S are the weights corresponding to each loss component.

Implementation of formula.

As previously mentioned, many studies address an iterative gradient-based method for solving (2.3). The process at step t can be defined as follows:

$$\delta_{t+1} = \text{clip}_{[0,1]} \left(\delta_t + \epsilon \cdot \sum_{x \in \mathbf{X}} \nabla_{\delta_{k,t}} \mathcal{L}_{Att}(f(\tilde{\mathbf{x}}), y_{\text{Adv}}) \right), \tag{2.11}$$

where $\tilde{\mathbf{x}} = g(\mathbf{x}, \delta, \Gamma_a, \eta)$, and ϵ represents the step size.

2.3.2 Related work

A pioneering study on physical adversarial attacks is Kurakin et al. [54], which proposed a strategy for crafting real-world adversarial pictures against image classifiers. Building upon this foundation, Athalye et al. [13] introduced the Expectation Over Transformation (EOT) algorithm, detailed above. This approach enabled the creation of objects that maintain their adversarial properties when collected from different points of view.

Subsequently, [26, 76] introduced adversarial patches, as localized, image-agnostic object capable of fooling neural networks when placed in the input scene. Adversarial patches are still largelly explored and used in the literature, as a versatile tool for understanding the real-world robustness of DNNs.

Later, a large set works have addressed patch-based physical attacks for vision tasks, such as image classification and facial recognition [26, 75, 77], single-object detection [76], and steering angle prediction [61]. Other works have extended the analysis of real-world attacks to dense prediction tasks like, multi-object detection [78, 79, 80, 63, 81, 82, 83], optical flow [84], LiDAR object detection [85], and monocular depth estimation [86]. Additionally, interesting works have explored the use of adversarial attacks as textures on wearable items, like adversarial T-shirts [73, 78].

Finally, it is also noteworthy that the large research in the community on real-world adversarial attacks has inspired the realization of several surveys, including [34, 19, 87, 88, 89, 90, 91].

The studies conducted in this thesis primarily focus on dense prediction tasks, with a particular emphasis on semantic segmentation models, an area not extensively explored before, concerning real-world adversarial attacks. Addressing dense prediction tasks in vision, such as semantic segmentation or object detection, as opposed to single output tasks like image classification, presents intriguing and novel challenges. Informally speaking, these challenges arise because deceiving areas outside the adversarial patch becomes significantly more complex due to the intrinsic architectural nature of convolutional models.

Attacks in Driving Scenarios

Many works across the literature of attacks on autonomous driving tasks [63, 64, 92, 71, 93, 94, 93] rely on single and multi-objects detection, and steering angle prediction.

Evaluating and attacking the robustness in the context of autonomous driving presents inherent challenges, due to the need of getting precise control over complex outdoor driving

environments. To address this, works like [63, 79] have turned to the use of autonomous driving simulators, with Carla [74] being a particularly valuable solution. Different from previous works that primarily focused on crafting 2D objects like patches [26], some studies have pushed the attention towards attacks that encompass the 3D space. For instance,n adversarial camouflage-based attacks that involve altering the appearance, shape, or texture of 3D objects, studies like [71, 66] have introduced interesting techniques as differentiable rendering [93] and adversarial optimization strategies [71, 66].

In our investigation, detailed in Chapter 4, we focused on optimizing 2D objects, such as adversarial patches and billboards, while paying special attention to 3D transformations during the optimization steps. This choice simplifies the optimization problem by focusing on 2D objects instead of 3D objects, thereby enabling more effective attacks and well transferability due to the use of 3D transformations.

On the Spatial Robustness

In addition to the discussed attack scenarios and tasks, important observations, which have inspired our studies conducted in Chapter 4, arise from [56] and [95]. These works provide fundamental insights concerning the robustness of convolutional neural networks models on dense prediction tasks.

Specifically, [95] was one of the first works to explore the concept of spatial robustness by empirically evaluating how much a patch can impact predictions beyond its local region. Recognizing the poor robustness of various models, the authors introduced an adversarial training mechanism designed to reduce the perceptive fields of the model. This approach encourages the prediction of each input pixel to depend more on its local area. While their method led to notable improvements in spatial robustness, we noticed that the capability of an attack to influence regions outside the patch is more related to architectural aspects of the model, rather than to the training of the model's weights.

The second work, [56], was a pioneering study addressing the robustness of semantic segmentation models against localized and image-specific perturbations (not necessarily adversarial patches or real-world attacks). They crafted imperceptible and localized perturbations, in specific areas of driving scenarios to understand their impact beyond the perturbed regions. Their extensive investigations across various models revealed significant variations of the in attack effectiveness depending on the specific architecture used. These findings have inspired us to conduct a deep investigations of the robustness of recent semantic segmentation models in real-world scenarios.

2.4 Defenses Against Real-World Attacks

Several defense strategies have been developed to improve the robustness of computer vision models against real-world attacks. In the subsequent sections, we will delve into the existing literature, with a specific focus on aspects and properties that are directly relevant to the research conducted in this thesis (5).

2.4.1 Taxonomy of Real-World Defense Mechanisms

Different surveys in the literature have reviewed recent defense mechanisms [96, 34, 88, 97]. While these papers primarily categorize methods based on their performance and target tasks, our approach will address the literature from various perspectives. In doing so, we aim to highlight the importance of considering practical aspects beyond standard defense performance. These aspects are crucial for demonstrating the applicability of a given defense in real-world scenarios, particularly in safety-critical real-time systems.

Table 2.1 summarizes the list of defense mechanisms related with the works proposed in this thesis (Chapter 5). The works have been categorized based on properties discussed in the following paragraphs.

Defense method	**Objective**	**CV Task**	**Comp. Cost**	**Evaluation**
SentiNet [98]	Det	IM	Multiple iterations + XAI algorithm	Dig
Jujutsu [99]	Det+Mask	IM	Multiple iterations + XAI algorithm	Dig
PatchGuard [100]	Det	IM	Multiple iterations	Dig
DetectorGuard [101]	Det	IM	XAI algorithm	Dig
Lance [102]	Det	IM	Multiple iterations + XAI algorithm	Dig
LGS [103]	Det	IM (+)	Filtering operation	Dig
MRD [104]	Det	IM	Multiple iterations.	Dig
PatchCleanser [57]	Det	IM	Multiple iterations	Dig
HyperNeurons [105]	Det	IM (+)	Statistical analysis	Dig
MaskNet [106]	Mask	OD (+)	Pre-filtering DNN	Dig+RW
Segment	Mask	OD (+)	Pre-filtering DNN	Dig+RW
PatchZero [59]	Mask	OD (+)	Pre-filtering DNN	Dig+RW
ZMask (ours) [107]	Det+Mask	Mult.	Statistical analysis + Multiple inferences.	Dig+RW
FPDA (ours) [24]	Det	Mult.	Statistical analysis	Dig+RW

Table 2.1: Overview of related defense mechanisms. For each method, we report: (i) the objective of the defense, which could be adversarial detection (Det) and/or adversarial masking (Mask); (ii) the computer vision task to which it is applicable, where '+' indicates that, although not tested in the original paper, the method could easily be generalized to other tasks. The absence of '(+)' indicates task specificity; (iii) notes on the computational cost of the method; and (iv) the evaluation performed to assess the correctness of the approach, which could be digital (Dig), or in real-world/simulated scenarios (RW).

Defense Objective

A first important categorization relies on the objective of the defense mechanisms. Closely related to the contributions described in this thesis, we distinguish between defenses that

perform *adversarial detection* and *adversarial masking*.

In adversarial detection the objective is to categorize samples that have been attacked, suggesting to reject the output associated with the input. In this context, the approach does not put particular effort into identifying the exact position of the real-world adversarial object, leading to the entire output being rejected. The second class of approaches, adversarial masking, aims at identifying the exact position of the real-world object, thereby masking its presence with a zero mask. This helps perform a new inference with the real-world adversarial object masked, ensuring that the output is no longer affected by possible corruptions. Figure 2.4(a) clarifies this distinction, while in the following we formalize these two objectives.

Adversarial detection A detection mechanism for adversarial patches is a binary classifier $D_d(\mathbf{x}, f)$ that outputs a binary decision, indicating whether a given image contains an adversarial attack or not, considering the victim model f. Formally, we can express $D_d(\mathbf{x}, f)$ as follows:

$$D_d(\mathbf{x}, f) = \begin{cases} \textit{True}, & \text{if } \mathbf{x} \text{ contains an adversarial attack,} \\ \textit{False}, & \text{otherwise.} \end{cases}$$

The detector D_d needs to be sensitive enough to recognize the effect of possible real-world adversarial attacks made by δ on the model f, while also limiting false alarms on not attacked images.

Adversarial masking The goal of an adversarial masking mechanism instead can be described by a function D_m, which aims at generating a mask $\mathcal{M}$ to cover the real-world adversarial object. This masking function can be expressed as follows:

$$\mathcal{M} = D_m(\tilde{\mathbf{x}}, f),$$

where D_m processes $\tilde{\mathbf{x}}$ and identifies the regions likely containing the adversarial patch, outputting a mask $\mathcal{M}$ that can be used to filter out δ from $\tilde{\mathbf{x}}$.

The mask $\mathcal{M}$ is typically a binary matrix of the same spatial dimensions of $\tilde{\mathbf{x}}$. The values in $\mathcal{M}$ indicate whether a pixel should be considered as part of the adversarial object (0) or not (1). Once $\mathcal{M}$ is computed, the cleaned image can be represented as:

$$\tilde{\mathbf{x}}_{\text{cleaned}} = \mathcal{M} \odot \tilde{\mathbf{x}} \tag{2.12}$$

The cleaned image $\tilde{\mathbf{x}}_{\text{cleaned}}$ is then processed by the victim model f, removing the effectiveness of the adversarial attack and preserving the integrity of the original image content as much as possible. Formally speaking, the previous objective can be represented through the following property:

$$\mathcal{L}(f(\tilde{\mathbf{x}} \odot \mathcal{M}), \mathbf{y}) \approx \mathcal{L}(f(\mathbf{x}), \mathbf{y}). \tag{2.13}$$

Practical considerations By definition, the adversarial masking task can be easily extended to include adversarial detection: once the location of a real-world adversarial attack is computed (i.e., the generated mask contains at least a minimum amount of zero pixels), the input can promptly be classified as adversarial.

The practical decision to use adversarial masking or detection techniques relies on the specific application context. In scenarios where adversarial attacks might dynamically emerge across multiple consecutive frames, opting for masking algorithms is reasonable. This avoids the need to reject numerous consecutive outputs, which could potentially reduce the effectivness of the AI algorithm. On the other hand, in environments where attacks are rare, non-consecutive, and where an sporadic reject of outputs is tolerable, relying on a adversarial detection mechanism could be a valuable option.

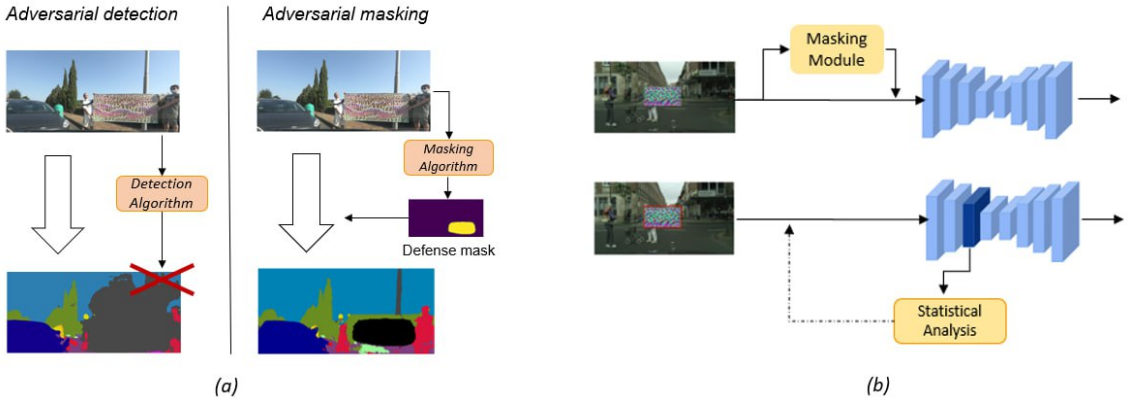


Figure 2.4: (a) A toy illustration highlighting the main differences between adversarial detection and adversarial masking. (b) A scheme of two sets of methods used to perform adversarial masking and detection. The first set (top) utilizes a masking module, which is often an image-to-image DNN as UNet [108]. This additional model is trained to compute a defense mask. In contrast, other approaches perform statistical analysis of internal features to detect any malicious or unexpected behaviors, thereby classifying the input as unsafe and allowing filtering out of the adversarial attack (as indicated by the dotted line).

Other Objectives in the Literature. For the sake of completeness, it is worth mentioning the existence of other defense objectives.

Adversarial Training. Other works [95, 106, 109, 110] investigated the use of adversarial training approaches to improve the model robustness against real-world attacks. However, unlike adversarial perturbations, the idea of adversarial training is less feasible for real-world adversarial objects. This is because the attacks based on adversarial objects (or patches) are local and perceptible, implying a highly active and dynamic region in the image. Moreover, the magnitude of noise in patch attacks is relatively much higher than norm-based perturbations. This implies that adversarial training within real-world attacks would lead to training on a divergent data distribution from the original one, impacting the decision boundary and sacrificing natural accuracy.

It is worth noting that some of the works highlighted in Table 2.1 (e.g., [106, 59, 60]) employ adversarial training techniques to train secondary models for the adversarial mask extraction. This form of adversarial training, while not altering the weights of the victim model, allows the secondary model to identify and learn patterns of features that are potentially adversarial for the victim model. Although these strategies have shown

promising results, they may lead to gradient masking issues [111, 112], thus producing a false sense of security and affecting the trustworthiness of the defense strategy.

Conversely, the methods proposed in Chapter 5 diverge from the use of additional DNNs, which introduce additional complexity and overhead. Instead, our approaches directly target properties of internal model features that shown to be susceptible to adversarial attacks. This is accomplished by running precise statistical analysis that address the presence of possible anomalous behaviours in the features space. Figure 2.4(b) clarifies this distinction.

Certified Approaches. Certifiable defenses are designed to provide provable guarantees against specific types of attacks [113, 114, 113, 115]. These defenses are typically computationally intensive, as they involve assessing extreme bounds during operations to certify the model robustness in worst-case scenarios. However, in line with the discussions in this chapter, our research prioritizes practical applicability, considering both computational efficiency and inference time. At the same time, we aim also to address crucial aspects of real-world safety-critical systems, as extracting an interpretable robustness that could practically provides a high sense of trustworthiness.

2.4.2 Practical Requirements

It is important to mention practical considerations that only a few works in the literature take into account for the design of a defense mechanisms against real-world attacks. The practical requirements listed below are essential for assessing the applicability and reliability of defense methods, especially when implemented in safety-critical systems such as autonomous robots and cars.

- **Computation time:** Due to the difficulties in providing certiable robust standalone DNNs, many AI systems are required to integrate run-time defense mechanisms at testing time. This introduces a strong need to understand the cost of the strategies as they may impact the overall inference time.

- **Transferability for different computer vision tasks:** As shown in Table 2.1, many defense mechanisms focuses on aspects only linked with image classification models. Only a limited subset of these strategies can be readily adapted to dense prediction tasks, such as object detection or semantic segmentation. This fact underscores the necessity of developing techniques that are capable of generalizing across different vision tasks, especially considering that many tasks, regardless of their specific nature, often rely on similar backbone models.

- **Exaplinable robustness and defense-aware attacks:** Another crucial aspect is the importance of providing clear interpretations of the operations perfomed by the defense strategy. These should indicate also potential failures of the approach and how an attacker might devise further adaptive attacks against it. Such transparency not only enhances the trustworthiness of the defense mechanisms but also allows a continuous refining against evolving threats.

- **Evaluation scenarios:** As for the evaluation of real-world attacks, realistic benchmarsks and tests should be taken into account. It allows assessing the effectivness of the defense mechanisms even when applied in a realistic environments.

2.4.3 Related Work on the Over-Activation Analysis

Among all the works mentioned above, [105, 116] have demonstrated that real-world adversarial attacks are strongly correlated with over-activations in network layers. More specifically, in [116], the authors attempt to establish a solid foundation for the existence of over-activation in image classifiers, particularly in relation to convolutional neural networks.

Without delving into the details of the formulations provided by [116], which will be discussed more deeply in Chapter 5, the main idea is that the patch needs to strongly over-activate features aligned with the weight of the target adversarial class. This is necessary for prevailing over all the remaning not attacked features, i.e., those not belonging to the patch. In Section 5.1, the formulation behind this idea will be reprosed from a different perspective, and adapted also for the case of dense prediction tasks, serving as a foundational concept for the contributions of this thesis.

Building upon the potential connection between over-activation and adversarial effects induced by real-world attacks, the previous works have implemented strategies concerning adversarial detection [105] or clipping layers to increase the robustness of the model [116].

In [105], the defense method relies on performing statistical analysis in internal network layers to determine if the cumulative magnitude of the features exceeds a precomputed threshold, hence deeming the input as unsafe.

In the second work, the authors in [116] implemented feature-norm clipping layers designed to mitigate the spread of over-activation caused by the patch across network's layers. While the theoretical foundation of this work is a important milestone for the research conducted in this thesis, the implementation of the feature-norm clipping layer raises some concerns. Specifically, it could be somewhat disruptive to the architecture of a pre-trained model, potentially affecting the performance and output decisions. Furthermore, the specific implementation proposed [116] assumes a predefined size for the adversarial patches, which could represent a strong assumption.

Inspired by the use of statistical analysis in the field, we noted that the approach adopted in [105] can be further adapted for dense prediction computer vision tasks, such as object detection and semantic segmentation. This adaptation and its implications are discussed in detail in Section 5.2.1. Additionally, we have expanded the application of statistical analysis to extract adversarial masks by leveraing information from multiple layers, as discussed in Section 5.3. Finally, we introduce an attention mechanism in Section 5.5, for multi-frame applications, showing to speed up the computational cost of adversarial masking defenses by leveraging temporal analysis of the over-activation.

Chapter 3

Exploring New Perspectives of DNN Testing

In the first part of the chapter, we introduce a novel approach conceived to extend the use of structural coverage criteria for run-time monitoring of DNNs. Specifically, in the following sections, we provide a discussion of the proposed run-time monitoring frameworks and define new structural coverage criteria tailored to this particular application.

Then, in the second part of the chapter, inspired by the discussion of the literature in Section 2.2, we explore new research opportunities for both DNN coverage testing and run-time coverage monitoring.

3.1 Run-time Monitoring of DNNs via Coverage Testing

Safety-critical systems, such as those used in aviation, automotive, and medical devices, are required to meet high standards due to their impact on human lives and the environment. To ensure that these systems operate as expected, rigorous testing strategies are required. However, it is also important to design proper continuous monitoring to keep track of the system reliability and safety after deployment [117, 118, 119, 120]. In the software engineering field, this process involves continuous observations and analysis of the systems at run-time, in order to detect, report, and, if necessary, respond to anomalies from the system intended behavior.

Inspired by the high similarity drawn between software algorithms and deep neural networks (Section 2.2), we propose a framework to monitor the behaviors of DNNs at run-time. Specifically, by leveraging the concept of coverage testing, where a test objective (defined by a specific coverage criterion) represents a particular model behavior, we posit that a similar approach could be also applied to interpret common patterns from known training samples and to verify whether new patterns emerging from new samples at run-time align with expected ones.

In this application scenario, the role of the coverage criterion diverges from the traditional concept of DNN coverage testing: its purpose is not to measure the patterns covered within a test set. Rather, it is intended to serve as a tool for understanding the model behavior based on previously learned data, enabling a check of whether the predictions for new inputs align with known and expected patterns of the network. Figure 3.1 provides

an illustration of our approach, contextualized within the MLOps pipeline.

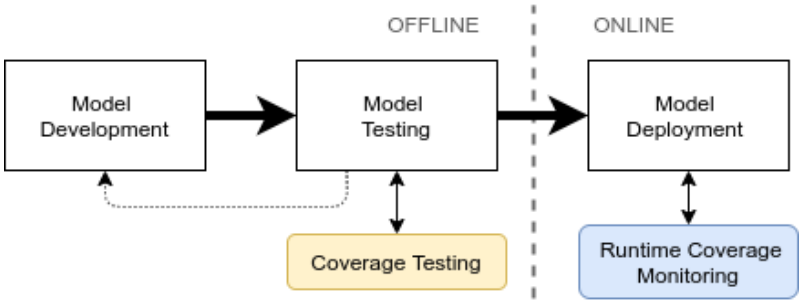


Figure 3.1: Differences between coverage testing and run-time coverage monitoring in the production pipeline of DNNs

Terminology and Notation

Before illustrating the proposed coverage monitoring pipeline and the coverage analysis methods, we further introduce some formalism specific for this chapter. This notation extends the preliminaries introduced in Section 2.1.

In this chapter, we omitted the use of the spatial and channel dimensions in internal tensors, as instead introduced in Section 2.1. Therefore, to simplify the following explanations, we can rearrange the 3D tensor representation into a 1D form by squeezing the channel and spatial dimensions, i.e., representing f as a fully-connected model.

To correctly keep track of all the neuron indices in the model f, we denote the j-th neuron at layer L_l by $n_{l,j} \in N_l$, where N_l is the subset of all the neurons at layer L_l. The whole set of neurons is defined as $N = \{n_{l,j} \mid l \in \{1, \dots, L_{N_L}\}, j \in \{1, \dots, N_l\}\}$.

Let $V_l(\mathbf{x}) \in \mathbb{R}^{(C^l \cdot H^l \cdot W^l)}$ be the vector denoting the activations of the neurons in layer L_l for an input $\mathbf{x}$, i.e., $V_l(\mathbf{x}) = f^{0 \to l}(\mathbf{x})$ and let $v_{l,j}(\mathbf{x})$ be the j-th element of $V_l(\mathbf{x})$, the neuron activation value is hence defined as

$$v_{l,j} = \phi_l(u_{l,j}) \qquad \text{with} \qquad u_{l,j} = b_{l,j} + \sum w_{l,h,j} \cdot v_{l-1,h}, \tag{3.1}$$

where $w_{l,h,j}$ is the *weight* of the connection between neurons $n_{l-1,h}$ and $n_{l,j}$, $b_{l,j}$ is the *bias* term of neuron $n_{l,j}$, and ϕ_l is the activation function.

This work focuses on classification tasks, where the DNN associates a generic input $\mathbf{x}$ to a class $\hat{y}$ belonging to a set of N_y classes.

Note that, the formalism adopted is a generalization of the approach presented in Section 2.1 and can be used also with convolutional neural networks (CNNs) [121].

3.1.1 Proposed Monitoring Framework

The core concept of the proposed monitoring architecture is to identify a series of activation patterns that are considered safe based on a given coverage metric. Subsequently, at runtime, the activation patterns from new inputs are compared against these safe patterns, measuring the degree of match using a specific confidence metric.

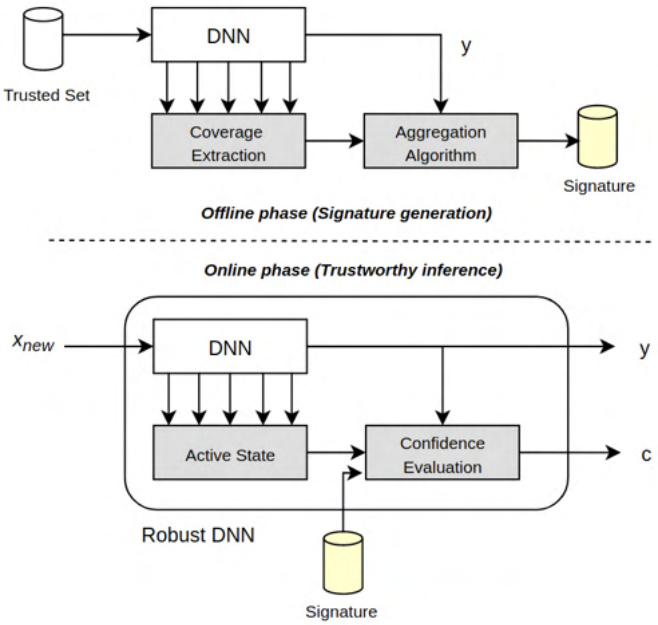


Figure 3.2: Overview of the monitoring architecture with its offline and online phases. Grey boxes denote meta-algorithms, i.e., those whose behavior depend on the selected coverage criterion.

The workflow of the proposed architecture is illustrated in Figure 3.2 and consists of an offline phase (also called *Signature generation*) and an online phase (also called *Trustworthy inference*).

Offline phase (*Signature generation*). In this phase, the network processes a set of trusted inputs, denoted by *Trusted Set*, that generate correct network outputs with a high prediction score. Each sample in the Trusted Set is used to perform inference on the target trained DNN. During inference, the intermediate results produced by the various layers of the DNN (e.g., the neuron outputs) are collected and used to apply a certain coverage paradigm, denoted by *Coverage Analysis Method* (CAM). The coverage results across all inputs in the Trusted Set are then grouped by the corresponding output classes and aggregated to produce a representation of the covered activation patterns (by the Aggregation Algorithm). This representation is then compressed to produce a file called *DNN Signature*. The DNN Signature encodes all network activation patterns that are deemed safe for each output label, under a certain CAM. Both the Coverage Extraction and the Aggregation Algorithm depend on the selected CAM.

Online phase (*Trustworthy inference*). The network is deployed together with a DNN Signature. At inference time, given a new input x_{new}, the same CAM used to generate the DNN Signature is applied to extract the coverage result generated by x_{new}, which is referred to as *DNN Active State*. Given the class $\hat{y}$ predicted for x_{new}, a Confidence Evaluation Algorithm computes the matching level between the Active State stimulated by x_{new} and the trusted one encoded in the DNN Signature corresponding to class $\hat{y}$, producing a confidence value for the prediction. When such a confidence is below a given threshold, the input x_{new} is deemed unsafe. The configuration of such a threshold is addressed later. The Confidence Evaluation Algorithm also depends on the selected CAM.

Notes on the low level implementation with Caffe

With the goal of deploying the proposed solution on constrained devices, the initial implementation of our architecture was developed as an extension of the Caffe framework [122].[1]

The extension introduces a new type of layer in Caffe, called *Coverage Layer* (CV-Layer), which operates in a transparent (i.e., pass-through) fashion, hence not altering the DNN hyperparameters. The CV-Layer applies a coverage criterion at inference time based on the tensor data it receives in input and is used during both the offline and online phases. When installing a CV-Layer, given that it operates as a pass-through component, connections between two layers, say L_l and L_{l+1}, can be preserved by connecting (i) the output of L_l to the input of the CV-Layer and (ii) the output of the CV-Layer to the input of L_{l+1}. Multiple CV-Layers can be installed depending on the behavior of the selected CAM. For instance, CAMs that work by analyzing all neuron outputs of the network require the installation of a CV-Layer after each layer of the original DNN. An example installation of the CV-Layer is illustrated in Figure 3.3b.

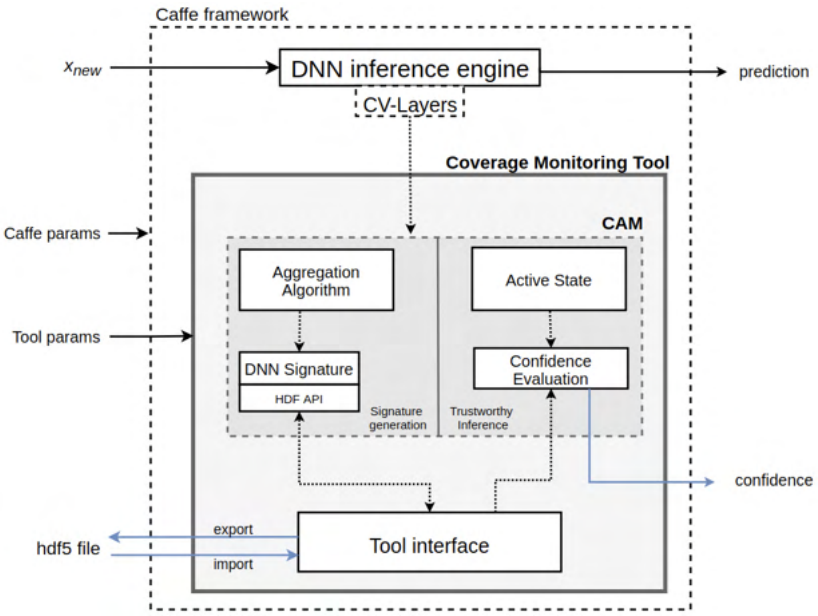
Figure 3.3a illustrates the interconnection between the principal software blocks of the tool and the Caffe framework. Under Caffe, networks are distributed via a `prototxt` model file, which should include the CV-layers in the network architecture, and a `caffemodel` file, which contains the trained weights (both referred to as 'Caffe params' in the figure). Regarding the tool, the required parameters (referred to as 'Tool params' in the figure) are a specification of the Trusted Set and the identifier of the selected CAM (to be chosen among those supported by the tool). The tool will then instantiate the implementation of the monitoring functions associated to the selected CAM.

Finally, the tool is capable of exporting and importing the DNN Signature using the hdf5 file format by means of the hierarchical data format (HDF) API: It manages the interaction with the DNN Signature, as the export at the end of the offline phase, or the import at the beginning of the online phase. The tool also allows reading the confidence evaluation outcome during the online phase, which represents the confidence metric of the applied CAM.

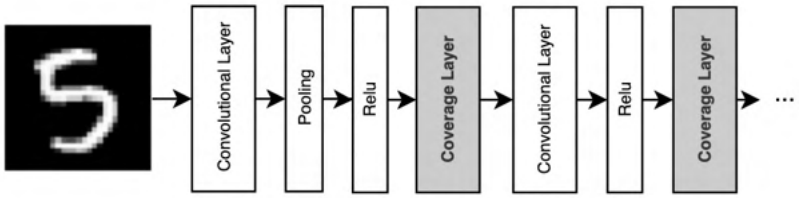
Each CAM is developed both in CUDA and C++. Since the CV-Layers are totally compliant with the other Caffe layers, the DNN framework will take care of forwarding tensors to the selected implementation based on the available architecture (e.g., GPU or CPU). At the same time, it is up to the tool to select the device implementation of the Confidence Evaluation Algorithms.

Another feature offered by the tool is the possibility of computing the signature in the offline phase using a mini-batch approach. The Trusted Set is split into multiple mini-batches where each one represents a single, but large input tensor that is analyzed by the CV-Layer on the same inference pass. The values extracted from the inputs of the mini-batch, according to the selected CAM, are then merged into the DNN Signature with the results obtained by previously-processed mini-batches. This approach significantly allows speeding up the creation of the DNN Signature creation time, especially when it is possible to parallelize the procedure.

[1]The rationale behind implementing the proposed mechanism in Caffe at the time was based on the advantages of the low-level (C and CUDA) optimization provided by Caffe. This approach enables the mechanism to also address constrained and embedded platforms effectively.



(a)



(b)

Figure 3.3: (a) Overview of the tool architecture. (b) Example network architecture with two CV-Layers installed after two ReLu activation layers.

3.1.2 Coverage Analysis Methods

This section presents three CAMs together with the corresponding algorithms required to instantiate the monitoring architecture illustrated in Figure 3.2.

Note that classic structural coverage crtieria for DNNs, introduced in Section 2.2, were conceived for the purpose of DNN coverage testing. Nevertheless, they inspire the following CAMs, which are instead designed for being applied to the monitoring scheme proposed in Section 3.1.1.

Single-Range Coverage (SRC)

This CAM is based on the Neuron Boundary Coverage criterion [32] and works by analyzing the range of output values produced by each neuron. During the offline phase, the minimum and maximum output values produced by the neurons when testing the Trusted Set are recorded to set the range of the "typical" behavior of the network when stimulated by trusted inputs.

The Trusted Set S is split into N_y subsets $S_1, ..., S_{N_y}$, one for each class, where S_i

denotes the set of inputs in S belonging to the i-th class. The DNN Signature σ_i for the i-th class (with $i = 1, \ldots, N_y$) is a collection of pairs $\sigma_{i,l,j} = (v^{\min}_{i,l,j}, v^{\max}_{i,l,j})$, where $v^{\min}_{i,l,j}$ and $v^{\max}_{i,l,j}$ denote the minimum and maximum output values produced by neuron $n_{l,j}$, respectively, over all inputs in S_i. The Aggregation Algorithm of SRC is summarized in Algorithm 1.

Algorithm 1 Aggregation Algorithm of SRC.

Require: Trusted Set S, trained DNN
Ensure: DNN Signature σ

Algorithm:
for $S_i \in S$ **do**
  $\sigma_i = \{\}$
  for $n_{l,j} \in N$ **do**
    $v^{\min}_{i,l,j} = \min_{x \in S_i}\{v_{l,j}(x)\}$
    $v^{\max}_{i,l,j} = \max_{x \in S_i}\{v_{l,j}(x)\}$
    $\sigma_{i,l,j} = (v^{\min}_{i,l,j}, v^{\max}_{i,l,j})$
    Add $\sigma_{i,l,j}$ to σ_i
  end for
end for
return $\sigma = \{\sigma_1, \ldots, \sigma_m\}$

During the online phase, given a new input x_{new} and the corresponding class $\hat{y}$ predicted by the DNN, the DNN Signature $\sigma_{\hat{y}}$ is compared against the output values produced by the neurons, denoted as the DNN Active State. The Confidence Evaluation Algorithm of SRC returns a confidence value that is computed as a function of the number of neurons whose output value $v_{l,j}(x_{\text{new}})$ is outside the range specified by pair $\sigma_{\hat{y},l,j}$. The confidence is computed as $c = \exp(-\frac{\eta \cdot ln(2)}{\tau_{\hat{y}}})$, where η is the number of neurons out of range and $\tau_{\hat{y}}$ is a class-dependent parameter, called *threshold*, which tunes the slope of the exponential function (the higher the threshold, the faster the function goes to zero). Note that, if $\eta = \tau_{\hat{y}}$, then $c = 0.5$. The specific values for parameters $\tau_{\hat{y}}$ are set with a calibration procedure presented in Section 3.1.3. The rationale of this formula is to map the computed coverage cost η to a confidence value between 0 and 1, to be compliant with typical softmax scores. A smooth exponential function is adopted also because it allows assigning a high confidence when just a few values fall outside the ranges in the signature.

This procedure is reported in Algorithm 2, assuming to monitor all the network's neurons N. In practice, to balance performance with computation time, all these algorithms can also be executed on a subset of N (see Section 3.1.1).

Multi-Range Coverage (MRC)

This CAM is based on the K-Multi-Section coverage criterion [32] and extends SRC by introducing multiple ranges to analyze the output of the neurons. The offline phase works as follows. First, as for SRC, the minimum and maximum output values produced by the neurons when testing the Trusted Set are recorded. Then, each output range is evenly

Algorithm 2 Confidence Evaluation Algorithm of SRC.

Require: input x_{new}, DNN Signature σ, trained DNN, thresholds τ_i
Ensure: confidence c

1: ---

2: $\hat{y} = \text{argmax}_{1 \leq j \leq N_{class}}\{f(x_{\text{new}})\}$
3: $\eta = 0$
4: **for** $n_{l,j} \in N$ **do**
5: Extract $\sigma_{\hat{y},l,j}$ from $\sigma_{\hat{y}}$
6: $(v^{\min}_{\hat{y},l,j}, v^{\max}_{\hat{y},l,j}) = \sigma_{\hat{y},l,j}$
7: **if** $v_{l,j}(x_{\text{new}}) \notin [v^{\min}_{\hat{y},l,j}, v^{\max}_{\hat{y},l,j}]$ **then**
8: η ++
9: **end if**
10: **end for**
11: **return** $c = \exp(-\frac{\eta \cdot \ln(2)}{\tau_{\hat{y}}})$

split into Q sub-ranges. Finally, for all the inputs of the Trusted Set, the algorithm counts the number of times in which the neuron outputs fall in a given sub-range.

Formally, using the same notation introduced for SRC, for each output class with index i, the output range of each neuron $n_{l,j}$ is split into Q sub-ranges of size $\Delta_{i,l,j} = (v^{\max}_{i,l,j} - v^{\min}_{i,l,j})/Q$. For convenience, the last of such sub-ranges is defined as $[v^{\min}_{i,k,j} + (Q-1)\Delta_{i,l,j}, v^{\max}_{i,l,j}]$, while the others have open right endpoints. The DNN Signature σ_i for the i-th class is a collection of tuples

$$\sigma_{i,l,j} = (v^{\min}_{i,l,j}, v^{\max}_{i,l,j}, \lambda^1_{i,l,j}, \lambda^2_{i,l,j}, \ldots, \lambda^Q_{i,l,j}),$$

where $\lambda^q_{i,l,j} \in [0,1]$ (with $q = 1, \ldots, Q$) is given by the number of times the output of neuron $n_{l,j}$ falls within the q-th sub-range when the network is stimulated by inputs in S_i, divided by the cardinality of S_i itself. The Aggregation Algorithm of MRC, reported in Algorithm 3, first computes the minimum and maximum output values of the neurons, and then the corresponding occurrencies to produce the DNN Signature.

The idea behind the online phase of MRC is that the more a new input x_{new} produces neuron output values outside the sub-ranges matched by the inputs in the Trusted Set, the more x_{new} is likely to be unsafe. Given x_{new}, the DNN Active State is composed of the identifiers of the sub-ranges to which the output of the neurons belong to, if any. This is used to assign a cost $\Theta_{l,j}(x_{\text{new}}) \in [0,1]$ to each neuron $n_{l,j}$ that quantifies how much the neuron output is deemed unsafe. Being i the index of the class assigned to x_{new}, this cost is formally defined as

$$\Theta_{k,j}(x_{\text{new}}) = \begin{cases} 1, & \text{if } v_{n_{l,j}}(x_{\text{new}}) \notin [v^{\min}_{i,l,j}, v^{\max}_{i,l,j}] \\ 1 - \lambda^{q^*}_{i,l,j}, & \text{otherwise,} \end{cases} \tag{3.2}$$

where q^* denotes the index of the sub-range of $n_{l,j}$ to which $v_{n_{l,j}}(x_{\text{new}})$ belongs to.

The Confidence Evaluation Algorithm, reported in Algorithm 4, computes the sum of such costs $\Theta_{l,j}(x_{\text{new}})$ over all neurons, denoted here as η. Hence, the confidence score is computed as $c = \exp(-\frac{\eta \cdot ln(2)}{\tau_{\hat{y}}})$.

Algorithm 3 Aggregation Algorithm of MRC.

Require: Trusted Set S, trained DNN, number of sections Q
Ensure: DNN Signature σ

1: —

2: **for** $S_i \in S$ **do**
3: $\sigma_i = \{\}$
4: **for** $n_{l,j} \in N$ **do**
5: $v^{\min}_{i,l,j} = \min_{x \in S_i}\{v_{l,j}(x)\}$
6: $v^{\max}_{i,kl,j} = \max_{x \in S_i}\{v_{l,j}(x)\}$
7: $\Delta_{i,l,j} = (v^{\max}_{i,l,j} - v^{\min}_{i,l,j})/Q$
8: $\lambda^1_{i,l,j} = 0, \lambda^2_{i,l,j} = 0, \ldots, \lambda^Q_{i,l,j} = 0$
9: **for** $x \in S_i$ **do**
10: $q = \max\{1, \lceil (v_{l,j}(x) - v^{\min}_{i,l,j})/\Delta_{i,l,j} \rceil\}$
11: $\lambda^q_{i,l,j}{+}{+}$
12: **end for**
13: $\lambda^1_{i,l,j}/ = |S_i|, \lambda^2_{i,l,j}/ = |S_i|, \ldots, \lambda^Q_{i,l,j}/ = |S_i|$
14: $\sigma_{i,l,j} = (v^{\min}_{i,l,j}, v^{\max}_{i,l,j}, \lambda^1_{i,l,j}, \lambda^2_{i,l,j}, \ldots, \lambda^Q_{i,l,j})$
15: Add $\sigma_{i,l,j}$ to σ_i
16: **end for**
17: **end for**
18: **return** $\sigma = \{\sigma_1, \ldots, \sigma_{N_{class}}\}$

k-Nearest Neighbors Coverage (kNNC)

This CAM is based on the DkNN algorithm presented in [123], which was originally proposed as an alternative inference logic and not as a way to detect unsafe inputs. The key idea of the kNNC is to compute a confidence score by applying the k-Nearest Neighbors (kNN) algorithm on the output values produced by the neurons of the various DNN layers.

For each output class with index i, the DNN Signature is composed of a collections of sets $\sigma_{i,l}$ of vectors, one for each layer L_l. Each set $\sigma_{i,l}$ is composed by aggregating the vectors $V_l(x)$ obtained from all inputs $x \in S_i$. The corresponding procedure is reported in Algorithm 5.

During the online phase of kNNC, given a new input x_{new} and its corresponding predicted class $\hat{y}$, the kNN algorithm is used to compute the confidence value. In the following, the parameter that controls the kNN algorithm is denoted as G, since l denotes the layer index. Hence, the G vectors $V_k(x)$ stored in the DNN Signature that are nearest to $V_l(x_{\text{new}})$ are identified by the kNN algorithm and their corresponding output class indexes are recorded and stored in a multiset Ω_l. Finally, the number of occurrences of the class index $\hat{y}$ in Ω_l is counted and denoted as η. Hence, the output confidence score is computed as $c = exp(-\frac{\eta \cdot ln(2)}{\tau_{\hat{y}}})$. The rationale behind this computation is the following. Given inputs x classified to the $\hat{y}$-th class by the DNN, the more vectors $V_l(x)$ in the signature are among the G closest ones to $V_l(x_{\text{new}})$, the more the active state produced by x_{new} resembles the ones produced by the trusted inputs that the DNN classifies as for x_{new}, hence positively contributing to the confidence value. The overall procedure is reported

Algorithm 4 Confidence Evaluation Algorithm of MRC.

Require: input x_{new}, DNN Signature σ, trained DNN, thresholds τ_i, number of sections Q
Ensure: confidence c

$\hat{y} = \text{argmax}_{1 \leq j \leq N_{class}}\{f(x_{\text{new}})\}$
$\eta = 0$
for $n_{k,j} \in N$ **do**
  Extract $\sigma_{\hat{y},l,j}$ from σ_i
  Extract $v^{\min}_{\hat{y},l,j}$ and $v^{\max}_{\hat{y},l,j}$ from $\sigma_{\hat{y},l,j}$
  if $v_{l,j}(x_{\text{new}}) \notin [v^{\min}_{\hat{y},l,j}, v^{\max}_{\hat{y},l,j}]$ **then**
    η += 1
  else
    $\Delta_{\hat{y},l,j} = (v^{\max}_{\hat{y},l,j} - v^{\min}_{\hat{y},l,j})/Q$
    $q^* = \max\{1, \lceil (v_{l,j}(x_{\text{new}}) - v^{\min}_{\hat{y},l,j})/\Delta_{\hat{y},l,j} \rceil\}$
    Extract $\lambda^{q^*}_{\hat{y},l,j}$ from $\sigma_{\hat{y},l,j}$
    η += $1 - \lambda^{q^*}_{\hat{y},l,j}$
  end if
end for
return $c = \exp(-\frac{\eta \cdot \ln(2)}{\tau_{\hat{y}}})$

Algorithm 5 Aggregation Algorithm of kNNC.

Require: Trusted Set S, trained DNN
Ensure: DNN Signature σ

for $S_i \in S$ **do**
  $\forall l = 1, \ldots, N_L,\ \sigma_{i,l} = \{\}$
  for $x \in S_i$ **do**
    for $L_l \in L$ **do**
      Add $V_l(x)$ to $\sigma_{i,l}$
    end for
  end for
  $\sigma_i = \{\sigma_{i,1}, \ldots, \sigma_{i,N_L}\}$
end for
return $\sigma = \{\sigma_1, \ldots, \sigma_{N_L}\}$

in Algorithm 6.

Illustrative example

To illustrate the proposed approach, let us consider a simple example based on a LeNet-4 [124] CNN trained on the MNIST dataset [125]. The Trusted Set consists of those

Algorithm 6 Confidence Evaluation of kNNC.

Require: input x_{new}, DNN Signature σ, trained DNN, thresholds τ_i, number of nearest neighbors G

Ensure: confidence c

$\hat{y} = \text{argmax}_{1 \leq j \leq N_{class}}\{f(x_{\text{new}})\}$
$\eta = 0$
for $L_k \in L$ **do**
  $\Omega_l = \text{kNN}(G, V_l(x_{\text{new}}), \sigma_{1,l}, \ldots, \sigma_{N_{class},l})$
  $\eta \mathrel{+}= |y \in \Omega_l : y = \hat{y}|$
end for
return $c = \exp(-\frac{\eta \cdot \ln(2)}{\tau_{\hat{y}}})$

MNIST samples that are correctly predicted by the network with a prediction score higher than 0.9. The selected CAM is the SRC presented in Section 3.1.2. Unsafe inputs are generated using the FGSM [12], setting $\epsilon = 0.05$ to obtain adversarial examples with small perturbations.

For simplicity, the SRC method is only applied to the first convolutional block, including a convolutional layer, a pooling layer, and a ReLU activation layer. Figure 3.4a illustrates a trusted input (i.e., a digit correctly classified as '8' with a prediction score of 0.974), which is not part of the Trusted Set, together with a histogram that reports the number of neurons per channel that have an output value outside the corresponding ranges $[v^{\min}_{i,l,j}, v^{\max}_{i,l,j}]$ representing the DNN Signature for class '8' under the SRC. In particular, note that only two neurons have outputs that do not match the signature, producing a prediction confidence $c = \exp(-\frac{\mathbf{2} \cdot ln(2)}{\tau_8}) = 0.8705$, where in this example all the thresholds τ_i are set to 10. Conversely, Figure 3.4b illustrates an adversarial example, incorrectly classified as '6' (with a prediction score of 0.918), together with the corresponding histogram. In this case, there are 63 neurons in the first block whose activations are outside the ranges stored in the DNN Signature for label '6', resulting in a prediction confidence $c = \exp(-\frac{\mathbf{63} \cdot ln(2)}{\tau_6}) = 0.0126$.

3.1.3 Experimental evaluation

This section reports the results of a set of experiments conducted to assess the performance of the proposed methods in terms of running time, memory footprint, and capability of detecting unsafe network inputs. Several types of unsafe inputs generated with state-of-the-art methods have been considered, which are summarized in Section 3.1.3. Before discussing the actual experimental results, Section 3.1.3 presents the experimental setting, while Section 3.1.3 presents the calibration of thresholds used.

Unsafe inputs generation

In recent years, many adversarial attacks [16] have been disclosed to efficiently generate adversarial examples (AEs) while minimizing the perturbation to be applied to turn a safe

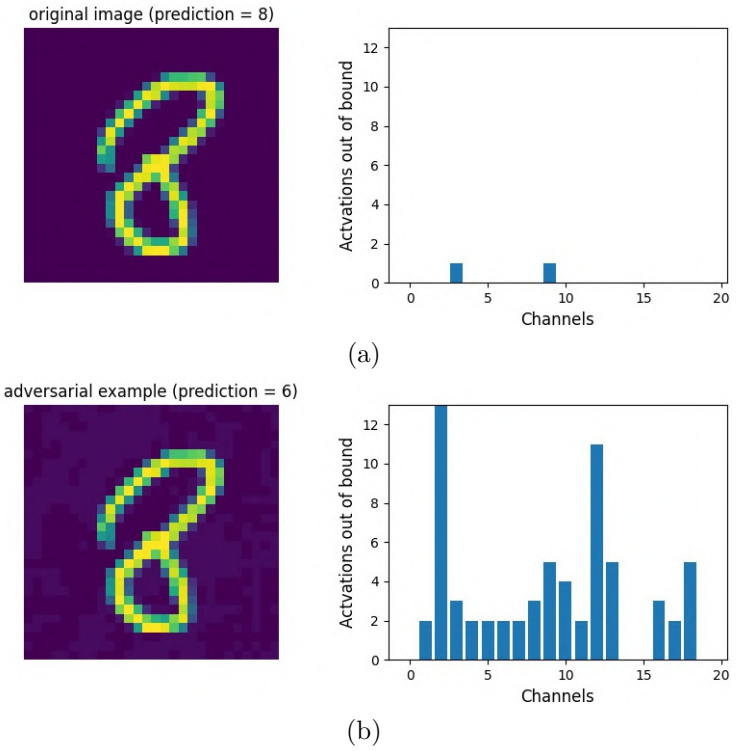


Figure 3.4: (a) A genuine sample and its corresponding activation histogram with respect to a DNN Signature obtained in the first convolutional block of LeNet, under the SRC. (b) An adversarial example and its corresponding activation histogram against the same DNN Signature.

input into an unsafe one. Besides AEs, a network can also fail for other types of unsafe inputs.

Adversarial examples. We considered the following attack methods under different settings for crafting both small and medium perturbations. The amount of perturbation is usually refereed with the ϵ parameter. Among the first and famous adversarial attacks, one is FGSM [12], which crafts adversarial perturbation in one shot. More advanced adversarial attacks have also been considered: they are PGD [51], BIM [54] and CW [55], which implement iterative methods capable of improving the attack effectiveness at each iteration. The number of iterations and the amount of perturbation introduced at each iteration are denoted by k and α, respectively. The attacks are configured as reported in Table 3.2.

Out-of-distribution unsafe inputs. Another class of unsafe inputs are those that are far from the distribution of data used to train the network, but are still predicted with a high score by the model. They commonly exist because a DNN works as a global classifier for the whole input space (e.g., all possible images of a given size and format). To test this class of unsafe inputs, we generated new samples with large perturbations, obtained with a variant of the FGSM attack, which are applied to samples in the training set of the tested networks. In particular, using a *multi-step targeted FGSM* method starting from

Dataset	Model	Accuracy
MNIST	conv(20,5,1) - ReLU - **CV** - MaxPool(2,2) - conv(50,5,1) - ReLU - **CV** - fc(500) - ReLU - **CV** - fc(10)	0.9907
F-MNIST	conv(20,5,1) - ReLU - **CV** - MaxPool(2,2) - conv(50,5,1) - ReLU - **CV** - fc(500) - ReLU - **CV** - fc(10)	0.9106

Table 3.1: Network models applied for the experimental evaluation. 'CV' is a Coverage Layer.

safe inputs, given a target label y_{Adv}, the approach can be formulated as:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} - \frac{\epsilon}{(k+1)^2} \cdot sign(\bigtriangledown_{\tilde{\mathbf{x}}_{t-1}} \mathcal{L}(f(\tilde{\mathbf{x}}_{t-1}), y_{\text{adv}})), \tag{3.3}$$

where $\tilde{x}_0 = x$. The idea is to move the perturbation in the opposite direction of the loss function gradient related to the input x and the target label y_{adv}. In this way, the perturbation leads the network to increase the confidence score of the target class. Note that, differently by the others iterative adversarial attacks presented above, here the perturbation is not clipped at each iteration within ϵ. In fact, the step size of the perturbation is scaled proportionally with k, such that the method generates large perturbations during the first step, and then smaller perturbations in the last ones to consolidate the prediction of the model with the target class. In this work, this iterative generation has been stopped when obtaining input samples that the network associates to y_{Adv} with a corresponding softmax score higher than or equal to 0.99.

Adversarial patches. Finally, we also studied the effect of adversarial patches. This are crafted following an iterative approach as shown in Section 2.3.1. Due to the simplicity of the addressed dataset here with respect to driving scenarios or large-scale images (COCO or Imagenet), we reduce the set of transformations to rotation and scaling, while the tested patches have a size of 8 × 8 pixels and are always placed at the top-left corner of the image.[2]

Experimental setting

The datasets considered in the experimental evaluation are MNIST [125] and F-MNIST [126]. MNIST is a dataset of handwritten digits while F-MNIST is a dataset of clothes articles. Both datasets contain grayscale images of 28x28 pixels, and provide 60,000 images for training and 10,000 images for testing. F-MNIST is slightly more complex than MNIST and DNNs usually achieve lower accuracy in classifying its images. Both the datasets have been processed with a LeNet-4 [124] CNN trained on 8 epochs using the Adam optimization algorithm [127] and a cross entropy loss function.

Please note that, within the scope of our work, we acknowledge that the models and datasets used are baseline with respect to state-of-the-art models and datasets. Despite potential concerns about the scalability of the evaluation, better discussed in the next section, we opted to align our work with application scenarios and settings used with previous works on classic structural coverage criteria.

Three Coverage Layers per network have been installed (different installations of Coverage Layers are analyzed later). Table 3.1 summarizes the network architectures and their corresponding classification accuracy on the original testing sets.

The experimental evaluation used the following sets of inputs:

[2]The size and position are decided to not overlapping the main content of images in MNIST and FMNIST.

- *Trusted Set*: it is the one used to generate the DNN Signature and has been obtained by selecting from the original *training set* those samples that the DNN classifies correctly with a prediction score (i.e., the softmax probability) larger than 0.9. This set contains 59309 and 52680 samples for MNIST and F-MNIST, respectively.

- *Trusted Test Set*: it is the one used to assess the classification performance of the DNNs when enhanced with one of the proposed CAMs. It is obtained by selecting from the original *testing set* those samples that the DNN classifies correctly with a prediction score > 0.9. This set contains 9000 samples.

- *Adversarial Set*: it is the one employed for evaluating the performance of the proposed methods in detecting unsafe inputs. It contains unsafe inputs for which the DNN makes a wrong prediction. Different definitions of this set have been tested depending on the selected attack method (see Section 3.1.3): they are summarized in Table 3.2 for both MNIST and F-MNIST. A prediction is considered to be wrong when the DNN classifies such inputs in a wrong class with a prediction score larger than 0.8, for AEs and adversarial patches, or larger than 0.99, for out-of-distribution inputs. Such thresholds have been selected by empirically finding the largest values that allowed generating a sufficient number of unsafe inputs to perform the experiments. Note that the higher the thresholds the more difficult the generation of AEs. Conversely, when using lower thresholds, the generated AEs tend to be less relevant, as they could be simply discarded by comparing the softmax score produced by the network against a certain threshold.

	MNIST			**F-MNIST**		
Attack method	Attack parameters	# of samples for calibration	# of samples for evaluation	Attack parameters	# of samples for calibration	# of samples for evaluation
FGSM-1	$\epsilon = 0.1$	500	500	$\epsilon = 0.05$	600	3400
FGSM-2	$\epsilon = 0.2$	800	3200	$\epsilon = 0.1$	600	1400
PGD-1	$(\epsilon, \alpha, k) = (0.1, 0.015, 40)$	800	1800	$(\epsilon, \alpha, k) = (0.03, 0.015, 40)$	600	3400
PGD-2	$(\epsilon, \alpha, k) = (0.18, 0.015, 40)$	800	3200	$(\epsilon, \alpha, k) = (0.10, 0.015, 40)$	600	3400
BIM-1	$(\epsilon, \alpha, k) = (0.05, 0.004, 10)$	450	500	$(\epsilon, \alpha, k) = (0.015, 0.004, 10)$	600	1500
BIM-2	$(\epsilon, \alpha, k) = (0.18, 0.004, 10)$	800	3200	$(\epsilon, \alpha, k) = (0.10, 0.004, 10)$	600	3400
CW	$k = 500$	450	500	$k = 500$	200	300
Out of Dis.	$\epsilon = 0.02, k = 80$	-	4000	$\epsilon = 0.02, k = 80$	-	4000
Patch	$\epsilon = 0.02, k = 200$	-	100	$\epsilon = 0.02, k = 200$	-	1000

Table 3.2: Parameters and settings used to generate the Adversarial Set.

In the following, the experimental results are reported for four CAMs: SRC, two versions of MRC with $Q = \{16, 32\}$, and kNNC with $G = 75$. Furthermore, we also compared the results against state-of-the-art run-time methods used for detecting AEs: FeaturesSqueezing [128] and VisionGuard [129].

Threshold calibration

This experimental evaluation was focused on a binary classification of the network inputs, i.e., either an input was deemed *safe* and the prediction made by the network was accepted, or the input was deemed *unsafe*, and the network prediction was rejected. This has been implemented by calibrating the tolerances τ_i used by the various Confidence Evaluation Algorithms so that an input is deemed safe if $c \geq 0.5$ and unsafe otherwise.

The calibration of thresholds has been performed using Receiver Operating Characteristic (ROC) analysis to compute the values τ_i, for each class with index $i = 1, \ldots, N_y$, that

	MNIST						F-MNIST					
Input type	*SRC*	*MRC-32*	*MRC-16*	*kNNC*	*VG*	*FS*	*SRC*	*MRC-32*	*MRC-16*	*kNNC*	*VG*	*FS*
FGSM-1	0.998	0.976	0.946	0.958	0.986	0.976	0.855	0.848	0.827	0.915	0.880	0.881
FGSM-2	1.0	0.994	0.989	0.941	0.930	0.936	0.929	0.952	0.937	0.883	0.785	0.814
PGD-1	0.989	0.968	0.943	0.970	0.989	0.976	0.852	0.841	0.792	0.935	0.901	0.973
PGD-2	1.0	0.919	0.852	0.955	0.810	0.882	0.979	0.942	0.943	0.859	0.175	0.325
BIM-1	0.968	0.943	0.912	0.964	1.0	0.985	0.842	0.848	0.789	0.916	0.997	1.0
BIM-2	0.995	0.861	0.815	0.952	0.8425	0.870	0.986	0.924	0.939	0.831	0.161	0.239
CW	0.971	0.959	0.933	0.960	1.0	0.989	0.813	0.829	0.822	0.907	0.993	1.0
Out of Dis.	1.0	1.0	1.0	0.346	0.0	0.0	1.0	1.0	1.0	0.842	0.00	0.00
Patch	1.0	0.891	0.923	0.995	0.957	0.984	1.0	0.911	0.858	0.951	0.962	0.975
Safe Samples	0.991	0.951	0.919	0.955	0.925	0.938	0.89	0.892	0.851	0.921	0.671	0.732

Table 3.3: Detection accuracy of all the tested methods on MNIST and F-MNIST. VG and FS correspond to Vision Guard [129] and FeatureSqueezing [128], respectively.

represent the best balance between minimizing the inputs that are wrongly rejected and accepted. In this regard, portions of the Trusted Test Set and the Adversarial Set introduced in the previous section have been used to test safe and unsafe inputs, respectively, during a calibration based on a simple ROC curves analysis (see supplementary material in [130]).

Detection performance

Experiments have been conducted to evaluate the performance of the four CAMs in correctly detecting whether an input is safe or unsafe using the thresholds τ_i calibrated as described above. Tables 3.3 reports the detection accuracy, defined as the ratio of inputs *correctly* classified as either safe or unsafe. The tables report the breakdown of unsafe inputs for all the methods introduced in Section 3.1.3 (see Table 3.2 for the number of samples used for the evaluation). The last rows of the tables are related to safe inputs, which correspond to the remaining inputs of the Trusted Test Set that have not been considered for calibration.

Although SRC is the simplest CAM, for inputs of the MNIST dataset it is often capable of detecting unsafe inputs better than the others CAMs. The only exception pertains to the BIM attack, but the performance of SRC is anyway extremely close to the one of kNNC. Note that SRC and MRC are also capable of detecting all unsafe out-of-distribution inputs, while kNNC exhibits very poor performance (close to coin tossing) for that kind of inputs.

The results for the F-MNIST dataset are quite different. In this case, there is no dominating CAM for AEs. Note that kNNC performs quite well with AEs generated using low perturbations (e.g., CW and BIM), while SRC significantly outperforms kNNC for unsafe inputs generated using the PGD attack. SRC and MRC are again capable of detecting all unsafe out-of-distribution inputs. This suggests that, given proper accelerated implementations, multiple CAMs could be used in parallel to improve the overall detection performance of unsafe inputs.

Table 3.3 also provides the results achieved with Vision Guard and FeatureSqueezing. They always obtain high performance with low-perturbation attacks on both MNIST and FMNIST, but fail when the magnitude of the perturbation increases (e.g., PGD-2 and BIM-2, which find more effective AEs). Moreover, they report worse performance on safe samples, since they are incorrectly rejected more times than the proposed CAMs. Most interestingly, Vision Guard and FeatureSqueezing totally fail with out-of-distributions

unsafe inputs, while three of the proposed CAMs are very effective.

Running time and memory footprint

With respect to a regular deployment of a DNN, each of the proposed CAMs introduces an additional inference time to execute the Confidence Evaluation Algorithm (implemented by CV-Layers) and an additional memory footprint to store the DNN Signature. This section is focused on evaluating these overheads, which in some cases may be very important in selecting the most appropriate CAM. For instance, to deploy a CAM on a resource-constrained, embedded device that has to operate in real time, it is essential to contain as much as possible the additional inference time and the memory footprint, even accepting reduced detection performance.

Additional inference time. The experiments for SRC, MRC, and NRC have been performed on a machine equipped with an Intel(R) Core(TM) i5-4670K CPU @ 3.40GHz, 8GB of RAM, and an NVIDIA GeForce GTX770 GPU. Due to their demanding memory requirements (see the results at the end of this section), the experiments for kNNC have been performed on a Nvidia DGX Station V100. The implementation presented in Section 3.1.1 has been used[3]. Figure 3.5 reports the additional inference time introduced by the four CAMs together with the corresponding detection performance for a representative setting (F-MNIST dataset and unsafe inputs generated by the FGSM-2 attack). Note that the y-axis at the left of the figure has a logarithmic scale. The figure reports the results for different installations of the CV-Layers, denoted by the sets reported on the x-axis of the plot. They correspond to cases in which the CAMs operate on a subset of the entire set of CV-Layers reported in Table 3.1. The calibration procedure presented in Section 3.1.3 has been performed for each of them (i.e., each installation is assigned different thresholds).

As it can be noted from the figure, when all the 3 CV-Layers are installed ({1,2,3} on the x-axis) the percentage increase of the interference time (with respect the original network inference) is of the same order of magnitude for SRC, MRC-16, and MRC-32. As one may easily expect, the best timing performance is achieved by SRC, with a percentage increase of 22.9%. The additional inference time introduced by kNNC is instead quite high and corresponds to a 1188% increase.

Interesting observations can be made by looking at the accuracy of the various installations and CAMs: SRC and MRC exhibit very poor detection performance if the first CV-Layer is not installed, while kNNC exhibits only slights variations of the detection performance when less CV-Layers are installed. Most interestingly, note that the detection performance may even increase when less CV-Layers are installed. For instance, these results reveal that it is not convenient to adopt kNNC with the first two CV-Layers installed, as the same detection performance can be achieved with just a 255% increase of the inference time, rather than a 1188% increase for the case in which all CV-Layers are installed.

Memory footprint. Table 3.4 reports the memory footprint of the DNN Signature of the various CAMs for each CV-Layer. As it can be noted from the table, SRC and

[3]Note that, due to implementation issues under resolution in integrating kNNC in our CUDA-based implementations of CV-Layers, the experiments for kNNC have been performed with a Python implementation of the tool presented in Section 3.1.1 that uses the Pytorch framework [131] (with GPU acceleration) and FAISS [132], which is notably the best-performing library that implements a GPU-accelerated exact nearest neighbors search.

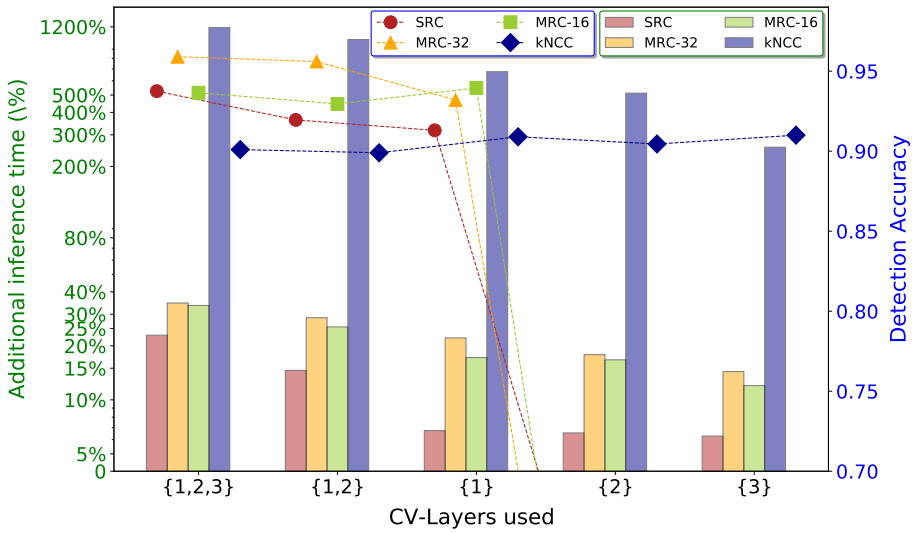


Figure 3.5: Timing performance and detection accuracy for five installations of the CV-Layers. The sets on the x-axis denote which of the CV-Layers reported in Table 3.1 are installed (numbered in the order with which they appear in the table). For instance, $\{1, 2\}$ refers to the case in which only the first two CV-Layers are installed.

	CV-Layer1	CV-Layer2	CV-Layer3
SRC	944 kB	278 kB	62,8 kB
MRC-16	8,3 MB	2,3 MB	387 kB
MRC-32	15,7 MB	4,4 MB	707 kB
NRC	483,6 kB	150,8 kB	42,8 kB
kNNC	2.6 GB	732 MB	114 MB

Table 3.4: Memory footprint of the DNN Signature (exported HDF file) of the various CAMs for each CV-Layer.

NRC have a very modicum memory footprint (in the order of 1MB in total), while kNNC is characterized by a huge memory footprint that even exceeds 3GB if all the three CV-Layers are installed. These results, together with the ones of Figure 3.5, confirm that under kNNC it does not worth to install CV-Layers between shallower layers of the network, and that SRC is a very good choice to balance performance with overheads. The results for the kNNC may vary if different (e.g., approximate) methods to implement the nearest neighbors search are adopted.

Comparison with other detection techniques. The average running times of both VG and FS were also measured. They introduce an additional latency of 138% and 237%, respectively, which correspond to the running time of multiple inferences, plus an additional overhead introduced by the used transformations. As far memory footprint is concerned, they are very lightweight approaches only when the multiple inferences they require are sequentially performed (as in the evaluated setting). Conversely, if they are performed in parallel, the overall memory footprint grows as a function of the amount of space needed for instantiating and running multiple replicas of the network model.

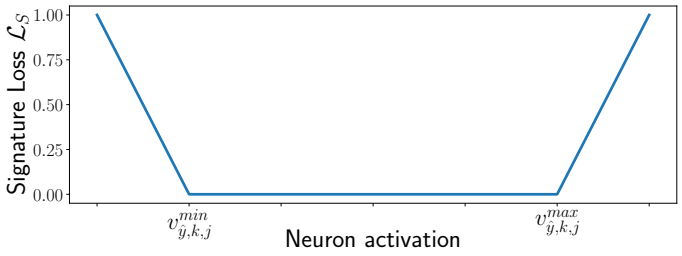


Figure 3.6: Illustration of the *Loss-Signature* for the case of a single neuron.

Adaptive attack and countermeasure

Although the presented CAMs are able to detect several kinds of unsafe inputs, ad-hoc attacks can still be devised to optimize unsafe inputs by directly exploiting the knowledge of the detection mechanism and the signatures.

Inspired by the famous gradient-based attacks (which are formally discussed in the supplementary material), we designed a white-box attack method, called *Signature-Attack*, for crafting adversarial inputs while trying to keep the activation of neurons within the range of the signatures. This is accomplished by optimizing the adversarial perturbation through two loss functions, $\mathcal{L}_{CE}$ and $\mathcal{L}_S$. The former is the common cross-entropy loss, also involved in all the other attacks tested in this work. The optimization process shall maximize such a loss to intensify the adversarial effect of the final perturbation (i.e., increase the probability of misclassifying the perturbed input). The latter is a problem-specific loss function, named *Signature-Loss*, which is conceived to return a positive cost when the activation of a neuron is outside the signature range $[v^{\min}_{\hat{y},l,j}, v^{\max}_{\hat{y},k,j}]$, where $\hat{y}$ is the label associated by the network to the input to be perturbed. Technically speaking, the Signature-Loss is implemented as $\mathcal{L}_S = \sum_{n_{l,j} \in N} \text{ReLU}(v_{l,j} - v^{\max}_{\hat{y},l,j}) + \text{ReLU}(-v_{l,j} + v^{\min}_{\hat{y},l,j})$, where $\text{ReLU}(x) = max(0, x)$. Figure 3.6 provides a sample illustration of function $\mathcal{L}_S$ in the case of a single neuron only. The Signature-Loss has to be minimized during the optimization process to reduce the chance of the adversarial example being detected by the proposed CAMs.

From a practical point of view, the optimization problem to accomplish the Signature-Attack is solved through an iterative formulation similar to [51], but based on a specific loss function $\mathcal{L}$ that takes into account both $\mathcal{L}_{CE}$ and $\mathcal{L}_S$:

$$\begin{aligned} \tilde{x}_t = \tilde{x}_{t-1} + \alpha \cdot sign(\bigtriangledown_{\tilde{x}_{t-1}} \mathcal{L}(f(\tilde{x}_{t-1}), \hat{y})), \\ s.t. \quad ||\tilde{x}_t - x|| \leq \epsilon, \end{aligned} \tag{3.4}$$

where α and ϵ denote the step size and the overall perturbation magnitude, respectively, as in [51]. The gradient of the loss function $\mathcal{L}$ is defined as

$$\bigtriangledown_x \mathcal{L} = (1 - \gamma) \cdot \frac{\bigtriangledown_x \mathcal{L}_{CE}}{|| \bigtriangledown_x \mathcal{L}_{CE}||_2} - \gamma \cdot \frac{\bigtriangledown_x \mathcal{L}_S}{|| \bigtriangledown_x \mathcal{L}_S||_2}, \tag{3.5}$$

where γ is a parameter introduced to balance the importance of $\mathcal{L}_{CE}$ and $\mathcal{L}_{\mathcal{S}}$. Since the gradients of $\mathcal{L}_{CE}$ and $\mathcal{L}_S$ may have very different scales, they are both subject to *norm-2* normalization to ensure that their effect can be properly balanced using a single parameter γ. Furthermore, it is also important to observe how the overall gradient of $\mathcal{L}$ goes in the same direction of $\bigtriangledown_x \mathcal{L}_{CE}$ and in the opposite direction of $\bigtriangledown_x \mathcal{L}_{\mathcal{S}}$. Thus, optimizing the

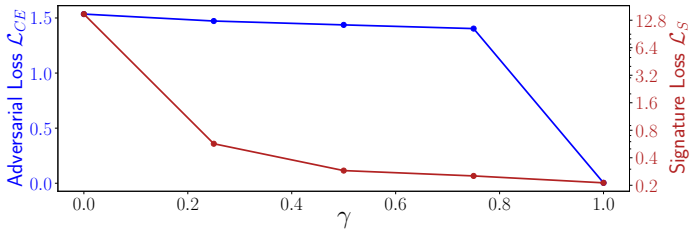


Figure 3.7: Average losses computed with inputs from the MNIST data set when attacked with the Signature-Attack, under different configurations of the control parameter γ.

	MNIST			**F-MNIST**		
γ	SRC	MRC-32	MRC-16	SRC	MRC-32	MRC-16
0.0	0.984	0.957	0.935	0.968	0.908	0.912
0.25	0.779	0.775	0.701	0.881	0.805	0.785
0.50	0.756	0.769	0.690	0.879	0.797	0.781
0.75	0.749	0.742	0.687	0.874	0.793	0.778

Table 3.5: Detection accuracy on FMNIST and MNIST against adversarial examples crafted with the Signature-Attack under four configurations.

perturbation in the gradient direction of $\mathcal{L}$ means increasing the classification loss function $\mathcal{L}_{CE}$ while reducing the Signature-Loss $\mathcal{L}_{\mathcal{S}}$.

Effects of the Signature-Attack. Multiple versions of the Signature-Attack were tested on both MNIST and FMNIST. Similar to the settings used in Table 3.2, we set α and ϵ to 0.004 and 0.1, respectively, and we evaluated five values for γ: 0.0, 0.25, 0.50, 0.75, and 1.0. The first and last values were tested for studying the behavior of the proposed attack in two limit cases (i.e., when only one of the two loss functions is considered).

Figure 3.7 reports the average classification loss $\mathcal{L}_{CE}$ and Signature-Loss $\mathcal{L}_S$ computed on the MNIST data set for the tested values of γ. Note that the classification loss (to be maximized) preserves a large value for the tested values of $\gamma < 1$, while the Signature-Loss (to be minimized) has a considerably low value for the tested values of $\gamma \geq 0.5$.

The effectiveness of the generated adversarial examples can be observed from the results reported in Table 3.5 as a function of γ (the results for $\gamma = 1.0$ are omitted since it was not possible to generate adversarial examples with this setting). As it can be noted from the table, the detection performance of the proposed CAMs is reduced, with respect to the results of Table 3.3, on both MNIST and FMNIST, demonstrating the effectiveness of the Signature-Attack with $\gamma > 0$.

Countermeasure. To counteract the Signature-Attack, we included a portion of the adversarial examples crafted with that attack to the calibration set in order to re-calibrate the thresholds. This approach helps improve the mechanism by tuning the thresholds at lower values such that also CAM-specific adversarial inputs can be detected without significantly reducing the accuracy of all the others.

To this purpose, we refined the set of inputs used for calibration (see Section 3.1.3) by adding 800 adversarial examples obtained by the Signature-Attack with $\gamma \in \{0.25, 0.75\}$.

Other 8000 adversarial examples (2000 for each $\gamma \in \{0.0, 0.25, 0.50, 0.75\}$) are appended to the test set introduced in Table 3.2. Table 3.6 shows the final results obtained after the re-calibration, i.e., by testing the detection mechanisms with the new thresholds. By comparing Table 3.6 with Table 3.5, it can be noted that the detection accuracy of the

	MNIST				F-MNIST			
Input type	SRC	MRC-32	MRC-16	kNNC	SRC	MRC-32	MRC-16	kNNC
FGSM-1	1.0	0.986	0.962	0.969	0.839	0.845	0.829	0.922
FGSM-2	1.0	0.995	0.991	0.951	0.925	0.948	0.922	0.891
PGD-1	0.991	0.953	0.887	0.969	0.849	0.842	0.804	0.939
PGD-2	1.0	0.977	0.960	0.953	0.961	0.889	0.874	0.868
BIM-1	0.993	0.910	0.953	0.923	0.839	0.847	0.792	0.919
BIM-2	0.999	0.967	0.940	0.949	0.980	0.921	0.924	0.844
CW	0.987	0.977	0.953	0.960	0.813	0.825	0.822	0.907
Out of Dis.	1.0	1.0	1.0	0.34	1.0	1.0	1.0	0.851
Patch	1.0	0.889	0.921	0.988	1.0	0.907	0.853	0.954
Signature-Attack ($\gamma = 0.0$)	0.998	0.896	0.841	0.973	0.963	0.889	0.884	0.851
Signature-Attack ($\gamma = 0.25$)	0.949	0.939	0.782	0.972	0.935	0.861	0.852	0.852
Signature-Attack ($\gamma = 0.50$)	0.939	0.831	0.767	0.971	0.933	0.861	0.852	0.853
Signature-Attack ($\gamma = 0.75$)	0.925	0.820	0.763	0.971	0.931	0.858	0.847	0.851
Safe Samples	0.986	0.938	0.89	0.957	0.890	0.886	0.854	0.916

Table 3.6: Detection accuracy after the re-calibration, performed with also unsafe inputs crafted from the proposed Signature-Attack.

proposed CAMs is improved for all the tested versions of the Signature-Attack. Also note that the re-calibration of thresholds did not significantly affect the detection performance for the other kinds of unsafe inputs, which shows similar results with respect to Table 3.3.

3.1.4 Summarizing the results

The following list summarizes the main results and insights of the previous work:

- **Coverage metric for run-time monitoring.** Inspired by classic coverage criteria, we propose a coverage-based real-time monitoring framework. The Coverage Analysis Method serves as a defense mechanism based on DNN signatures extracted in a data-driven manner.

- **On the limitations of simple structure coverage metrics.** The Coverage Analysis Methods proposed in this work extend the applicatoin scenario of traditional structural coverage criteria, demonstrating effective defense results for simple models and datasets, such as MNIST and FMNIST. However, we acknowledge the limited defense performance on medium and complex datasets and models (e.g., CIFAR10 and Imagenet). This limitation aligns closely with the constraints and concerns highlighted in the literature regarding classical structural coverage criteria, as discussed in Section 2.2.3.

  Despite this limitations, we see potential for future improvements in real-time coverage monitoring for DNNs. This can be achieved by implementing more sophisticated coverage analysis methods, which leverage the observations detailed in the following section. Additionally, we believe there are new opportunities to explore DNN applications beyond just computer vision. These could include various use cases in safety-critical systems, such as control or regression problems, including the 'remaining useful life' (RUL) problem, as discussed in Ferreira et al. (2022) [133] and Natural Language Processing.

3.2 Future Directions in Coverage Testing

The main concepts behind classic coverage criteria for DNN testing, such as neuron coverage, are based on the premise that DNNs and algorithms share similar logical structures and reasoning. However, this assumption remains somewhat vague and challenging to evaluate. Therefore, gaining a deeper understanding of the distinctions between these algorithms and DNNs could reveal fresh perspectives and insights.

3.2.1 Comparing Traditional Algorithms and DNNs

As known in software engineering, traditional algorithms are based on explicit rules set by programmers, which makes them more predictable. On the other hand, DNNs learn statistical relationships directly from data, providing a high degree of generalizability across different tasks and datasets.

Another significant difference lies in the interpretability: traditional algorithms, with their rule-based nature, are generally more straightforward to understand and interpret. In contrast, DNNs often face challenges with transparency and interpretability [134, 34]. In the following, we provide a more schematized list of differences between common software algorithms and deep neural networks.

Traditional Algorithms

1. **Rule-Based:** Traditional algorithms often follow explicit rules and instructions derived from logic and defined processes.
2. **Deterministic Output:** The output of these algorithms is usually deterministic, meaning for a given input, the output is predictable and consistent.
3. **Explicit Programming:** They require explicit programming for handling different scenarios and conditions.
4. **Limited Adaptability:** Traditional algorithms are typically less adaptable to new, unforeseen data or scenarios unless explicitly programmed for them.
5. **Transparency and Interpretability:** They are generally more transparent and easier to interpret, as each step of the algorithm is defined and understandable.

Deep Neural Networks

1. **Data-Driven:** DNNs learn from data. They identify patterns and make decisions based on the data they have been trained on, without explicit rule-based instructions.
2. **Probabilistic Outputs:** The outputs of DNNs are often probabilistic, providing predictions or classifications with associated confidence levels.
3. **Self-Learning:** DNNs are capable of learning features and patterns from data. This reduces the need for manual feature extraction and programming.
4. **High Adaptability:** They are highly adaptable to new data. With adequate training, DNNs can generalize and make accurate predictions on data they have not seen before.
5. **Interpretation Challenges:** DNNs are often considered "black boxes" because it's challenging to interpret how they arrive at a specific decision or outputs.

Based on the previous observations, we pose next discussions under two distinct directions. First, we explore the idea that more interpretable coverage criteria might emerge when the model in question closely resembles classical algorithms, for example, decision trees. Conversely, when considering a generic model, a more effective approach to implementing coverage criteria may involve leveraging statistical and data-driven aspects, inherent to DNNs.

3.2.2 Approaching DNNs from an Algorithmic Perspective

In recent years, numerous studies have explored the integration of decision-tree structures and methodologies with DNNs [135, 136, 137, 138, 139, 140, 141, 142, 143]. These studies highlight the significant advantages of incorporating decision trees into DNNs, particularly in terms of improvements in explainability, robustness, and performance [144, 145].

The integration of decision-tree-based methods with deep neural networks has led to hybrid approaches, as in [141], where the authors introduced Neural-Backed Decision Trees (NBDTs). In this architecture, the last fully connected layer of the DNN is adapted to follow a hierarchical class path. Specifically, classes are grouped into superclasses, and specific weights are assigned to predict if an input belongs to each superclass. This structure enhances the decision process granularity, offering a better approximation of a traditional decision-tree logic. In the context of coverage testing, an intriguing test objective can be devised to challenge the model by attempting to shift the predicted superclass to an incorrect one, while still maintaining accurate predictions within the correct class. This approach essentially tests the model's decision-rule behavior: if the model inaccurately identifies the superclass despite predicting the correct class, it may reveal a deviation from the intended decision-rule logic, thus exposing potential weaknesses in the model.

Conversely, another intriguing direction could involve utilizing surrogate models,based on more interpretable architectures like decision trees, for testing a reference and generic model. In this scenario, it is important to evaluate the level of transferability from samples generated from the surrogate models to the original one. This approach could potentially broaden the scope of coverage testing to include specific algorithms for generating black-box test cases, with a particular focus on the transferability.

To summarize, despite the increasing interest in leveraging decision tree models to boost the interpretability of DNNs, there is a noticeable gap in research concerning the integration of decision-rule-based methods for testing DNNs and developing coverage metrics.

3.2.3 Statistical Learning in DNN Testing

By definition, DNNs learn through identifying patterns based on data correlations. Thus, it becomes reasonable to push testing methodologies towards more statistical approaches rather than structural metrics.

In fact, a more statistical paradigm should move testing away from viewing individual neurons as well-defined decision units. Instead, it should emphasize the understanding of a set of neurons as distributions that contribute to characterize the decision-making logic of the model. This is also in accordance with known limitations of classic structural coverage techniques, which not adequately address the non-linearity and statistical nature of DNNs (see Section 2.2.3).

Out the domain of coverage testing, statistical analysis have been addressed by a many defense algorithms. For instance, across the literature some works implement defenses against adversarial examples and OOD samples based on Mahalanobis distance [146], kernel Gaussian functions [147, 148] and and support vector machines [149] for discerning safe regions within the feature space and identifying anomalous input samples. For the sake of clarity, also the defense mechanisms proposed in this thesis in Chapter 5 are based

on statistical analysis.

Also in the domain of coverage testing, there is a growing attention of the need for approaches that align more closely with the statistical nature of these networks. One of the firsts approache has been the idea of surprise coverage criteria [35], already discussed in Section 2.2. Similarly, the third Coverage Analysis Method (CAM) presented in 3.1 aims at assessing the safety of inputs by evaluating their distance from known data points, hence pretty close to the idea of surpire coverage but in the context of run-time coverage testing.

A noteworthy contribution to the field of statistical analysis for coverage testing is the research presented in [44]. This study offers an in-depth analysis that advocates for the use of layer-wise and distribution-aware criteria in DNN coverage testing. The authors start by demonstrating through preliminary investigations that existing coverage metrics fail to accurately capture the distribution of neuron activations. Building on this analysis, they propose eight requirements for correctly designing coverage criteria, which are summarized in the following:

- 1. Coverage criteria should accurately represent the continuous output of neurons, differently as simple metrics that binarize neuron activation via a threshold [33] or sign [150].
- 2-3. Coverage criteria must characterize the correlations between neurons and how the outputs of neurons in a layer are distributed, in contrast to traditional criteria that treat each neuron independently [33, 32].
- 4-5. Coverage criteria should incorporate knowledge from training data, reflecting the fact that DNNs, unlike classic algorithms, rely on training data for internal functionality. Specifically, coverage criteria should emphasize the density of neuron output distributions relative to training data, as anomalous samples are often those that deviate from the expected distribution.
- 6-8. Coverage criteria should be computationally efficient, compatible with existing frameworks, facilitate incremental updates (such as online model updates), and be user-friendly, i.e., *"ideally without the need for complex hyperparameter tuning"*.

Leveraging these insights, the authors introduced a novel coverage criterion called NeuraL coverage (NLC), which is designed to capture the distribution of training samples features within a given layer.

From a personal viewpoint, the eight requirements reported in [44] represent a significant advancement in the field of coverage criteria for deep neural networks. However, we believe further clarifications should be pointed out:

- Concerning point (4), while categorizing malicious samples as out-of-distribution in the input space is reasonable, generalizing the same for feature of a specific layer might be wrong. This is because it is possible to craft samples (e.g., adversarial examples) that could intentionally appear in-distribution within specific layers. Therefore, it is worth adding a requirement that extends the analysis to encompass all layers and their interconnections, as indicated in [28]. This may broader the scope of analysis, making the testing more challenging and computationally hard.

- An important aspect not covered in sections 6-8 is the role of coverage criteria in the test-case generation process. It is crucial to design coverage metrics that are compatible with appropriate test-case generation strategies. For example, in methods such as gradient-based generation, it becomes essential to develop coverage metrics that are differentiable, or to devise approaches for approximating them as differentiable functions.

- Explainability and Interpretability: we believe that the user-friendliness in 8 extends beyond the absence of hyperparameters tuning. A crucial factor is the development of more explainable and interpretable metrics, as preliminary investigated in [47].

3.3 Summarizing Future Directions

To conclude this chapter, we draw upon the insights gained from previous discussions and the state-of-the-art in coverage criteria. This sets the stage for an exploration of promising future directions:

- **Explainability in testing metrics:** The trustworthiness of any testing method is deeply connected to the interpretability of the criteria used. In this realm, we can distinguish two types of interpretability: (i) a structural and stand-alone interpretation of the test objective, as done in classic structural coverage criteria, which considers the activation of neurons as representative of distinct model behaviors, and (ii) the integration of coverage criteria with explainable AI methods (e.g., [47]).

  In the case of a structural interpretation, it is critical to establish a realistic relationship with the underlying assumptions. For example, assessing the existence of strong correlations between individual neurons and distinct model behaviors is important. The lack of such consideration in initial coverage works has led to the emergence of various studies that have raised concerns, as discussed in Section 2.2.3.

  Regarding the second approach, it is important to recognize that explainable AI methods themselves might not be entirely trustworthy. This presents challenges in providing comprehensive testing mechanisms.

  Nevertheless, the importance of developing coverage criteria that are interpretable and explainable from a human perspective should be a priority in future research.

- **Distribution-aware criteria**: Drawing inspiration from [44], it becomes evident that recent coverage criteria based on statistical distribution and pattern correlations, appear well-suited to yield better results in the context of DNNs. While pursuing this direction is certainly worthwhile, an important aspect to keep in mind is the explainability of the identified patterns. That is, ensuring that the patterns identified are not only statistically significant but also human-interpretable.

- **Inter-layer criteria**: In [28], Sun et al. proposed structural coverage criteria focused on the relationships between activations in adjacent layers. Drawing inspiration from this idea, future efforts should aim to propose more scalable strategies. This could involve integrating distribution-aware approaches into the inter-layer analysis framework, thereby enhancing the applicability of these criteria.

- **Expanding coverage metric analysis beyond adversarial perturbations**: Much of the current research on coverage metrics focuses on test sample generation using gradient approaches and ϵ-bounded perturbations, as done for classic adversarial attacks [51, 12]. Nevertheless, other malicious and unsafe scenarios can compromise the model robustness and performance in real-world applications, both from security aspects (e.g., patch adversarial attacks, universal adversarial attacks) and safety perspectives (out-of-distribution samples, domain shift scenarios).

- **Addressing coverage testing in continuous domain shift and OOD samples**: Designing continuous training strategies for DNNs to tackle domain shift problems presents a well-known challenge in various application scenarios, including autonomous driving and natural language processing. In this context, it is crucial to develop metrics that assess the model's ability to handle potential domain shifts and evaluate its generalizability. Conversely, it is also important to create coverage metrics that assess the model robustness against out-of-distribution samples [151].

- **Independent testing of backbone and classifier head**: Exploring testing techniques that target the backbones and classifier heads of DNNs separately could prove to be worthwhile. This direction would be strongly supported by the growing trend of using pre-trained large models, where task-specific fine-tuning typically involves adapting only the final network layers, while leaving the backbone part unchanged.

Chapter 4

Evaluating Real-World Attacks in Driving Scenarios

Real-world adversarial attacks, which involve the manipulation of physical objects, represent an alarming and important challenge that demands particular attention, as discussed from in Section 2.3.1. Due to the complexity of providing exhaustive testing techniques in the context of real-world attacks, it becomes crucial to delve deeper into the robustness of vision models by defining more effective attacks. To this end, this chapter introduces a set of attack strategies and conducts comprehensive experiments on real-world adversarial attacks in the realm of semantic segmentation (SS) models for driving scene understanding. A preliminary illustration of the effects induced by adversarial patches, crafted using the proposed methods, is shown in Figure 4.1.

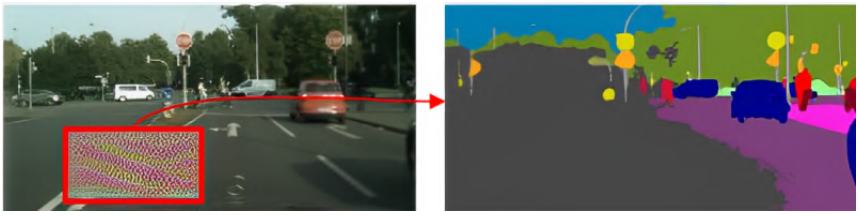


Figure 4.1: An example of the attack proposed to assess the *spatial robustness* of semantic segmentation models: the adversarial patch was crafted with the intention of maximizing misclassification throughout the entire driving scene.

4.1 Proposed attack

In Section 2.3, we pose initial definitions that serve as the foundation for crafting physical adversarial attacks aimed at deceiving a broad spectrum of computer vision tasks. The subsequent sections in this chapter extend these formulations into the context of multi-patch attacks.

All patch-based attacks considered in this analysis are generated using the pipeline illustrated in Figure 4.2. Similar to the approach described by Equation (2.3) in Section 2.3.1, the scheme shown in Figure 4.2 also represents a generalized optimization pipeline. It allows for the customization of attack parameters and settings. All these

aspects and settings of the attack are evaluated and discussed in the following paragraphs.

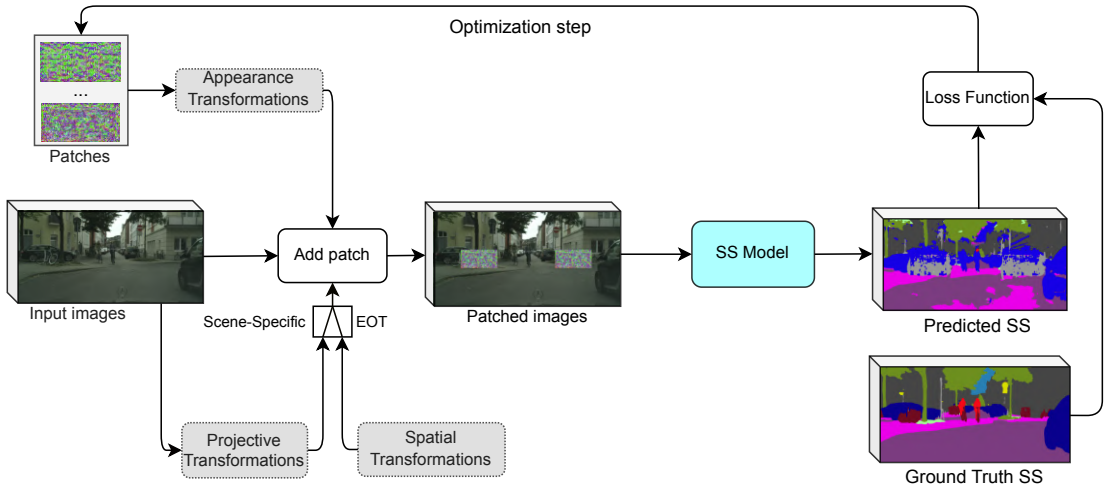


Figure 4.2: Scheme of the proposed approach for crafting both the EOT-based and the scene-specific patches.

4.1.1 Generalize to Multi-Patch Attacks

In the context of multi-patch attacks, we first need to expand our initial definition (Equation (2.3)) to consider a set of patches $\Delta = \{\delta_k : k = 1, ..., N_p\}$, where N_p is the number of patches used for the attack. In particular, the objective is to find an optimal patch set Δ^* by optimizing a certain loss function $\mathcal{L}_{Att}$ for all the patched images in expectation, according to the distribution of transformations used to apply the patch set Δ on the image set $\mathbf{X}$.

To this end, the optimization problem defined in Equation (2.3) can be extended as follows:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \ \mathbb{E}_{x \in \mathbf{X}, \zeta_a \in \mathbf{\Gamma_a}, \eta} \ \mathcal{L}_{Att}(f(\tilde{\mathbf{x}}), y_{adv}) \tag{4.1}$$

Hence, following an iterative approach, as introduced in Section 2.3.1, the attack implementation for multi-patch can be define as:

$$\delta_{k,t+1} = \text{clip}_{[0,1]} \left(\delta_{k,t} + \epsilon \cdot \sum_{x \in \mathbf{X}} \nabla_{\delta_{k,t}} \mathcal{L}_{Att}(f(\tilde{\mathbf{x}}), y_{adv}) \right), \tag{4.2}$$

where $\delta_{k,t=0}$ is the random inizialized $k-th$ patch, $\tilde{\mathbf{x}} = g(\mathbf{x}, \Delta, \Gamma_a, \eta)$, $k = \{1, ..., N_p\}$, and ϵ represents the step size. Please note that, we made revisions of the appearance-changing transformation function and the patch placement and application functions, previously defined in 2.3.2, to account for multiple patches:

- A **set of appearance-changing transformations** Γ_a: each element of Γ_a is a composition of illumination changes (brightness and contrast) and noise addition (uniform or Gaussian). This set of transformations is randomly sampled and directly

applied to the patches Δ. We defined the specific set of transformations in the experimental settings 4.2.

- A **patch placement function** η that defines which portion of the original image x is occupied by the patches. The function η can have different definitions depending on the chosen attack. Section 4.1.3 provides details of the patch placement function.
- A **patch application function** $g(\mathbf{x}, \Delta, \Gamma_a, \eta)$ that replaces the area(s) of the image $\mathbf{x}$ specified by η with transformed versions of the patches in Δ obtained by transformations randomly selected from Γ_a, hence returning the patched image $\tilde{\mathbf{x}}$. In following, the patches positions are both addressed with randomly 2D masks (digital evaluation and classic EOT optimization), but also by using 3D projective transformations (see scene specific attack discussed later).

4.1.2 Attack Objectives

The attacker objective is encoded in the performed optimization to craft adversarial patches. In particular, in case the attacker might want to maximize the prediction error of the network, regardless of the output classes (*untargeted* attack), or force the network prediction towards a specific output (*targeted* attack). In the next sections, we evaluate the models robustness against both untargeted and targeted attacks, which are formulated as follows:

Untargeted Attacks. In our formulation, the objective of an untargeted attack is then to minimize the loss function $\mathcal{L}_{adv}(f(\tilde{x}), y)$, where y is the ground-truth label. Hence, in Equation (4.2), $\mathcal{L}_{adv}(f(\tilde{x}), y) = -\mathcal{L}(f(\tilde{x}), y)$. Note that additional losses, in $\mathcal{L}_{Att}$, for physical realizability of patches are omitted for simplicity. Therefore, the minimization of $\mathcal{L}_{adv}$ implies a maximization of $\mathcal{L}(f(\tilde{x}), y)$.

Targeted Attacks. Conversely, to perform a targeted attack, the attacker has to first specify the desired prediction of the network. There are many ways to define a target for this problem (for instance, providing the label of a completely different scene [152]), but we focus to a case that is more interesting for real-world applications: forcing the network to make a specific class "disappear" from its prediction. This can be done by uniformly changing the pixels belonging to class $c_{attacked}$ into the ones of another class c_{target}, or by applying the nearest neighbor algorithm [56], as illustrated in Figure 4.3. The nearest neighbor algorithm associates to each pixel belonging to $c_{attacked}$ the class of the closest pixel of a different class.

We define our target label as $y_t = \tau(y, c_{\text{attacked}}, c_{\text{target}})$, where c_{attacked} might indicate either a specific class, or the nearest neighbor approach (specified as a pseudo-class **NN**). The targeted attack is then performed by considering, in Equation (4.2) $\mathcal{L}_{adv}(f(\tilde{x}), y_t)$.

4.1.3 Scene-Specific Attacks

The patch placement within the image might follow two different approaches: (i) randomizing the patch position, scale and rotation at each iteration (i.e., using the EOT method), or (ii) using accurate projective transformations.

Figure 4.3: Illustration of how the original label (left) can be modified to attack the class *pedestrian*: by changing *pedestrian* with *road* class (middle), or by using the nearest neighbor approach (right).

While the first is a general-purpose method for real-world adversarial patch generation (hereby, refered to as "EOT-based attack"), the latter, named *scene-specific attack*, exploits the geometrical information provided by the CARLA simulator to compute camera extrinsic and intrinsic matrices, and the pose of the attackable surface (a billboard). This enables the computation of precise camera-to-billboard 3D rototranslation (see Section 2.3.2) to warp the patch according to the point of view of the camera in each image of the dataset.

Formally, the function η in Equation (4.2) is a composition of randomized translation, rotation, and scaling for the EOT-based attack, or a precise projective transformation for the scene-specific attack. This novel method presents two main advantages: it generates stronger attacks (since the placement is more accurate and specific), and it eliminates the need to extensively randomize the patch placement, thereby saving time during the optimization process.

To apply this method, a digital representation of the target scene is required to extract geometrical data. Although CARLA can import cities via OpenStreetMaps[1], some amount of manual effort is required to model 3D meshes and include objects in the virtual world. Such objects must be carefully designed to ensure that patches will transfer well to the real world.

Motivating the scene-specific attack. The main motivations that inspire us to present and study this approach is that it helps the optimization process focuses the attack on specific reallistic placements. Although it reduces the generalization of the attack (not accounting for all the possible positions within the scenario), on the other hand it helps increases the effectivness of the attack on predetermined positions. Practically speaking, we assume that these such positions are chosen in advance by the attacker, which could depict a reasonble attack setup up for ourdoor scenarios.

In the experimental section, we provide a comparison of this method against the EOT-based attack. Furthermore, it represents a main attack block of the pipeline benchmarking tool, based on CARLA, proposed in Section 4.3.

4.1.4 Weighted Pixel-Wise Cross-Entropy Gradient

The pixel-wise cross-entropy (CE) loss, denoted by $\mathcal{L}_{CE}$, has been shown to work well for common untargeted adversarial examples by adding a perturbation r to pixels values [56] [153]. In this case, the loss is $\mathcal{L}_{adv}(f(x+r), y) = \frac{1}{|\mathcal{N}|} \cdot \sum_{i \in \mathcal{N}} \mathcal{L}_{CE}(f_i(x+r), y_i)$,

[1] https://www.openstreetmap.org/

where $f_i(x+r) \in [0,1]^{N_y}$ is the output of the model for each pixel (i.e., a probability distribution over the N_y classes), and $\mathcal{N} = \{1, ..., H \times W\}$ is the entire set of pixels in $\tilde{x}$.

However, we noticed that in the case of localized attacks, as adversarial patches, the previous formulation is not effective since some output pixels tends to produce a larger gradient magnitude so reducing the attention of attacking the others. Therefore, we propose a new formulation of the use of the loss function in the iterative optimization process of Equation 4.2.

Let $\tilde{\mathcal{N}}_k \subseteq \mathcal{N}$ represents the set of pixels belonging to a patch δ_k, and the entire set of pixels comprising all the patches defined as $\tilde{\mathcal{N}} = \cup_{k=1}^{N_p} \tilde{\mathcal{N}}_k$. The previous pixel-wise CE loss computed on the subset of pixels $\mathcal{N} \setminus \tilde{\mathcal{N}}$ can be split into two terms:

$$\mathcal{L}_M^{\tilde{x}} = \sum_{\substack{i \in \mathcal{N} \setminus \tilde{\mathcal{N}} \\ SS_i(\tilde{x}) = y_i}} \mathcal{L}_{CE}(f_i(\tilde{x}), y_i), \quad \mathcal{L}_{\overline{M}}^{\tilde{x}} = \sum_{\substack{i \in \mathcal{N} \setminus \tilde{\mathcal{N}} \\ SS_i(\tilde{x}) \neq y_i}} \mathcal{L}_{CE}(f_i(\tilde{x}), y_i) \ , \tag{4.3}$$

where $\hat{y}_i(\tilde{x})$ is the predicted class for the pixel $i \in \{1, \ldots, H \times W\}$, $\mathcal{L}_{\overline{M}}^{\tilde{x}}$ is a new loss function that accounts for the cumulative CE for the misclassified pixels, while $\mathcal{L}_M^{\tilde{x}}$ is the same but for the other pixels. Note that both $\mathcal{L}_M^{\tilde{x}}$ and $\mathcal{L}_{\overline{M}}^{\tilde{x}}$ do not include pixels of the patch ($i \in \mathcal{N} \setminus \tilde{\mathcal{N}}$) to focus the attack on image areas away from the patch.

That said, the previous loss terms allow us to balance the contributions given by correctly and incorrectly classified pixels. This is accomplished by defining the gradient of the overall adversarial loss as follows:

$$\nabla_{\delta_k} \mathcal{L}_{adv}(f(\tilde{x}), y) = \gamma \cdot \frac{\nabla_{\delta_k} \mathcal{L}_M^{\tilde{x}}}{||\nabla_{\delta_k} \mathcal{L}_M^{\tilde{x}}||_2} + (1-\gamma) \cdot \frac{\nabla_{\delta_k} \mathcal{L}_{\overline{M}}^{\tilde{x}}}{||\nabla_{\delta_k} \mathcal{L}_{\overline{M}}^{\tilde{x}})||_2} \ , \tag{4.4}$$

where $\gamma \in [0,1]$ is a balancing factor that determines whether the optimization should focus on decreasing the number of still correctly classified pixels or improving the adversarial strength for the currently misclassified pixels, and $k = \{1, \ldots, N_p\}$. In other words, γ balances the importance of $\mathcal{L}_M$ and $\mathcal{L}_{\overline{M}}$ at each iteration t of the optimization problem in Eq. (4.2).

To provide an automatic tuning of γ at each iteration, an adaptive value of $\gamma = \frac{|\Upsilon|}{|\mathcal{N} \setminus \tilde{\mathcal{N}}|}$ is proposed, where $\Upsilon = \{i \in \mathcal{N} \setminus \tilde{\mathcal{N}} | SS_i(\tilde{x}) = y_i\}$. The idea is to initially focus on boosting the number of misclassified pixels. As this number increases, the focus of the loss function gradually shifts toward improving the adversarial strength of the patch on the misclassified pixels.

Motivating the weighted pixel-wise gradient. The main rationale that expired us to split the use of the cross-entropy loss into two distinct subsets of pixels relies on the fact that the behavior of pixel gradients of δ_k during the t-iteration appear significantly different across all the image's pixels and needs a balancing to properly update the attack at each step. In particular, using the baseline approach (i.e., computing and updating the patch pixels accordingly with the gradient of the simple pixel-wise loss function), we observed that pixels close to the patch yield, on average, larger gradients. This observation is reasonable since, dealing with CNNs, a lower distance provides to the patch a higher control [154, 116]. In fact, during the initial update steps, the closer pixels are more prone

to being manipulated and so fooled. However, even if these pixels are misclassified, they pertain a large gradient within the overall computation of the update. This, in turn, makes it challenging to allocate room for fooling other pixels, which are more likely to be situated farther away from the adversarial patch and so with a lower gradient in the loss function.

Therefore, the main objective of the following formulations is to strike a balance in considering the gradients of pixels both near and far from the patch. As shown in the experimental part (Section 4.2), this strategy is more spatially effective, impacting pixels across various distances from the patch. Moreover, we also noticed that it allows a faster convergence in the attack optimization.

Notes on other robustness losses. As mentioned in Section 2.3.1, other losses are addressed in addition to the adversarial loss, such as the smoothness loss and the non-printability score. In practice, we noticed that the adversarial loss (for the untargeted attack formulation) tends to increase in norm during the optimization. This increases masks the effect of the other losses during the advancement of the optimization. To make the weighting actually effective, the gradient of each loss is computed individually, then it is normalized, and then averaged according to the weights. This total gradient is applied to advance the optimization as in Equation (4.1).

Note that overall loss used account also the smoothness loss described in Section 2.3.1. In particular, the weights of the losses are: $w_{adv} = 1, w_N = 0, w_S = 0.1$. Empirically, we noticed that the non-printability score is not strictly needed for this kind of evaluation, while the smoothness loss is crucial for transferring to the real world.

4.2 Experiments on Digital Attacks

This section presents the set of experiments carried out to evaluate the performance of the proposed attack and defense approaches[2]. First, the experimental setup is described, including the networks, the datasets, the hyperparameters, the performance metrics, and the hardware involved. Then, the results of the untargeted (single- and double-patch) and targeted attacks are presented on Cityscapes; and lastly the effect of untargeted single- and double-patch attacks are reported for the CARLA-generated images.

4.2.1 Experimental Setup

All the experiments were performed using PyTorch [131] and a set of 8 NVIDIA-A100 GPUs, while the CARLA simulator was run on a system powered by an Intel Core i7 with 12GB RAM and a GeForce GTX 1080 Ti GPU.

The optimizer was Adam [127], with learning rate empirically set to 0.5. The effect of the adversarial patches on the semantic segmentation (SS) models was evaluated using the mean Intersection-over-Union (mIoU) and mean Accuracy (mAcc) [155] on the subset of the image pixels not belonging to the patch.

Datasets Several datasets were used for the experiments. The Cityscapes dataset [156] is one of the most common dataset of driving images for semantic segmentation. It is composed of 2975 and 500 high resolution images (1024×2048) for training and validation, respectively. This dataset was used to perform single- and multi-patch attacks, both with untargeted and targeted formulation. These patches were optimized on 250 images randomly sampled from the training set, while the entire validation set was used to evaluate the effectiveness of the resulting universal patches.

Other datasets were created using the CARLA simulator by modifying the built-in Town01 map to insert some billboards that served as attackable surfaces. In particular, two different versions of three scenes (denoted by '*scene1*', '*scene2*', and '*scene3*') were considered: one for single-patch attacks (i.e., one billboard close to the road), and one for double-patch attacks (i.e., two billboards). To mimic the setting used in Cityscapes, RGB images of size 1024×2048, along with their corresponding SS tags, were collected by placing a camera on-board the ego vehicle.

For each CARLA scene (three single-patch and three multi-patch), a dataset of 150 images was collected (with no patch attached) for patch optimization. These datasets contain information about the position and orientation of both the camera and the billboard, to allow computing the roto-translations and projection matrices for different points of view in the scene. Details can be found in the supplementary material.

Once optimized, the resulting patch is imported in CARLA and applied on the target billboard. Additional 100 images per scene were collected and used for the performance evaluation of the attack. Please note that, since these datasets already include a patch, the metrics are evaluated on the entire image, and therefore produce lower mIoU and mAcc values in the random case with respect to the Cityscapes dataset.

To obtain an acceptable performance of the selected networks on such CARLA datasets, it was necessary to fine-tune them. To this purpose, a training and a validation dataset

[2]The code is available at https://github.com/retis-ai/SemSegAdvPatch

Model	mIoU / mAcc	
	cityscapes	CARLA (val - scene1 - scene2 -scene3)
ICNet	0.78 / 0.85	0.70 / 0.84 - 0.53 / 0.70 - 0.64 / 0.74 - 0.62 / 0.74
BiSeNet	0.69 / 0.78	0.47 / 0.69 - 0.47 / 0.69 - 0.61 / 0.74 - 0.47 / 0.73
DDRNet	0.78 / 0.85	0.72 / 0.88 - 0.54 / 0.74 - 0.62 / 0.76 - 0.64 / 0.78

Table 4.1: mIoU and mAcc of the tested models on Cityscapes (pre-trained) and our CARLA dataset (fine-tuned).

(denoted by '*val*') were also collected.

Finally, an additional custom dataset of real-world images was collected to optimize the real-world patch that was eventually printed. This dataset is detailed in Section 4.2.6.

Models Three real-time SS models suited for autonomous driving applications were used to evaluate the attacks addressed, namely DDRNet [3], BiSeNet [2], and ICNet [157]. These models were tested in two settings: one with Cityscapes, using the pre-trained weights provided by the authors, and one on CARLA, where the models were refined with our fine-tuning procedure (further details are reported in the supplementary material of [24]). Table 4.1 summarizes the performance of these models.

4.2.2 Effects of Untargeted Attacks on Cityscapes

The untargeted attack is evaluated on the Cityscapes dataset using both single- and double-patch attacks. Three patch sizes were used to test the effect of small, medium, and large patches. In particular, the sizes considered for the single-patch attack are 150×300, 200×400 and 300×600 pixels, while, for the double-patch attacks, we used 106×212, 141×282, and 212×424 pixels. This setting allowed to make a fair comparison between the double and single-patch formulations, since the overall area covered in each type of attack is roughly the same.

Transformation Γ_a includes only Gaussian noise with standard deviation 5% of the image range, whereas the patch placement function η includes random scaling ($80\% - 120\%$ of the initial patch size) and random translation defined as follows: if (c_x, c_y) is the center of the image, the position of the patch is randomized within the range $(c_x \pm \tilde{r} \cdot \tilde{W}/2\ ,\ c_y \pm \tilde{r} \cdot \tilde{H}/2)$, where $\tilde{r} \in [0, 1]$ is random variable with a uniform distribution. The translation range was kept limited, rather than considering the full image space, to ensure better optimization stability and faster convergence. The same transformation settings were used for the double-patch attack, with the only difference that the center (c_x, c_y) corresponding to each of the two patches are the center of the left and right halves of the image. The patches were optimized over 200 epochs.

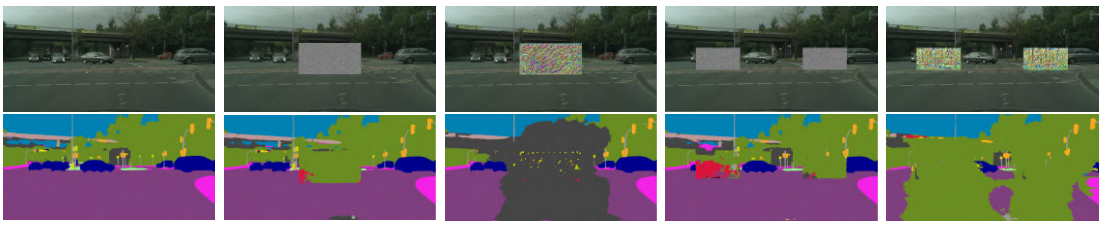
As shown in Table 4.2, the double-patch formulation achieves, in general, higher attack performance with respect to the single patch version. In particular, for DDRNet and ICNet, the double-patch attack gets a lower mIoU in all the tested sizes. Different considerations arise for BiSeNet, where patches with small and medium sizes achieve better results on the single-patch attack. These results suggest that, for some models, the per-patch size could be more relevant than the number of patches involved. Further results

Model	mIoU - mAcc (rand / EOT)					
	150 × 300		200 × 400		300 × 600	
	double - 106 × 212		double - 141 × 282		double - 212 × 424	
ICNet	0.76 / 0.62	0.84 / 0.72	0.75 / 0.55	0.83 / 0.66	0.75 / 0.43	0.82 / 0.48
	0.74 / 0.49	0.83 / 0.57	0.72 / 0.38	0.81 / 0.47	0.68 / 0.21	0.77 / 0.30
BiSeNet	0.67 / 0.48	0.76 / 0.63	0.67 / 0.32	0.75 / 0.46	0.65 / 0.22	0.74 / 0.34
	0.64 / 0.50	0.72 / 0.64	0.63 / 0.45	0.71 / 0.57	0.60 / 0.20	0.68 / 0.30
DDRNet	0.77 / 0.70	0.84 / 0.79	0.77 / 0.63	0.84 / 0.74	0.76 / 0.53	0.83 / 0.60
	0.75 / 0.60	0.82 / 0.71	0.73 / 0.55	0.81/ 0.62	0.72 / 0.35	0.79 / 0.43

Table 4.2: Adversarial patch results in mIoU and mAcc (calculated out of patches areas) extracted from the validation set of Cityscapes. Each cell reports the mIoU obtained from a random patch (no optimization) and the EOT optimization. The white rows refer to single patch attacks, while the grey rows refer to double patches having the same areas of the single configuration.

are provided in the supplementary material of [24][3].

Figure 4.4 shows the adversarial effect produced by the proposed attacks, illustrated for the single (300 × 600) and double-patch formulation (212 × 424).



(a) (b) (c) (d) (e)

Figure 4.4: Images of the Cityscapes validation set and their semantic segmentations obtained from DDRNet with (a) no patch, (b) random patch (300 × 600), (c) adversarial patch, (d) double random patches (212 × 424) and (e) double adversarial patches.

4.2.3 Effects of Targeted Attacks on Cityscapes

While the objective of an untargeted attack is to induce to maximize misclassifications in the network predictions, regardless of the classes that are predicted, a targeted attack must follow a much more constrained optimization process.

In particular, with a targeted attack, by pushing the network prediction towards the target label y_t, we are asking the patch not only to change the prediction on the pixels originally belonging to the class c_{attacked} to the class c_{target} but also to keep all the other pixels untouched (those corresponding to not attacked classes). This resulted to be a very difficult task, especially when considering image-agnostic attacks: the patch should generalize the targeted attack for an entire set of images that might differ largely on the distribution of the attacked class (e.g., pedestrians have different location and appearances in different images).

[3]We omitted a complete illustration of all the results for enhancing the discussion and readability of the thesis.

Previous work [56] on targeted localized adversarial perturbations for SS models only deal with *image-specific* (i.e., non image-agnostic) attacks: this means that the perturbations are tested on the same single image they are optimized on.

We provide a similar analysis to understand whether there are classes that can be easily attacked with a real-world attack, further validating our loss function formulation when attacking particular classes of interest for the real-world case.

Table 4.3a reports the effects of some image-specific patch attacks of interest for the real-world case. We decided to perform a double-patch attack with a total area of 600×300, since it is the strongest attack we tested. In fact, we empirically found that it is very difficult to carry out these targeted attacks with a single patch. The table shows the IoU of the attacked class at the beginning and at the end of the optimization process. IoU values are averaged on 100 different attacks. Different networks show different strengths and weaknesses: in particular, DDRNet and BiSeNet are easily attackable with a **road→sidewalk** attack, while ICNet is not. Conversely, ICNet suffers the **sidewalk→NN** attack, while BiSeNet does not. This table also gives us an indication of whether it might be possible to perform universal attacks. In fact, if an average image-specific attack is not completely successful (i.e., IoU ≈ 0 on the attacked class), there is no hope that the attack will extend to an entire set of images.

Attack	ICNet	BiSeNet	DDRNet
road → sidewalk	0.96 / 0.81	0.96 / 0.07	0.97 / 0.00
sidewalk → NN	0.77 / 0.12	0.78 / 0.47	0.84 / 0.08
pedestrian → NN	0.48 / 0.19	0.56 / 0.30	0.65 / 0.26
car → NN	0.86 / 0.45	0.86 / 0.44	0.90 / 0.30

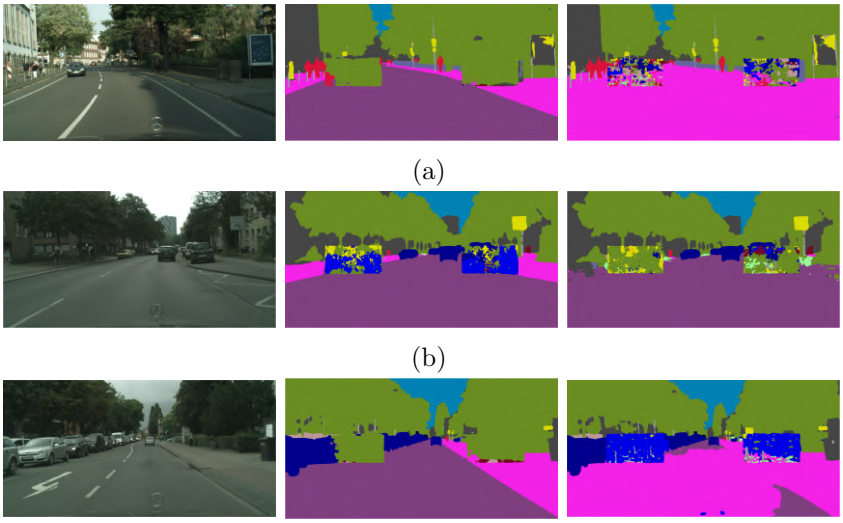
(a) Image-specific attacks

Attack	ICNet	BiSeNet	DDRNet
road → sidewalk	0.98 / 0.94	0.98 / 0.37	0.98 / 0.12
sidewalk → NN	0.84 / 0.71	0.85 / 0.83	0.90 / 0.52
pedestrian → NN	0.76 / 0.73	0.85 / 0.70	0.89 / 0.88

(b) Universal targeted attacks

Table 4.3: Effectiveness of (a) some image-specific attacks and (b) selected universal targeted attacks for different networks. Each cell reports the IoU of the attacked class at the start and at the end of the optimization among 100 samples of the Cityscapes validation set.

We empirically assessed this argument by performing universal targeted attacks for the attacks defined above. Table 4.3b presents the IoU of the attacked class for each attack and each network. Please note that these values are only indicative of the performance of the attack: in fact, since we are evaluating image-agnostic attacks, we are applying the same patch to each image of the validation set, which surely differ for the distribution of the pixels belonging to each class. These experiments lead to the conclusion that each network shows robustness for different classes. The only universal targeted attack with good performance is **road → sidewalk**. The others tested attacks resulted less effective. Figure 4.5 illustrates some of the attacks performed during the evaluation.



(a) (b) (c)

Figure 4.5: Effect of targeted patches: (a) an image-specific **road** → **sidewalk** attack on DDRNet, (b) an image-specific **sidewalk** → **NN** attack on ICNet, (c) a universal **road** → **sidewalk** attack on DDRNet. First column is the original image, while middle and right columns are the predictions with random and adversarial patches, respectively.

4.2.4 Effects of Untargeted Attacks on CARLA

The scene-specific attack is evaluated against the EOT-based attack using single- and double-patch attacks in CARLA. As explained in Section 4.1.3, additional information on the relative pose of the camera and the billboards are extracted from CARLA together with the corresponding images and then used to apply accurate patch warping transformations during the optimization process, to account for different points of view of the same urban scene.

We decided to use patches of 150×300 pixels (corresponding to a real-world dimension of 3.75m×7.5m). For all the following experiments, Γ_a includes contrast and brightness changes (both randomized within ±10% of the image range) and Gaussian noise (with standard deviation equal to 10% of the image range).

Since this paper investigates the effect of real-world adversarial patches, the mIoU and mAcc scores are evaluated on an additional dataset for each tested network: this time, the patch is not digitally projected, but rather is imported in CARLA and applied to a virtual billboard as a *decal* object. Hence, the resulting dataset includes images of the billboards with an adversarial patch already applied and rendered with the same level of graphic detail. This allows simulating real-world adversarial examples in CARLA.

The third column of Table 4.1 reports the performance of the tested networks on the considered scenes (with no patch), while Table 4.4 reports the corresponding adversarial effect in terms of mIoU and mAcc scores for both single- and double-patch attacks. Note that, for each setting reported in Table 4.4, the scene-specific attack achieves better results than the EOT formulation.

For single-patch attacks, the performance of the two approaches is comparable and their adversarial effect is marginal. While for the Cityscapes dataset we considered double-

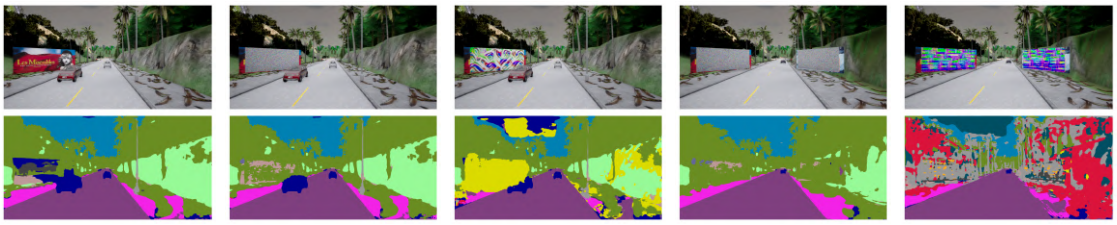
patch and single-patch attacks with the same total patch area, in this case, double-patch attacks use total areas two times larger than those for single-patch attacks (i.e., we used two patches with the same size 3.75m×7.5m). This choice was due to the poor performance of single-patch attacks.

In the double-patch case, the performance of the attack largely improves, as well as the difference between the performance of the scene-specific and the EOT-based attacks.

It is worth observing that the performance of the tested networks on CARLA scenes is worse than the one related to Cityscapes: this is partly due to the differences in evaluating the performance for Cityscapes and for the CARLA-generated datasets. For Cityscapes, the area corresponding to the patch is not considered during the evaluation: this helps ignore parts of the image that are wrongly predicted as occluded by the patch. Conversely, for CARLA images, we decided to take into account also the patch area, since it is already present in the 3D virtual scene of CARLA. Figure 4.6 shows the effect of some representative attacks on CARLA images.

Model	mIoU | mAcc (rand / EOT / scene-specific)					
	Scene1		Scene2		Scene3	
ICNet	0.51 / 0.49 / 0.48	0.60 / 0.56 / 0.54	0.64 / 0.61 / 0.61	0.74 / 0.73 / 0.73	0.63 / 0.59 / 0.59	0.76 / 0.73 / 0.74
	0.43 / 0.43 / 0.39	0.56 / 0.55 / 0.50	0.58 / 0.57 / 0.55	0.66 / 0.67 / 0.67	0.64 / 0.61 / 0.54	0.76 / 0.73 / 0.68
BiSeNet	0.44 / 0.36 / 0.31	0.63 / 0.55 / 0.49	0.60 / 0.58 / 0.58	0.76 / 0.74 / 0.74	0.47 / 0.46 / 0.45	0.74 / 0.73 / 0.73
	0.39 / 0.37 / 0.23	0.88 / 0.54 / 0.53	0.55 / 0.54 / 0.53	0.75 / 0.72 / 0.70	0.44 / 0.43 / 0.42	0.74 / 0.67 / 0.62
DDRNet	0.51 / 0.46 / 0.46	0.70 / 0.69 / 0.69	0.62 / 0.52 / 0.49	0.76 / 0.71 / 0.66	0.65 / 0.58 / 0.59	0.78 / 0.76 / 0.76
	0.48 / 0.48 / 0.39	0.67 / 0.66 / 0.66	0.58 / 0.57 / 0.47	0.75 / 0.74 / 0.69	0.66 / 0.66 / 0.58	0.79 / 0.78 / 0.76

Table 4.4: Adversarial patch results on the three scene CARLA datasets. The Table reports the mIoU and mAcc obtained with random, EOT-based and scene-specific patches. For each network, the first row indicates results for single-patch attacks, whereas the second row reports results for double-patch attacks.



(a) (b) (c) (d) (e)

Figure 4.6: Semantic segmentations obtained with BiSeNet on CARLA scene-1 with (a) no patch, (b) single random patch, (c) single scene-specific patch, (d) double random patches, and (e) double scene-specific patches.

4.2.5 Evaluating the proposed loss function and parameters

The proposed loss function formulation presented in Section 4.1.4 was evaluated against the standard cross-entropy loss for several values of γ and for the different attacks.

Figure 4.7 shows the evolution of the mIoU score during the optimization process for the untargeted EOT-based attack on Cityscapes and the scene-specific attack on CARLA.

In both cases, for each tested value of γ, our loss formulation outperforms the standard cross-entropy, both in terms of attack performance and convergence rate.

Figure 4.8 shows the optimization process of the image-specific targeted attack **pedestrian→NN** (the values are averaged on 100 different attacks). The plot at the top shows the IoU of the attacked class against the original labels, while the one at the bottom shows the IoU of the class **road** against the target labels. As it is shown in the latter plot, the IoU of the class that is not under attack grows because the target label includes pixels of the class road (or other classes) that replaced the attacked class, and the prediction is forced to mimic the target label. Please note that, in the targeted case, the tested γ values are the ones that perform best, i.e., those < 0.5. This is opposed to the untargeted case, since the attack is formulated to minimize the loss, and not to maximize it. Also in the targeted case, our loss formulation outperforms the standard cross-entropy loss.

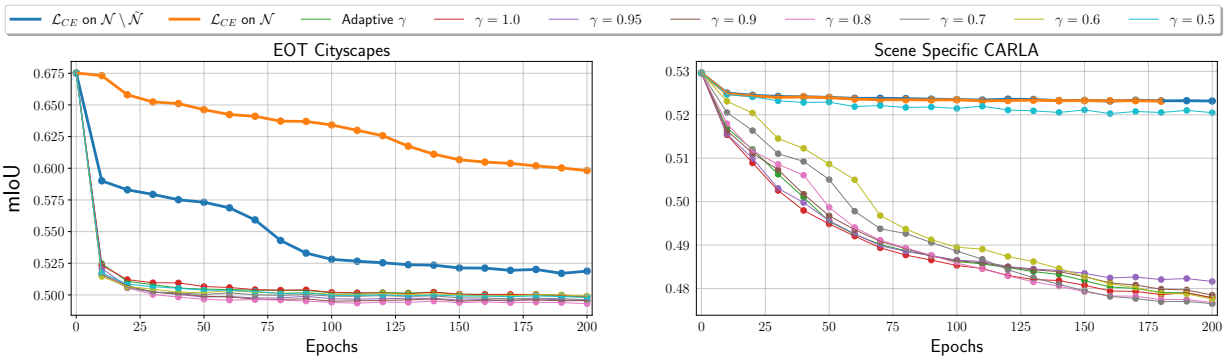


Figure 4.7: Comparison of adversarial patch optimizations (200×400) on ICNet and Cityscapes using different loss functions: two versions of the standard pixel-wise cross-entropy and our formulation with multiple values of γ. $\mathcal{L}_{CE}$ on $\mathcal{N}$ is the original version used by [56], while $\mathcal{L}_{CE}$ on $\mathcal{N} \setminus \tilde{\mathcal{N}}$ is an improved version based on the rationale presented in Section 4.1.4.

4.2.6 Real-World Evaluation

To prove the effectiveness of the attack pipeline proposed in semantic segmentation models, we used a custom dataset to craft an adversarial real-world patch using the EOT-based formulation and fixing $\gamma = 1.0$ for the proposed loss function optimization in Eq. (4.2). The dataset is composed of 1000 images that were collected by mounting an action camera on the dashboard of a real car, using a setup similar to the one of the Cityscapes dataset, and then driving the car through the streets of our city. The patch was optimized for 200 epochs on the original pre-trained version of ICNet (since it showed good performance also on our personal real-world dataset). Figure 5.5a shows a sequence of frames recorded while moving in the direction of the adversarial patch, printed as a $1m \times 2m$ poster. The illustrations remark how the optimized patch can alter a significant area of the predicted SS when it appears close to the camera. In fact, we can also see as the attack performance increases as we move close to the patch.

It is worth remarking that testing adversarial patches for autonomous driving in the real world poses a series of difficulties that heavily limited the tests. First, it is not easy to find a proper urban corner that shows good performance and is not crowded with moving

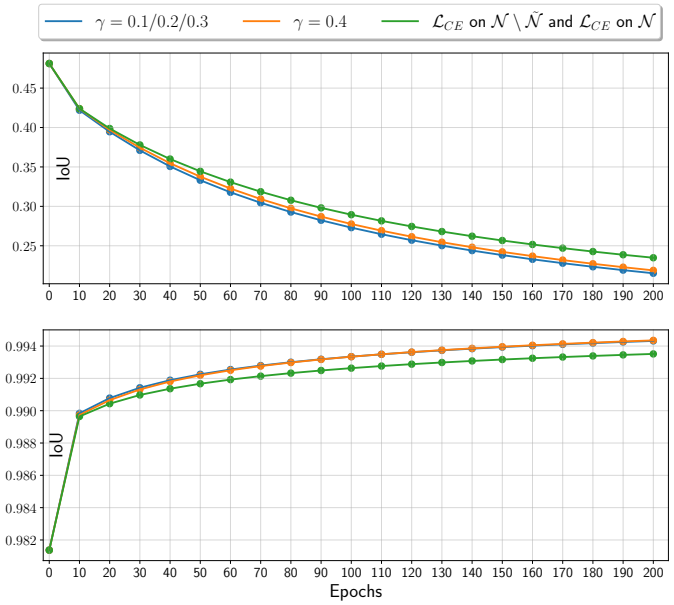
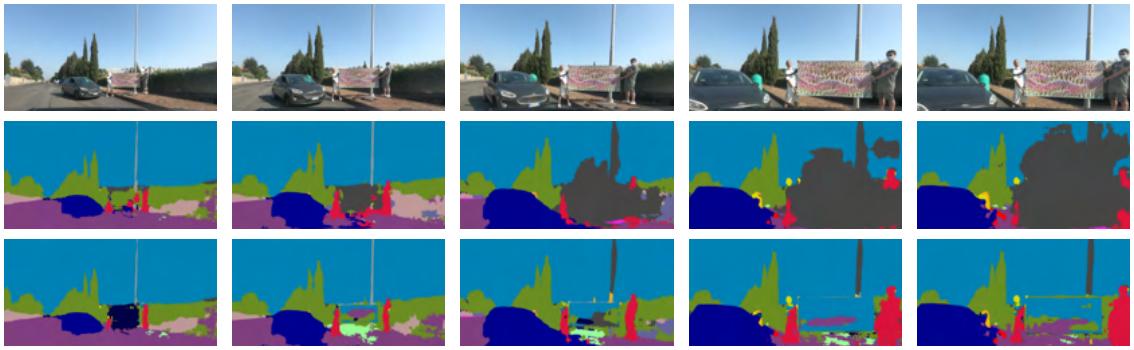


Figure 4.8: Summary of the optimization process for an image-specific **pedestrian→NN** attack. The IoU values at each epoch are averaged over 100 different attacks. Top image shows the IoU of the class **pedestrian**, whereas bottom image shows the IoU of the class **road**. For each value of γ tested, our loss formulation outperforms the standard cross-entropy loss.



(a)

Figure 4.9: Real-world evaluation on ICNet of the attack method and the defense algorithm. In inset (a), the first row contains the original images with the printed adversarial patch, while the second row contains their obtained semantic segmentations. The third row instead shows the outcomes obtained by masking the patch areas of the previous inputs with white pixels.

vehicles (which might be dangerous). Second, the patch must be printed in the highest resolution possible on a large rigid surface, which might get expensive.

4.3 A systematic Tool for Assessing Robustness

Expanding upon our prior results on the CARLA simulator, which allows us to craft precise patches using 3D data information, we have extended the use of CARLA in a more systematic manner. The objective of the study conducted in this section was to create a comprehensive tool for systematically assessing the robustness of computer vision models in driving scenarios.

The core concept behind CARLA-GeAR (**Ge**neration of datasets for **A**dversarial **R**obustness evaluation with CARLA) is straightforward: Given an urban scenario available from the CARLA tool, where an adversarial patch might be placed (for instance, on a billboard), CARLA-GeAR first integrates the patch onto the selected surface. Subsequently, it iteratively positions the vehicle and its camera around the scene, capturing high-definition RGB images, ground-truth labels, and additional information regarding camera intrinsic and extrinsic matrices, as well as the billboard's pose in the scene.

In cases where the patch is not readily available, the tool provides an automatic pipeline for generating the attack scenario. This is achieved by initially gathering a dataset around a specific billboard with no patch attached and then executing the scene-specific optimization algorithm discussed in Section 4.1.3.

The resulting patched dataset can be employed to assess the performance of different defense mechanisms or to evaluate the adversarial robustness of a target Convolutional Neural Network (CNN). Multiple datasets, encompassing diverse attack scenarios, can be collected and utilized for a more systematic and comprehensive evaluations of several vision tasks. Figure 4.10 illustrates some representative driving scenarios considered in CARLA-GeAR.

4.3.1 On the Need of Benchmarking Real-World Attacks

Addressing the vast literature on real-world adversarial attacks, we acknowledged that no much attention was devoted in the literature to the development of datasets or benchmarks for a systematic evaluation of the adversarial robustness of CNNs against physical adversarial attacks, nor for assessing the performance of adversarial defense methods in a unified framework, in particular concerning autonomous driving scenarios. This might be due to the practical difficulties in the autonomous driving domain, which could result poorly customizable and/or dangerous. However, the rise of high-definition simulators is paving the way for fully-controllable, photo-realistic driving scenarios that allow for an extensive evaluation of potentially dangerous situations.

Looking at the literature, only a few works proposed datasets and benchmarks for a systematic evaluation of the real-world robustness of models. In fact, most of them are focused on digital adversarial examples [158], [159], [160], [161] and none of them considers the autonomous driving context. In [162], the authors systematically evaluate the effectiveness of adversarial patches. Other works explore other kinds of attacks against autonomous driving systems [61, 163]. A work close to CARLA-GeAR is DeepBillboard [164], which generates adversarial patches to be attached on billboards. However, DeepBillboard attacks end-to-end autonomous driving models, whereas CARLA-GeAR allows addressing multiple perception tasks individually. Furthermore, being built on a simulator, CARLA-GeAR allows for fully-controllable scenarios, which is a crucial feature for evaluating of the robustness of a system.

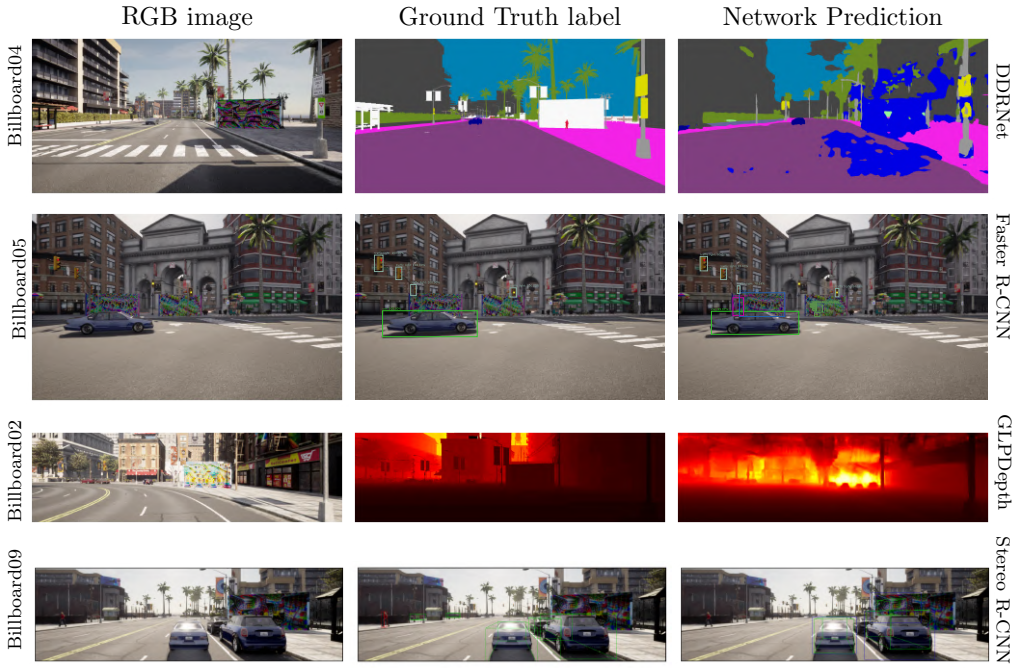


Figure 4.10: Examples of different attack scenarios and tasks. Best viewed in digital, zooming in. DDRNet for semantic segmentation on the billboard04 scenario, Faster R-CNN for 2d object detection on billboard05, GLPDepth for monocular depth estimation on billboard02, and Stereo R-CNN for stereo 3d object detection on billboard09. First column shows the RGB image, second column shows the ground truth generated by CARLA, and the third column shows the prediction of the corresponding CNN.

To the best of our records, APRICOT [58] is the only publicly available dataset that includes physical-world adversarial patches. However, it can only be used to test 2D object detection models on COCO-like images. Furthermore, the dataset does not include non-adversarial images, the patches are not always effective due to patch bending or extreme view angles, and it does not address specifically driving scenarios.

This lack in the literature motivates us for the design of datasets and benchmarks for the evaluation of the adversarial robustness of CNNs and defense methods in the physical world. To this end, we present CARLA-GEAR (**Ge**neration of datasets for **A**dversarial **R**obustness evaluation with CARLA), a tool built on top of the Python API of the CARLA simulator [74], which allows constructing photo-realistic synthetic datasets for four vision tasks, namely semantic segmentation, 2D and 3D stereo object detection, and monocular depth estimation.

Attack situations This work considers real-world adversarial attacks based on patches. The tool considers two different types of situations: (i) a patch (or two in the case of multi-patch attacks) on a billboard on the side of the road, and (ii) a patch on the back of a truck placed in front of the camera. Each billboard situation is specified in a `yml` file as the fixed position of the billboard in the map. The same file specifies the spawn positioning limits of the ego vehicle and the NPCs relative to the billboard. The spawning is then randomized within these limits, as illustrated in Figure 4.11. In this way, it is possible to

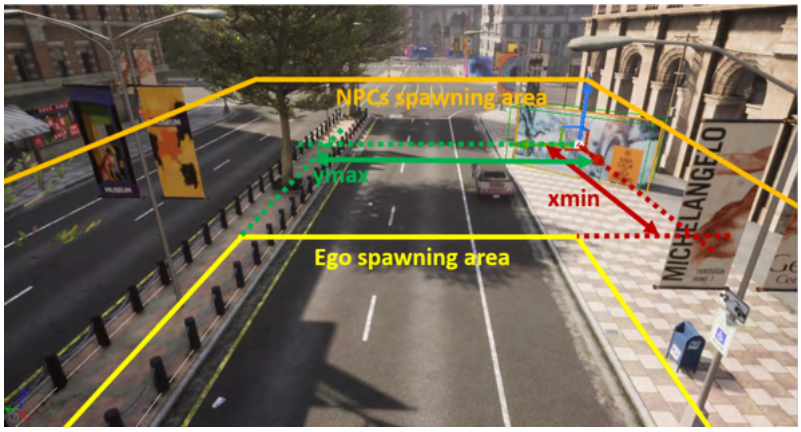


Figure 4.11: An example of spawn area for ego vehicle and NPCs. All the distances are relative to the billboard.

generate different views of the same potentially dangerous scene.

While the billboards have fixed positions, the truck is randomly spawned in the map, and the ego vehicle is spawned behind it at different randomized distances.

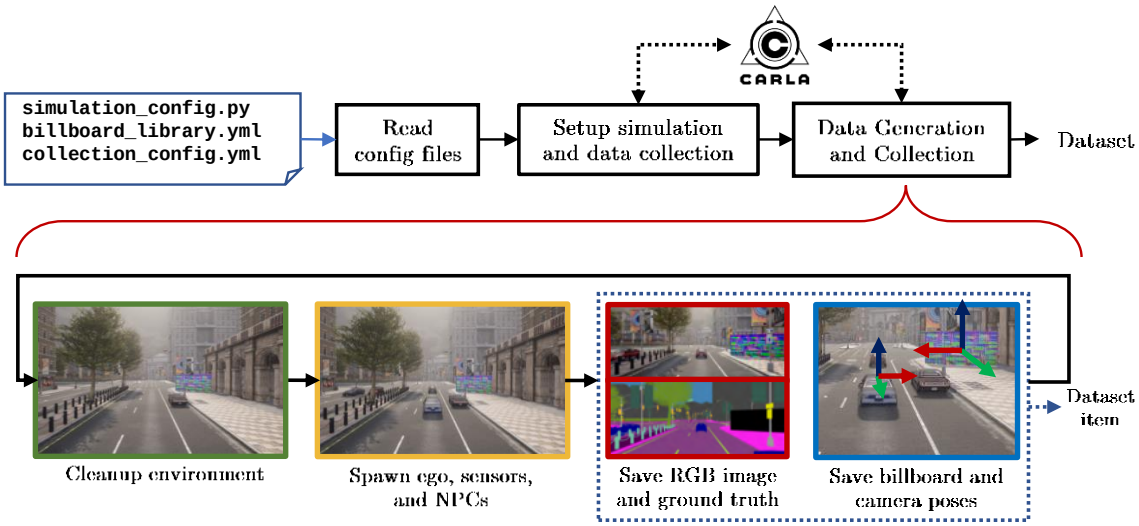


Figure 4.12: Complete generation flow (top) and detail of the data generation and collection phase.

Configuration files Three configuration files are required: (i) `collection_config.yml`, which is the main collection configuration file, (ii) the `billboard_config.yml` file, which includes information about the billboard positioning and the limits for the spawning area of the ego and NPC vehicles, and (iii) the `simulation_config.py` file, which sets the seed and other simulation parameters (number of NPCs in a scene and other Unreal utilities).

The pipeline described in this section generates single dataset splits (train, validation,

test, and so on) starting from the main configuration file `collection_config.yml`, that specifies:

- The target task (one between semantic segmentation, 2D object detection, stereo 3D object detection, monocular depth estimation). This directly specifies the folder structure and the ground truth annotation types required;
- The CARLA town. Any CARLA town would work, but the one used throughout this thesis is Town10HD, since it is the town with the most photo-realistic meshes in CARLA;
- The dataset root folder and split;
- The desired scene. As explained in Section 4.3.1, this configures the adversarial surface positioning, the ego vehicle, and NPCs spawning.
- The desired patch to be uploaded and additional info on it. The patch path, if specified, is used to render the patch on the surface selected. If it is not specified, no patch is rendered. This possibility is better detailed in Section 4.3.1.

The aforementioned `billboard_config.yml` and `simulation_config.py` configuration files are then read to define the attack situation and simulation settings.

Data generation algorithm Figure 4.12 shows the data generation and collection pipeline, which is discussed next.

1. **Read config files** is a set of parsers that extract information from the configuration files presented in Section 4.3.1.
2. **Setup simulation and data collection** sets up the client-server communication with CARLA through its Python API and writes a few necessary settings. It then loads the specified town and spawns the selected billboards. It also sets up the camera types required for the specific task, the seed for repeatability, and generates the folder tree to properly store images and annotations.
3. **Data Generation and Collection** is the core of the pipeline illustrated in Figure 4.12. The dataset is constructed by iterating three steps: (i) cleanup of any additional vehicle/pedestrian, (ii) spawn of the ego vehicle, its sensors, and randomized NPCs following the spawning limits of the specific situation, (iii) save the RGB image, the ground truth (this task-specific operation is detailed in Section 4.3.1), the billboard, and the camera poses. Additional details are provided in the supplementary material of [165].

Dataset structure and annotations RGB images are always saved as uint8 with different resolutions, while each task has different ground-truth annotations and folder structure: semantic segmentation datasets follow the CityScapes [156] format, 2D object detection uses the COCO format [166], stereo 3D object detection the Kitti Stereo Object Detection format[167], and monocular depth estimation the Kitti Depth format. This

choice is motivated by the fact that the same original metrics, evaluation procedures and deep learning framework data loaders can be used to evaluate these custom datasets.

CARLA has special semantic segmentation and depth camera sensors that allow immediate ground-truth computation. It can also compute all the 3D bounding boxes in the simulated world. However, the 2D and 3D bounding boxes that are visible in each scene must be computed heuristically (details are reported in the supplementary material of [165]).

Additionally, CARLA-GeAR collects all the billboard positions together with the camera extrinsic and intrisic matrices. This allows knowing the precise pose of the attackable surface in the scene, which is crucial to make an accurate digital patch placement (see details in Section 4.3.1) and obtain the ground truth for evaluating the accuracy of masking defense methods against adversarial patches.

Patch generation As described in Section 4.3.1, the `collection_config.yml` file specifies the path of the patch that must be uploaded and rendered on the selected billboard. If the file path is not defined, the tool does not render anything on the billboard. This is useful to have an additional test split to check the performance of a CNN or a defense method in a non-adversarial case. Furthermore, these non-adversarial dataset splits can be used to train adversarial patches for that specific situation. We follow the *scene-specific* attack method proposed in Section 4.1.3, which requires camera and billboard poses to accurately reproject the digital patch to maintain differentiability and the possibility to perform white-box attacks.

4.3.2 Testing The Tool

This section presents an instance of how the CARLA-GeAR toolbox might be used to benchmark the robustness of CNNs and the performance of a set of defense and detection algorithms. In Section 4.3.2, preliminary tests were performed to set the simulation configuration used for the generation of datasets. Furthermore, in Chapter 5 the datasets generated are used to compare a selection of state-of-the-art defenses and detection methods for each task. The experimental setup is briefly listed in Section 4.3.2 and better detailed in the supplementary material, together with additional experiments and illustrations from the datasets.

CARLA-GeAR's settings

The CARLA simulator version 0.9.13 was used (UnrealEngine 4.26 on Ubuntu 18.04) on a machine equipped with an Intel i7-4790K CPU @ 4.00GHz × 8 and an NVidia GTX 1080Ti GPU. The adversarial patch optimizations were performed on an NVidia Tesla A100 GPU using PyTorch.

The CNNs used were DDRNet23Slim [3] and BiSeNetXception39 [2] for semantic segmentation, Faster R-CNN [168] and RetinaNet [169] for 2D object detection, GLPDepth [170] and AdaBins [171] for monocular depth estimation and Stereo R-CNN [172] for stereo 3D object detection.

The comparison metrics used are (i) mIoU for semantic segmentation, (ii) the COCO mAP for 2D object detection, (iii) root mean square error (RMSE) in meters for monocular depth estimation, and (iv) kitti AP for "moderate" label difficulty for stereo 3D object detection.

Life-cycle of a patch To evaluate the effects of real-world adversarial patch attacks in CARLA it is necessary to follow a few steps. Given a certain task and an attack scenario, it is possible to collect a training dataset split to digitally optimize the patch to attack a specific CNN using the *scene-specific* optimization algorithm. The same training set can be used to craft patches for different networks but for the same task. Once the patch is optimized, it can be imported in CARLA and applied to the billboard to generate the test set. In this way, it is rendered realistically as the other objects in the scene and it constitutes a "virtual" real-world adversarial object. Different patches (for different networks) are used to generate different test sets. However, each test set shares the same seed. This allows generating datasets with exactly the same images, except for the selected patch, hence providing a more fair comparison. Furthermore, each adversarial patch is tested against a control test set that includes a random patch or no patch.

The datasets used in the following evaluation include 10 attack situations, namely 9 billboard-based attacks (billboard01 to billboard09, of which billboard05 to billboard07 are double billboard for double patch attacks) and 1 truck-based attack.

Each of these dataset includes images collected around the attack surface, which are arranged in the following splits:

- A train split (100 images) with no patch attached. This can be used to optimize patches for the specific situation.
- A val split (50 images) with no patch attached. It can be used as a validation split during the patch optimization.
- A test_net split (50 images), which includes a patch crafted specifically for the model indicated by "net" (e.g., ddrnet, bisenet, adabins, and so on).
- A test_random split (50 images), which includes a random patch.
- A test_nopatch split (50 images), which includes no patch on the attack surface.

Please note that each test split has the same seed. Therefore, the only difference between the three test split images described above is the patch applied on top of the attack surface, as showed in Figure 4.13. This is useful for a fully controllable and fair comparison of the adversarial effect of the patch, as well as the evaluation of the adversarial robustness of different defense methods and models.

Furthermore, for the conducted tests we provide a `no_billboard` dataset, which includes only a train split (100 images) with no billboard spawned. The ego vehicle and its camera are moved randomly around the city. This dataset can be used as a reference for the distribution of the scenes produced by CARLA. In fact, it was used to update the batch normalization parameters of DDRNet, BiseNet and AdaBins to improve the performance and handle an unavoidable domain shift between CARLA scenes and realistic images from Cityscapes. It was also used to train the defense methods that required a set of not attacked samples. (see Section 5.3).

(a) (b) (c)

Figure 4.13: Comparison of RGB images (top) and network prediction (AdaBins, bottom) for a sample of the dataset billboard02: (a) test_nopatch, (b) test_random, (c) test_adabins

On the flexibility of CARLA-GeAR. The proposed framework is aimed at systematically evaluating the adversarial robustness in several attack situations. However, the experimental results presented in this section are just a subset of those that can be obtained with the proposed tool. More specifically, we tested untargeted attacks that produce highly effective patches based on robust and flashy adversarial input patterns.Different threat models might be considered during the patch generation phase to obtain more inconspicuous patterns (such as the one in [61]) or different attack settings (illustrated in Section 2.3.1).

Ablation studies

This section reports the results of ablation studies performed in the early stages of the experimentation to set the parameters used for the data generation. The objective was to find a good trade-off between the CNN performance and the attack effectiveness while maintaining reasonable computation times. We decided to use the same billboard configuration that allows the application of a 3.7m×7.4m patch with 150×300 pixels. The CNNs used for these studies are Faster R-CNN for 2D object detection, GLPDepth for depth estimation, and DDRNet for semantic segmentation.

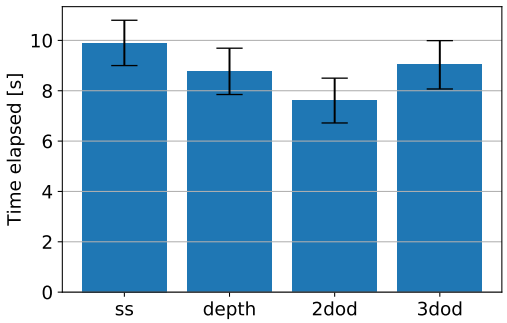
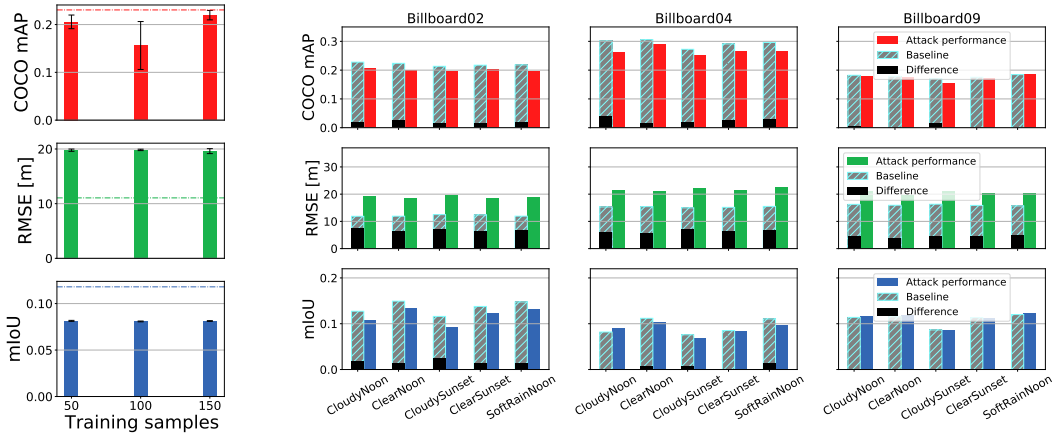


Figure 4.14: Time required for the generation of a dataset item for each task. The error bar represents the standard deviation.

Computation time. The creation of such datasets is a time-consuming task. In this section we show the average time required to generate and save an RGB image and

ground-truth annotation for each task. From the results reported in Figure 4.14, it is clear that the most time-consuming task is semantic segmentation. This is attributed to the fact that segmented images (and ground-truth labels) have the highest resolution, following the Cityscapes format. Then, stereo 3D object detection and depth estimation follow, since the first saves pairs of RGB images and the latter saves an RGB and a depth image. 2D object detection is the least expensive task for generation, since it has to save the RGB image only, while the annotation is a json file. The timing data have been collected during the generation of the same 50-sample test set, changing the seed ten times. Hence, the generation of a 50-sample dataset takes roughly eight minutes on average.



(a) (b)

Figure 4.15: (a) Performance of the attack for each task as a function of the number of samples used for the optimization of the image-agnostic patch. The experiment was performed on billboard02 ten times, changing the seed for each experiment. The dash-dotted line represents the baseline performance (random patch) whereas the error bar represents the standard deviation. (b) Effect of the different weather presets of CARLA on the baseline performance (random patch) against the attack effectiveness for each task. The experiments were performed for three different attack situations with different camera-sunlight relative orientations.

Number of samples for the optimization. An important parameter that must be set is the number of samples required to run the optimizations properly. Too few samples might not be enough to obtain effective attacks, while too many samples slow the generation and optimization processes down. Figure 4.15a shows the attack performance with respect to the baseline CNN (obtained with a random patch) as a function of the number of samples for each task. The results were averaged over ten different optimizations, changing the seed each time. Since optimizing a patch with 100 samples leads to better results overall, we set the training set dimension to 100 images in the generation stage.

Weather conditions. The CARLA simulator allows full control of the weather parameters, ranging from the time of day (i.e., the elevation angle of the sun) to the amount of rain and wetness of the road, from the fog density to the Rayleigh scattering. Given the large number of different parameters to evaluate, which would lead to a combinatorial explosion in the number of settings, we decided to limit the study to the weather presets available in CARLA, ignoring those that lie outside the distribution of the datasets used to

train the CNNs under test (i.e., no heavy rain, puddles, night, or fog). The tested weather presets were `CloudyNoon`, `ClearNoon`, `CloudySunset`, `ClearSunset`, `SoftRainNoon`. Nevertheless, note that in CARLA-GEAR the weather parameters are fully configurable.

We decided to restrict the search to three attack situations with different billboard orientations (and, hence, camera orientation) with respect to the sunlight position: billboard02 has the sun behind (shining directly on the billboard), billboard04 has the sun on the side, and billboard09 has the sun in front. The results in Figure 4.15b show that, for each task tested, the baseline performance (i.e., applying a random real-world patch) when using presets `CloudySunset` and `ClearSunset` is almost always slightly worse. The attack performance resulted not to be particularly influenced by the weather conditions. Hence, we used the `ClearNoon` weather for the following experiments, since it performs slightly better overall. We reported additional illustrations of the adversarial effect of patches under different weather conditions in the supplementary material of [165].

Evaluating Defense Mechanisms

The scenarios addressed in CARLA-GEAR offer a valuable tool for evaluating the performance of defense algorithms, particularly in executing adversarial detection or implementing adversarial masking on top of DNNs for various specific tasks. The results and advantages offered by CARLA-GEAR are detailed in Chapter 5, which includes a comprehensive description of the defense mechanisms proposed in this thesis.

4.4 Discussion and Future Directions

This section summarizes the results achieved in this chapter and outlines open perspectives for future research.

4.4.1 Evaluating the Spatial Robustness

The extensive experiments presented in Section 4.2 offer a new perspective for studying semantic segmentation models in autonomous driving. The tested networks demonstrated a quite interesting degree of robustness to real-world and patch attacks. This was particularly evident fully real-world and simulated settings, where the impact of the patch was mostly confined to the area of the attacked surface. Through this investigation, several promising research directions have emerged.

Addressing the robustness of specific blocks We observed significant variance in the effectiveness of the attacks across the set of models tested. Further investigations revealed that these vulnerabilities in the models are heavily dependent on the intrinsic characteristics of specific blocks used in each architecture, which increase the network susceptibility to real-world attacks.

More specifically, from preliminary experiments, we have observed that the use of global average pooling layers for channel-attention, such as the Squeeze and Excitation Block in BiSeNet [173], can compromise the spatial robustness of the model. In brief, it facilitates an adversarial patch in influencing features belonging to areas far from the patch itself.

These preliminary insights inspire further in-depth analysis with the goal of establishing design principles to enhance the robustness of future models.

On the relationship between targeted and untargeted attacks. It is worth noting that, despite the apparent differences between the target and untargeted formulations, our experiments have revealed that untargeted attacks often exhibit patterns that closely resemble targeted ones. More specifically, untargeted attacks tend to focus on a particular target class and propagate their influence beyond the mask region (see ICNet and DDRNet in Figure 4.4). We believe that this phenomenon is not only reasonable but also linked to the universal property of a real-world attack, where the fact that the patch needs to generalize the adversarial effect across different images is, in some sense, closely tied to the necessity of addressing a specific target class. This insight may highlight the need of investigate the interconnections between untargeted and targeted approaches in the realm of univesal and physically-realizable adversarial attacks.

4.4.2 Future directions with the use of simulators

In the second part of the chapter, we introduce CARLA-GEAR , a tool for an automatic generation of adversarial, photo-realistic synthetic datasets in driving scenarios. The experimental results presented in this chapter, and in the next chapter 5, focus on its use for evaluating real-world robustness of pretrained models and defense strategies. This represents a step towards benchmarking real-world attacks in outdoor driving scenarios.

Ethically, this work highlights security and safety concerns regarding the deployment of safety-critical systems that rely on AI vision applications. We contend that the extensive customization options and photo-realism provided by CARLA-GEAR not only offer a systematic tool for evaluation but also reveal crucial insights into the effectiveness of testing neural networks and defense strategies against physical adversarial attacks in driving scenarios.

Despite the novelty of this work, it is important to acknowledge some limitations and open problems, which also suggests possible directions for future research.

Domain shift on CARLA. Although the meshes used in Town10HD of CARLA are photo-realistic, there is a clear domain shift between the CARLA-generated images and common real-world datasets used for scene-understanding (e.g., COCO, CityScapes, Kitti). This raises concerns about the generalizability of the evaluation conducted with our tool to real-world scenarios. Nevertheless, real-world evaluations are challenging and expensive to achieve, as pointed out by related studies (e.g., [24] and [61]), thus remarking the necessity of using simulators. The need for clarification on the proper understanding of result generalization opens further avenues for investigation. To address this, future efforts might focus on two areas: firstly, understanding the level of transferability in light of recent advancements in photorealism within driving simulators; secondly, developing new optimization attack methods that adapt adversarial features to transfer also within real-world scenarios.

Testing scenarios. Another issue with the simulated images is that the urban scenarios in CARLA towns are inevitably too limited and thus do not allow for a comprehensive evaluation of driving scenarios. In other words, the scenarios addressed in this paper do not cover all the possible patch attacks in the wild. In our experiments, we tried to bridge the gap between the complexity of real-world scenarios and simulated environments by randomizing Non-Playable Characters (NPCs), such as pedestrians and cars, across all images within a given scenario. Additionally, we proposed two types of attack settings: static attacks using adversarial billboards and dynamic attacks with adversarial trucks. These settings are designed to effectively generalize CARLA-GEAR across various attack scenarios. For instance, we can simulate adversarial road signs or walls using billboards. Despite these efforts, we acknowledge that the intrinsic limitations of the CARLA environments might not provide the same diversity of scenarios as the real world. This recognition guides our future work, where we plan to delve deeper into these issues.

Computational and timing costs. The proposed dataset generation pipeline is computationally intensive and requires a considerable amount of time, even with a powerful machine. Such costs are mainly due to the optimization algorithms involved in crafting proper adversarial attacks. Future work will need to refine these optimizations, thus reducing the processing time and making the pipeline affordable even for more resource-constrained architectures. Nevertheless, at the current stage, we aim at alleviating this problem by (i) providing a set of datasets with pre-computed patches for a set of CNNs spanning four different tasks, and (ii) providing support for custom patch and dataset generation upon request to the authors.

Ad-hoc robustness metrics. To comprehensively evaluate the real-world robustness of CNNs, designers should face with the definition of proper ad-hoc, model-specfic metrics that can better express the model robustness. In this work, we considered common task-based metrics (such as mAP for object detection, MioU for semantic segmentation, and RMSE for depth estimation), which are typically involved in evaluating the model performance on not attacked scenarios. The use of these metrics is a common practice for the majority of related work. Future work should instead be devoted to the derivation of more expressive metrics, which better depict and measure the concrete adversarial robustness of a given model.

Chapter 5

Interpretable Defenses Against Real-World Attacks

Recent works have proposed various methods to mitigate real-world adversarial attacks. However, as discussed in Section 2.4.2, merely pointing out the defense performance is not sufficient. It is also crucial to consider practical constraints, which may include computational cost, interpretability of the steps taken by the method, and an 'explainable sense' of robustness (i.e., achieving a sufficient level of transparency in understanding the potential failures and weaknesses of the approach).

As known in the literature, achieving these requirements can be hard and might constrain the performance of defense strategies. To address these challenges, this chapter introduces several works focused on developing effective yet interpretable defense mechanisms for both adversarial detection and adversarial masking. The primary emphasis on high robustness and interpretability is based on an over-activation analysis, which is introduced in Section 2.4.3 and explored more in depth in Section 5.1.

We organized the chapter's sections as follows: Firstly, to introduce and consolidate the theory behind the over-activation analysis, we reexamine the over-activation phenomenon produced by real-world adversarial attacks. As previously mentioned in the related literature (Section 2.4), Yu et al. [116] investigated the relationship between over-activation in internal network layers and adversarial patches, with a focus on image classifiers. Inspired by their analysis, we extend that investigation from a different viewpoint, broadening its applicability to dense prediction tasks.

Then, we report in the subsequent sections three defense algorithms. In Section 5.2.1, we present a fast adversarial detection mechanism tested within semantic segmentation tasks to quickly and robustly flag the presence of adversarial patches. Then, in Section 5.1, we extend the use of over-activation analysis to the case of masking by introducing modules that enable the extraction of over-activation-based heatmaps without relying on unpredictable blocks like additional neural networks. Finally, we tackle the challenge of implementing more efficient and straightforward adversarial masking, with a focus on computational cost and inference time, in multi-frame applications. This is achieved by providing further insights into the over-activation phenomenon and integrating proper temporal attention mechanisms.

5.1 Over-Activation Analysis

The theoretical insights provided by Yu et al. in [116] offer a mathematical explanation on the existence of over-activation in internal neurons when the model is exposed to an adversarial patch. In their formulation, Yu et al. confined their analysis to image classification scenarios, and introduce multiple assumptions during the theoretical derivation.

In this section, inspired by the work of Yu et al., we propose new insights into the necessity of over-activating internal features to successfully inject adversarial effects in real-world attack scenarios. We employ straightforward algebraic constructions and focus on models for dense prediction tasks

Let us consider a semantic segmentation model $f : \mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{N_y\times H\times W}$. Let us denote L_B as the last layer of the backbone, i.e., the last convolutional block before the fully connected classification layer. For simplicity, we can assume $H^B = H$ and $W^B = W$, indicating that the height and width of the backbone's output match those of the input. Let us also focus on a targeted output pixel (i, j) that we want to misclassify through a targeted adversarial attack, i.e., maximizing the probability of the model output towards an adversarial class y_{adv}.
We define $h \in \mathbb{R}^{C^B\times 1\times 1}$ as the feature vector at layer L_B corresponding to pixel (i, j). [1] Similarly, the feature vector for pixel (i, j) in the case of the attacked image $\tilde{\mathbf{x}}$ is represented as $\tilde{h} \in \mathbb{R}^{C^B\times 1\times 1}$, computed from $f^{0\rightarrow B}(\tilde{\mathbf{x}})$. Note that the attacked image $\tilde{\mathbf{x}}$ is generated using a **universal** patch attack δ.

As illustrated in Figure 5.1, the perturbation in the feature space of the layer L_B, concerning the feature at the position (i, j), can be represented as

$$\hat{h} = \tilde{h} - h.$$

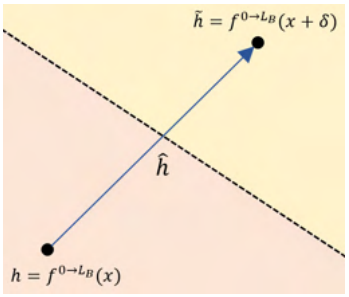


Figure 5.1: Representation of the used notation in the features space. The yellow area depicts the y_{adv}-class region, while the orange one is the correct y-class region.

Let us assume that y is the ground truth class for the pixel (i, j). The attack is successful if the final logit for y_{adv} is significantly higher than for any other class $y \in \{1, \ldots, N_y\} \setminus \{y_{adv}\}$. This ensures that the output probability for the class y_{adv} is close to 1. Let us also assume that the non-attacked features h achieved a high probability score with the correct class y, meaning the logit vector of h in y is much higher than with the other classes. Thus, to craft an effective attack, we need:

[1] Note that for simplicity, we have omitted the layer and spatial indices in h, as in this analysis we consistently refer to layer L_B and pixel (i, j). For clarity, $h = \mathbf{h}_{B,(i,j)} = f^{0\rightarrow B}(\mathbf{x})$

$$W_{y_{adv}}^T \tilde{h} \gg W_y^T \tilde{h}$$

To simplify the problem, we introduce the weights $\bar{W} = W_{y_{adv}} - W_y$, such that we can write the previous disequention as a binary class problem:

$$\bar{W}^T \tilde{h} \gg 0$$

Thus, considering that $\tilde{h} = \hat{\mathbf{h}} + h$, as illustrated in Figure 5.1, we can write:

$$\bar{W}^T \hat{h} + \bar{W}^T h \gg 0$$

From $a \cdot b = \|a\| \cdot \|b\| \cdot \cos(\angle(a, b))$, we have:

$$\|\bar{W}\| \cdot \left\|\hat{h}\right\| \cdot \cos(\angle(\hat{h}, \bar{W})) + \|\bar{W}\| \cdot \|h\| \cdot \cos(\angle(h, \bar{W})) \gg 0$$

$$\implies \left\|\hat{h}\right\| \cdot \cos(\angle(\hat{h}, \bar{W})) + \cdot \|h\| \cdot \cos(\angle(h, \bar{W})) \gg 0$$

We now make the following assumptions to account for the dynamics of universal adversarial attacks:

1. *Alignment of universal perturbation with target adversarial class weight*: A crucial assumption is $\cos(\angle(\hat{h}, W_{y_{adv}})) \approx 1$. This suggests that for a successful universal attack, the perturbed feature vector at pixel (i, j) should closely align with the weight vector of the target class y_{adv}. This alignment is a key characteristic of universal attacks (perturbations and patches), which are effective across various input samples (see [174]). Formally, we can support this assumption by observing that, to fool different samples $\mathbf{x} \in \mathbf{X}$, the direction should, on average, align to the target class in order to generalize:

   $$\mathbb{E}_{\mathbf{x} \in \mathbf{X}} \cos(\angle(\hat{h}|_{\mathbf{x}}, W_{y_{adv}})) \to 1$$

   where $\hat{h}|_{\mathbf{x}}$ indicates the feature perturbation for the pixel (i, j) induced by δ when applied to a generic input $\mathbf{x}$. Following this assumption, since we have reformulated the problem as binary class problem, we have

   $$\cos(\angle(\hat{h}, \bar{W})) \approx 1$$

2. *Alignment of the non-attacked sample with the correct class weight*: Building on the initial assumption that the input $\mathbf{x}$ is correctly predicted with the ground class y, we have that $1 \leq \cos(\angle(h, W_y)) > 0$. If the classification is given with high confidence, the previous values should be close to 1.

   Following similar considerations, we have $-1 \leq \cos(\angle(h, W_{y_{adv}})) \leq 0$, implying that the input has been classified with $y \neq y_{adv}$.

With these assumptions, we have the following derivation:

$$\left\|\hat{h}\right\| \cdot \underbrace{\cos(\angle(\hat{h}, \bar{W}))}_{\approx 1} + \cdot \|h\| \cdot \underbrace{\cos(\angle(h, \bar{W}))}_{<0} \gg 0 \quad \implies \quad \left\|\hat{h}\right\| \gg \|h\| \cdot \alpha$$

where $\alpha > 0$ is defined as $\alpha = -\cos(\angle(h, \bar{W}))$. Note that α is indicative of the extent to which the non-attacked features align with the correct class-y features. We assume that for a well-trained model, an input classified correctly could yield $\alpha \approx 1$.

This analysis prove that $\hat{h}$ needs to produce a high over-activation, which implies an over-activation also for the vectors $\tilde{h}$.

It is important to note that the proposed analysis is also valid for the case of universal adversarial perturbations. In fact, the critical aspect is the universal property, which is assumed both in the case of patches/objects and universal perturbations. Different considerations arise in the case of non-universal perturbations (common adversarial examples), where the condition $\cos(\angle(\hat{h}, W_{y_{adv}})) \rightarrow 1$ could not be true.

Although these preliminary insights showing an over-activation required by universal attacks (either patch or perturbation), the subsequent paragraphs will delve into additional constraints specific to real-world adversarial objects (e.g., patches), which further increase the level of over-activation required by the attacks.

Spatial Decay for Localized Attacks

In [154], Luo et al. discussed the properties of effective receptive fields to measure how much each input pixel can impact the output features. Specifically, let us consider a CNN with N convolutional layers. Luo et al [154] studied the impact of the input pixel (p_1, p_2), with respect to the feature vector corresponding to a pixel at the position (i, j).

The effective receptive field analysis measures how much a pixel $\mathbf{x}_{(p_1,p_2)}$ contributes to $\mathbf{h}_{l,(i,j)}$ by calculating the expectations of the partial derivative $\frac{\partial \mathbf{h}_{l,(i,j)}}{\partial \mathbf{x}_{(p_1,p_2)}}$. The authors demonstrated that the analysis converges to the probability density function of a 2D Gaussian distribution:

$$\Phi_N \sim \mathcal{N}\left(d((i,j),(p_1,p_2)), \mathrm{Var}[\Omega_n]\right), \quad n = 1, 2, \ldots, N$$

where $d((i,j),(p_1,p_2))$ is a l_p-distance between (i,j) and (p_1,p_2), and Ω_n is a random variable following the distribution of the weights of the n-th convolutional layer.

Based on this assumption, we can infer that the activation induced by an adversarial pixel should be more pronounced when the attacked pixel is distant from (p_1, p_2). In other words, the propagation of the adversarial effect should result in significant over-activation of specific features to counterbalance the spatial decay constraint. Indeed, as demonstrated in the previous chapter, this spatial decay factor significantly increases the challenge of crafting adversarial patches within large images.

5.2 Fast Patch Detection Algorithm

Leveraging insights from previous observations, this section introduces the first defense method of the chapter, namely the Fast Adversarial Patch Detection Algorithm (FPDA) [24]. The evaluation of the method follows the setup presented in Chapter 4, covering both digital and real-world attacked scenarios for semantic segmentation models.

5.2.1 Proposed Method

This section describes the FPDA, designed to recognize output predictions affected by adversarial patches (adversarial detection). Inspired by the HN method [105], the basic idea of the proposed approach relies on identifying and measuring the presence of over-activated internal features, which are likelly induced by adversarial attacks.

Given a semantic segmentation (SS) model f and a specific layer L_l, the pseudo-code of FPDA is reported in Algorithm 1. The function `getActivation`$(f, \mathbf{x}, l)$ (*line 1*) returns all the neuron activations at the layer L_l, denoted by

$$\mathbf{h}_l = \{\mathbf{h}_{l,(c,i,j)}, \quad \forall c, i, j\}.$$

Since the number of neurons at layer L_l might be large in semantic segmentation models, making the next algorithmic steps computationally expensive at run-time and therefore not suited for real-time applications, a *features-compression* is performed at *line 2*, returning

$$\mathcal{C}_l = \{\max_{c \in C_l} |\mathbf{h}_{l,(c,i,j)}|, \quad \forall i, j\}.$$

This operation preserves the properties of the over-activated neurons by using the max operator.

Algorithm 7 Fast Patch Detection Algorithm

1: $\mathbf{h}_l = \texttt{getActivation}(f, \mathbf{x}, l)$
2: $\mathcal{C}_l = \{ \ \max_{c \in C_l} \ |\mathbf{h}_{l,(c,i,j)}| \ , \quad \mathbf{h}_{l,(c,i,j)} \in \mathbf{h}_l\}$
3: $\hat{\mathcal{C}}_l = (\mathcal{C}_l - \mu_l)/\sigma_l$
4: $\mathcal{S}_l = \hat{\mathcal{C}}_l.\texttt{where}(\hat{\mathcal{C}}_l > \theta_\nu)$
5: $score = \sum \mathcal{S}_l$
6: **if** $score > \varrho$ **then**
7: *Return* **True**
8: **end if**
9: *Return* **False**

Then, $\mathcal{C}_l$ is normalized (*line 3*) using the mean μ_l and standard deviation σ_l, which are computed offline from a dataset of clean images (i.e., no patched inputs). The normalized features are so denoted as $\hat{\mathcal{C}}_l$.

To filter out all the common activations resulting from clean inputs, a subset of $\hat{\mathcal{C}}_l$ is selected (*line 4*), including only those neurons with an activation larger than a *selection-threshold* θ_ν and thus deemed as unsafe. The threshold θ_ν is determined offline as the ν*-percentile* value computed on the normalized features $\hat{\mathcal{C}}_l$ of the previously defined clean dataset.

Finally, a score value is obtained (*line 5*) as the sum of these over-activated features. Such a score is then compared (*line 6*) with a *decision-threshold* ϱ in order to decide whether the prediction $f(x)$ is safe or not (i.e., whether $\mathbf{x}$ is genuine or includes an adversarial patch). Threshold ϱ is tuned offline on a second dataset, composed of both clean and patched images.

The performance of the proposed algorithm depends on ν, ϱ, and the two datasets used to extract μ_l, σ_l, θ_ν, and ϱ. These parameters and the related tuning strategies are discussed in details in evaluation part (5.2.2).

It is useful to highlight the main differences between the proposed method and HN [105]. Although both achieve similar detection performance, our approach has a significantly smaller computation time thanks to the features-compression step described above. Furthermore, while the features-selection step of HN is performed at run-time on the layer activations, our approach exploits an off-line computation of a threshold θ_ν, computed on a given clean dataset, to reduce the run-time detection latency.

5.2.2 Experiments

This section provides a set of experiments aimed at assessing the detectability of patched images and the real-time performance of the algorithm. A comparison with the original HN formulation is also provided. Finally, a comprehensive analysis against defense-aware attacks is shown to illustrate the strengths and weaknesses of FPDA. The setup of the experiments is the same as proposed in Chapter 4, where both digital patches on Cityscapes and real-world patches have been evaluated.

Tuning the thresholds

The Cityscapes dataset is used to set the algorithm's parameters μ_l, σ_l, ν*-percentile*. In particular, a set of features $\mathcal{C}_l$ extracted from 1000 images of the original training set are used to compute μ_l and σ_l. Then, after the normalization, the corresponding features $\hat{\mathcal{C}}_l$ are used to compute the ν*-percentile* (where ν is set to 0.999 for both HN and FPDA) and so θ_ν. A second dataset (composed of other 1000 clean images from the training set, plus the corresponding adversarial images) is used to compute the decision threshold ϱ.

The detection algorithm requires specifying a given layer L_l. The choice of the layer has been the subject of many preliminary experiments. The real-time SS models considered in this paper fuse information extracted from different parallel branches. The layer L_l is chosen as the first layer that joins all the model's branches, which resembles the last or close to the last conv block. We noticed that it is reasonable to detect anomalies in the first fusion point: it might be possible to create adversarial patches capable of bypassing the detection mechanism placed within some branches of the model by exploiting the vulnerabilities of others.

For completness, in Table 5.1 we report the list of layers addressed for FPDA and HN for DDRNet, BiseNet and ICNet. We also list the settings used to evaluate FPDA and HN with other vision tasks for the evaluation performed through CARLA-GEAR in Section 5.4.

Net	Layers (FPDA and HN)
DDRNet	layer5
BiseNet	context_path_2
ICNet	conv5-4-k1
FRCNN	backbone.body.layer1[1].conv1
RetinaNet	backbone.body.layer4
Ababins	encoder.original_model.blocks[1][0]
GLPDepth	encoder.patch_embed1.proj
Stereo-FRCNN	RCNN_smooth1

Table 5.1: Settings for FPDA and HN. For reproducibility, layer names refer to the modules names available in the code at github.com/retis-ai/CARLA-GeAR.

On the validity of the method

Figure 5.2 reports the distribution of clean and patched images extracted from the validation set of Cityscapes as a function of the *score* computed by the proposed method with the BiSeNet model. The same kind of patches analyzed in the untargeted attacks of Section 4.2.2 are used ($150 \times 300, 200 \times 400, 300 \times 600$).

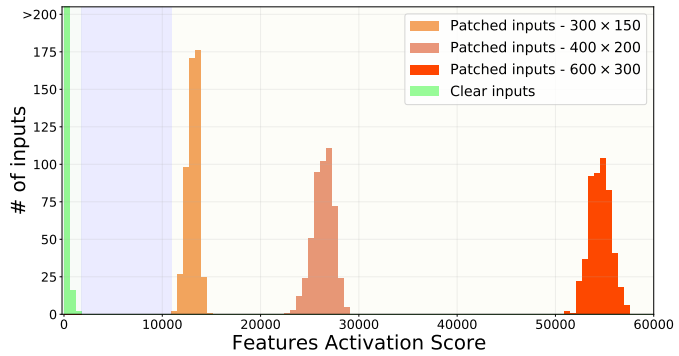


Figure 5.2: Distribution of the clean and patched images as a function of the features over-activation *score* computed by the Algorithm 1 with BiSeNet. The grey area highlights the set of ϱ thresholds that achieve a perfect detection accuracy with the tested images.

As shown in Figure 5.2, the distribution of the clean images is much closer to zero than all the other adversarial distributions. In particular, the larger the adversarial patch (and consequently more effective), the higher the score computed by Algorithm 1. The area of score values colored in grey highlights the set inside which any selected threshold ϱ is able to detect all the adversarial images without introducing false negatives (i.e., no clean image is detected as unsafe).

Timing analysis

The following experiments show how the proposed method allows the processing of large-sized layer activations (see Table 5.1) without affecting the timing performance of the

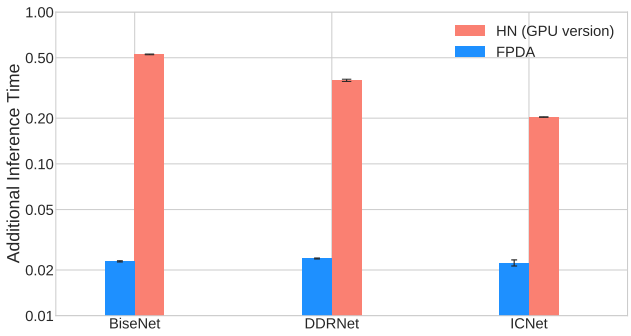


Figure 5.3: Additional inference time computed as the ratio between the execution time of the detection algorithm and the original model inference time.

model. Figure 5.3 plots the additional inference time averaged over 8000 iterations: the y-axis reports the relative execution time fraction introduced by Algorithm 1 with respect to the nominal inference time required by the model. As it can be noted from the figure, the proposed method adds a small latency, whereas HN presents larger additional execution times: from 20% with the ICNet model to 50% with BiSeNet.

While it is fair to remark that HN is not conceived for semantic segmentation models (where the number of features to take into account is usually larger than common image classification models), the proposed approach solves this issue by processing a large number of features in negligible additional time.

Defense-aware attack

Although the adversarial patches tested in Section 5.2.2 are easily detectable by FPDA, it is important to assess the goodness of FPDA against ad-hoc attacks that exploit the knowledge of the applied defense. To this purpose, an extension of the attack formulation proposed in this paper was concevied to craft untargeted adversarial patches with the intention of also deceiving the FPDA.

Since the FPDA is based on the detection of over-activated features, we crafted patches that are adversarial (i.e., capable of misclassifying a large number of pixels predicted in SS outcomes), while, at the same time, are able to keep the activated features within the range of the original distribution (i.e., minimizing the over-activations). This is accomplished by solving the iterative optimization method discussed in Section 2.3.1 but using a loss function $\mathcal{L}(f(\tilde{x}), y)$ that includes both the adversarial loss function $\mathcal{L}_{adv}(f(\tilde{x}), y)$ (in short $\mathcal{L}^{\tilde{x}}_{adv}$) and an additional *activation loss* $\mathcal{L}^{\tilde{x}}_{\hat{\mathcal{C}}_l}$. The former is the loss function introduced in Section 2.3.1, while the latter is a new loss function that returns a cost proportional to the squared over-activation of the normalized features $\hat{\mathcal{C}}_l$ (computed by performing the first operations of Algorithm 1 on $\tilde{x}$), i.e, $\mathcal{L}^{\tilde{x}}_{\hat{\mathcal{C}}_l} = ||\hat{\mathcal{C}}_l||^2_2$.

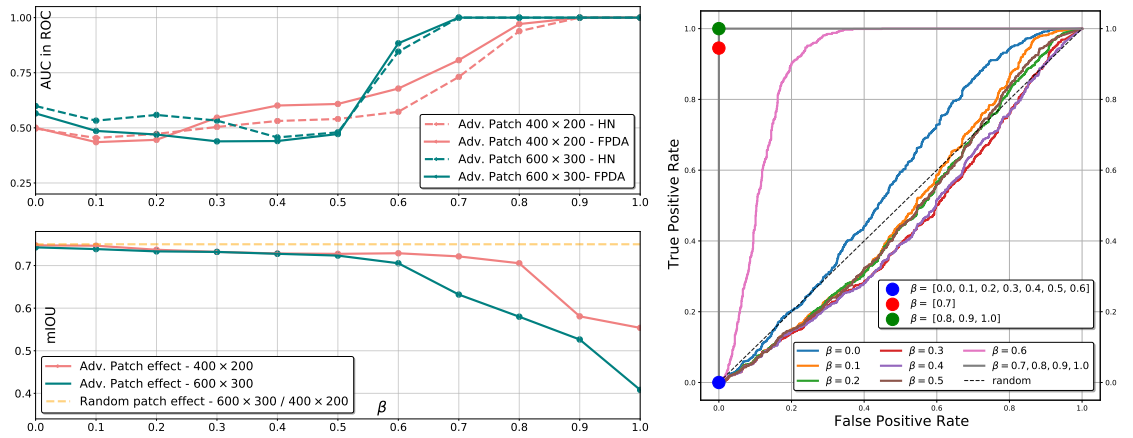
Therefore, the gradient of $\mathcal{L}(f(\tilde{x}), y)$ used during the optimization method was redefined as:

$$\nabla_{\delta_k}\mathcal{L}(f(\tilde{x}), y) = \beta \cdot \frac{\nabla_{\delta_k}\mathcal{L}^{\tilde{x}}_{adv}}{||\nabla_{\delta_k}\mathcal{L}^{\tilde{x}}_{adv}||_2} - (1-\beta) \cdot \frac{\nabla_{\delta_k}\mathcal{L}^{\tilde{x}}_{\hat{\mathcal{C}}_l}}{||\nabla_{\delta_k}\mathcal{L}^{\tilde{x}}_{\hat{\mathcal{C}}_l})||_2} \ , \tag{5.1}$$

where β is a parameter introduced to balance the importance of $\mathcal{L}_{adv}$ and $\mathcal{L}_{\hat{c}_l}$, and $k = \{1, ..., N_p\}$. In particular, when $\beta = 1.0$, the resulting patch will be optimized to minimize the activated features. Conversely, moving β to lower values reduces the previous effect and increases the adversarial strength.

The following experiments investigate the relation between detectability and adversarial effect. To this end, several adversarial patches (300x600 and 200x400 pixels) are crafted using several values of $\beta \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. In particular, given an adversarial patch crafted with a certain β, the detectability is evaluated by performing an ROC analysis with 1000 images of the Cityscapes train set and the corresponding attacked ones. Thus, such a subset is used to study the True Positive Rate and False Positive Rate as a function of the decision threshold ϱ. On the other side, the adversarial effect is evaluated using the mIoU computed on the Cityscapes validation set. Lower values indicate a more effective adversarial attack.

Figure 5.4a shows a comparison between the detection accuracy, specified through the AUC values (extracted from each ROC curve) and the adversarial effect obtained with each tested version of β. Clearly, for low values of β (when the optimization is mainly focused on keeping the activated features small), the detectability of all the tested methods is very poor. However, these patches have a low adversarial effect since the obtained mIoUs are close to the ones computed with random patches. Increasing the value of β, the adversarial effect of the patches increases but, at the same time, also their detectability grows. These results remark that it exists an intrinsic relation between the adversarial effect and the over-activation of features, such that highly effective patches are more prone to be detected by the proposed strategy.



(a) (b)

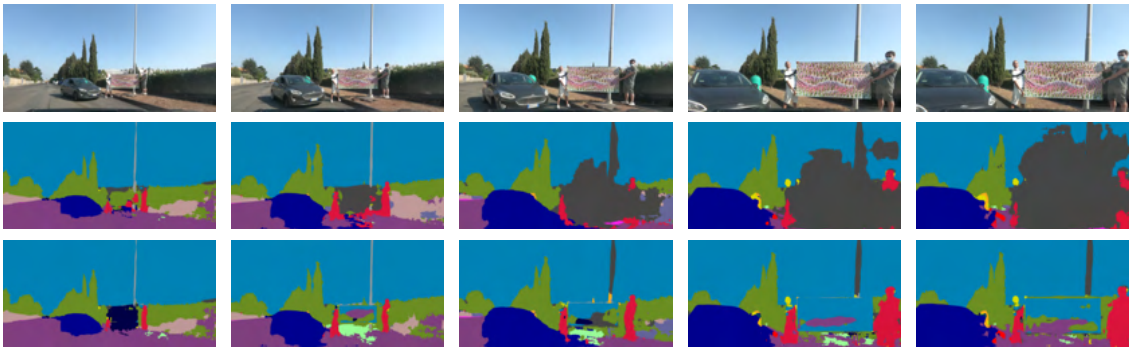
Figure 5.4: Figure (a) reports a comparison between the detectability and adversarial effect of mulitple patches crafted changing the β parameter; Figure (b) plots the ROC curves corresponding to all the previous tested adversarial patches and highlights their TFPs/FPRs corresponding to an unique threshold selected with $\beta = 0.8$. These results were obtained with the ICNet model.

Figure 5.4b illustrates another important result, showing all the ROC curves computed for the 300×600 patches (their AUC score is the one shown in Figure 5.4a). By looking at these curves, it is to possible to figure out the True Positive Rate (TPR) and False Positive Rate (FPR) with respect to a fixed threshold ϱ. The threshold ϱ is set as the

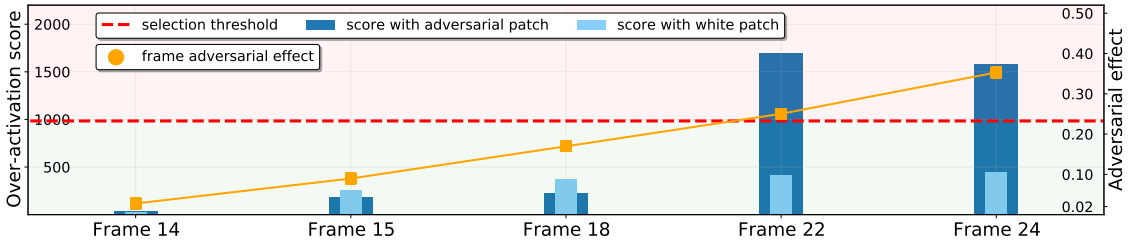
optimal value (cut-off point) of the ROC analysis performed with $\beta = 0.8$, where the AUC achieves 1.0, meaning that safe and unsafe samples can be perfectly classified for that β value.

As highlighted by the plot's legend, only the most dangerous patches are detected correctly (those with $\beta \geq 0.8$). Also, and most importantly, all the ROC points have an FPR equal to 0.0, meaning that clean images are always classified as safe.

The above experiments were also performed on the BiSeNet and DDRNet models, showing similar results (see supplementary material [24]).



(a)



(b)

Figure 5.5: Real-world evaluation on ICNet of the attack method and the defense algorithm. In inset (a), already shown in Section 4.2, the first row contains the original images with the printed adversarial patch, while the second row contains their obtained semantic segmentations. The third row instead shows the outcomes obtained by masking the patch areas of the previous inputs with white pixels. Inset (b) shows the over-activation score of the corresponding frames for the SS predictions in rows 2 and 3, and which of them are detected by using a proper selection threshold. Also, the adversarial effect is shown for each frame, which is extracted by comparing the original attacked images with the corresponding white-masked version.

Detectability of a Real-World Patch

We tested FPDA on the real-world frames presented in Section 4.2. Figure 5.5b reports the *over-activation score* corresponding to each frame shown in Figure 5.5a. The red dashed line represents the threshold ϱ, used to distinguish safe and unsafe predictions. To provide a clear visualization of the spatial effect caused by the printed adversarial patch (second row in Figure 5.5a), we also report the SS obtained from the same images where the adversarial patch was masked with white pixels (third row in Figure 5.5a) and their corresponding scores (light blue in Figure 5.5b).

Since the mIoU does not properly encode the adversarial effectiveness on individual images, we show a different metric that we call *adversarial effect*, computed as $1 - \frac{1}{|\mathcal{N}\setminus\tilde{\mathcal{N}}|}\sum_{i\in\mathcal{N}\setminus\tilde{\mathcal{N}}}(y_i == SS_i(x))$. The adversarial effect measures the average pixel dissimilarity between the predicted SS and a ground truth y. Given the absence of proper ground truth labels y, we use the SS predicted from the white-masked images. As done in Section 4.2.2, when evaluating the effect of patches on Cityscapes, we did not consider those pixels corresponding to the masked areas (which approximate the patch placement in the real world).

Note that in this real-world experiment the tuning of ϱ was performed on the same 1000 images of the Cityscapes dataset with an adversarial patch having $\beta = 0.8$ and 300×600 (the same patch used to compute the threshold in Figure 5.4b). However, it is worth noting that transferring patches in a real-world environment reduces the activation level of the perturbed features (as well as their adversarial effect). To this purpose, the threshold ϱ should be decreased. This can be obtained by considering an FPR equal to 0.01 as a reasonable compromise (i.e., 1% of clean images in the calibration set are wrongly classified as unsafe). This helps shift the threshold to a lower value, more likely to be closer to the over-activation score implied by patches transferred in the real world.

As it can be noted from Figure 5.5b, frames 22 and 24 are above the threshold ϱ (red line) meaning that the detection mechanism works also in a real world scenario. Moreover, their masked counterparts achieve a lower score, clearly below threshold ϱ, which stems to reason that high over-activations are mainly caused by the presence of the printed adversarial patch. Conversely, earlier frames are classified as safe, meaning that the internal features do not have a considerable number of over-activated values. In fact, in these latter frames the patch has a lower adversarial effect, which is under 0.15, meaning that less than 15% of out of patch pixels are classified differently from the masked versions.

5.3 Adversarial Masking via Over-Activation Analysis

In this work [107], we proposed a novel use of the over-activation analysis. The objective here is to address over-activations by extracting adversarial masks, thereby mitigating the impact of physically-realizable adversarial objects. The proposed method, namely *Z-Mask*, utilizes specific processing modules, which have been tested in this section and in Section 5.4 across 2D and 3D object detection, semantic segmentation, and depth estimation models. Furthermore, as in the first study conducted for FPDA, we analyzed the method against adaptive attacks to assess a worst-case scenarios.

5.3.1 Proposed Method

Inspired by the spatial propagation effect of CNNs [175] and the over-activation phenomenon, we noticed that a set of shallow layers contains high/medium over-activations in the spatial image areas corresponding to adversarial objects, while in deeper layers such over-activations grow in magnitude while referring to lower spatial resolutions. Based on such evidences, *Z-Mask* combines the analysis of multiple layers to precisely detect and mask potential adversarial objects. Figure 5.6 illustrates the proposed defense approach

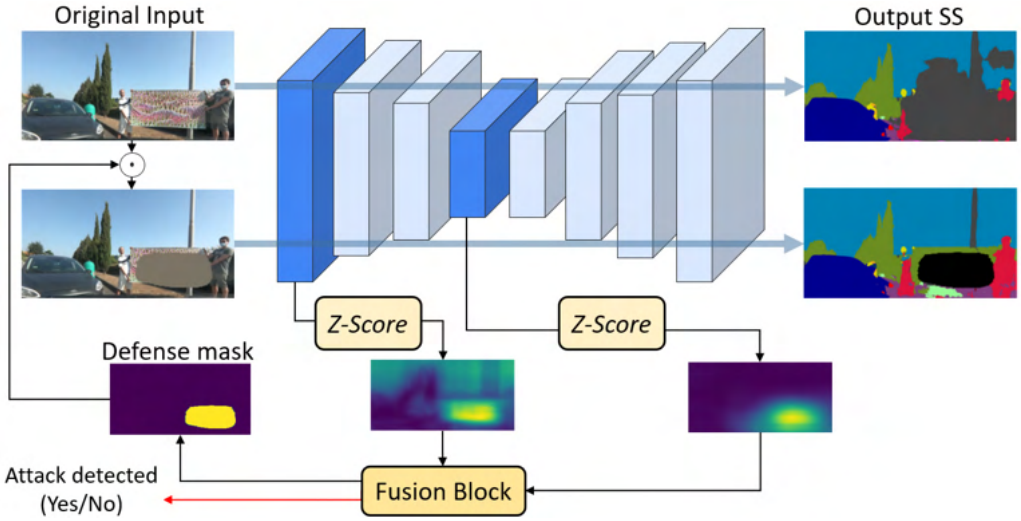


Figure 5.6: Illustration of the proposed approach.

for the case of semantic segmentation. To extract preliminary adversarial region proposals, *Z-Mask* runs an over-activation analysis (presented in Section 5.3.1) on a set of selected layers. This analysis exploits a *Spatial Pooling Refinement* (SPR) to filter out high-frequency noise in over-activated regions. For each of these layers, the analysis produces an adversarial region proposal expressed through a heatmap. Then, all the heatmaps are aggregated into a *shallow heatmap* $\mathcal{H}^{\mathcal{S}}$ and a *deep heatmap* $\mathcal{H}^{\mathcal{D}}$, which summarize the over-activation behavior at two different depth levels. Finally, these two heatmaps are processed by a *Fusion and Detection Block* (described in the following Sections) that flags the presence of an adversarial object and generates the corresponding defense mask.

From an Over-Activation Heatmap to a Adversarial Mask

Given an attacked input $\tilde{\mathbf{x}}$, a defense mechanism based on internal analysis studies one or more deep features to extract a heatmap $\mathcal{H}_l \in \mathbb{R}^{H^l \times W^l}$, which highlight the position of the adversarial object within the input image.

Then, the heatmap can be scaled and binarized, using a threshold, to obtain a mask $\mathcal{M}_\delta \in \{0,1\}^{H \times W}$, where the elements set to 0 are deemed adversarial while those set to 1 are not. Formally speaking, in the case of a single layer under analysis L_l, the above steps can be summarized by means of a function

$$\Lambda^\xi : \mathbb{R}^{C^l \times H^l \times W^l} \rightarrow \{0,1\}^{H^l \times W^l}, \tag{5.2}$$

which takes as input the features $\boldsymbol{h}_l$ from a given layer L_l and produces a mask $\mathcal{M}_\delta$ based on a pre-determined threshold ξ. The resulting mask $\mathcal{M}_\delta$ can then be applied following the Equation 2.12

Generalizing the formulation for the case of multiple layers under analysis, we can define Λ^ξ as:

$$\Lambda^\xi : \mathbb{R}^{C^{l_1} \times H^{l_1} \times W^{l_1}} \times ... \times \mathbb{R}^{C^{l_{|L_A|}} \times H^{l_{|L_A|}} \times W^{l_{|L_A|}}} \rightarrow \{0,1\}^{H \times W}, \tag{5.3}$$

where $L_A = \{l_1, l_2, \dots, l_{|L_A|}\}$ is a set of layers under analysis.

Layer-Wise Over-Activation Analysis

In this work, the adversarial mask $\mathcal{M}_\delta$ is generated by leveraging multiple over-activation analysis (from multiple layers), which are then aggregated through a *Fusion Block*.

As introduced in Section 2.1, let $\boldsymbol{h}_l \in \mathbb{R}^{C^l \times H^l \times W^l}$ be the output features of layer L_l, obtained during the forward pass of $f^{0 \rightarrow l}(\mathbf{x})$, where H^l and W^l are its spatial dimensions. The heatmap $\mathcal{H}_l$ is obtained by applying the following operations to $\boldsymbol{h}_l$ (illustrated in Figure 5.7). First, for a layer L_l, the channel-wise *Z-score* $\boldsymbol{z}_l = \frac{\boldsymbol{h}_l - \mu^l}{\sigma^l}$ of $\boldsymbol{h}^l$ is computed, where μ^l and σ^l are the channel-wise mean and standard deviation of the output features, respectively, obtained from a dataset $\boldsymbol{X}$ that does not include attacked images. The *Z-score* is then processed in cascade by a sequence of m Average-Pooling operations $(A_1, \dots, A_i, \dots A_m)$ as follows:

$$\begin{cases} \boldsymbol{a}_i^{(l)} = \mathcal{R}(A_i(\mathcal{R}(\boldsymbol{z}^{(l)}))) \odot \frac{\boldsymbol{a}_{i-1}^{(l)}}{\|\boldsymbol{a}_{i-1}^{(l)}\|_\infty}, \quad i = 1, \dots, m \\ \boldsymbol{a}_0^{(l)} \equiv 1, \end{cases} \tag{5.4}$$

where each A_i has kernel size k_i and $\mathcal{R}$ is an operator that resizes (by interpolation) the spatial dimensions of a given tensor to a configurable size $H^\mathcal{R} \times W^\mathcal{R}$. Note that the $i^{\texttt{th}}$ kernel is larger than the $(i+1)^{\texttt{th}}$ one. Also, the resize operation is performed before and after each A_i to enable the use of the pixel-wise product and the same sequence of Average-Pooling operations on different network layers, which otherwise may require working with tensors of different sizes. The rationale for using such Average-Pooling operations is the following. Observe that the *Z-score* itself provides a pixel-wise metric capable of highlighting the over-activated pixels (i.e., pixels with internal activation values that are significantly far from $\mu^{(l)}$ in terms of $\sigma^{(l)}$). However, since we aim at masking

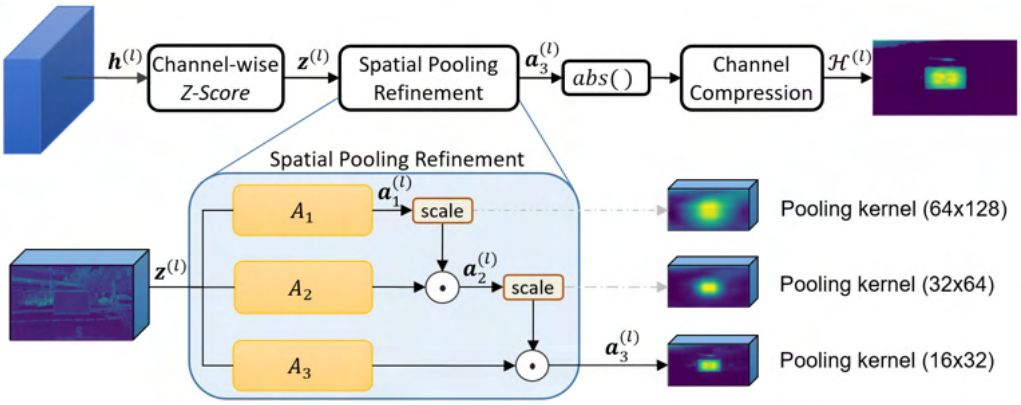


Figure 5.7: Over-activation pipeline performed by *Z-Mask* on a given layer with $m = 3$ Average-Pooling stages. The *scale* blocks refer to the ∞-norm used in Equation 5.4. Resizing operations are omitted in the figure.

adversarial patches, we are interested in highlighting *contiguous* over-activated portions of the image rather than spurious over-activated pixels (i.e., pixels whose neighbors have activation values close to $\mu^{(l)}$). To do that, the SPR implements a cascade filtering [176] that reduces the effects of spurious over-activated pixels. The process is iteratively refined: first larger kernels identify macro-regions that include over-activated contiguous pixels and then smaller kernels refine the analysis within such macro-regions. Finally, to obtain the desired heatmap $\mathcal{H}_l$ (of size $1 \times H^{\mathcal{R}} \times W^{\mathcal{R}}$), the absolute values of $\boldsymbol{a}_m^{(l)}$ are averaged across the channels. As shown in the experimental section, this process yields a heatmap of the over-activated region with sharper areas (Figure 5.10).

Fusion and Detection Mechanism

This section explains how the mask $\mathcal{M}_\delta$ is generated by merging the information of two sets of heatmaps, $\mathcal{S}$ and $\mathcal{D}$. The set $\mathcal{S}$ contains $N_\mathcal{S}$ heatmaps belonging to the selected shallow layers only, while $\mathcal{D}$ contains $N_\mathcal{D}$ heatmaps belonging to deeper layers and possibly to shallow layers. Leveraging these sets of heatmaps, we reduce the analysis to two aggregated heatmaps $\mathcal{H}_\mathcal{S} = \mathcal{F}(\mathcal{S})$ and $\mathcal{H}_\mathcal{D} = \mathcal{F}(\mathcal{D})$, where $\mathcal{F}(\cdot)$ is an operator that merges multiple heatmaps belonging to a given set. In practice, a pixel-wise *max* function is used for $\mathcal{F}(\cdot)$. $\mathcal{H}_\mathcal{S}$ and $\mathcal{H}_\mathcal{D}$ summarize the over-activation behavior at different depths in the model: $\mathcal{H}_\mathcal{S}$ represents the over-activated regions in the shallow layers, while $\mathcal{H}_\mathcal{D}$ takes into consideration also deep layers. The reason for using these two heatmaps emerged after a series of experimental observations. From a practical perspective, $\mathcal{H}_\mathcal{S}$ allows highlighting the over-activated portions of the image (i.e., the regions that may contain adversarial objects): it provides a high spatial accuracy, but a limited capability of discriminating adversarial and non-adversarial regions. Conversely, $\mathcal{H}_\mathcal{D}$ provides a high accuracy in identifying adversarial over-activations, but with a much lower spatial accuracy. In fact, experiments showed that over-activations coming from non-adversarial regions do not propagate their effect to deeper layers (a more detailed analysis of this effect is provided in the supplementary material). Hence, $\mathcal{H}_\mathcal{D}$ can be used to filter out the regions highlighted by $\mathcal{H}_\mathcal{S}$ that are not adversarial, yielding a more accurate heatmap.

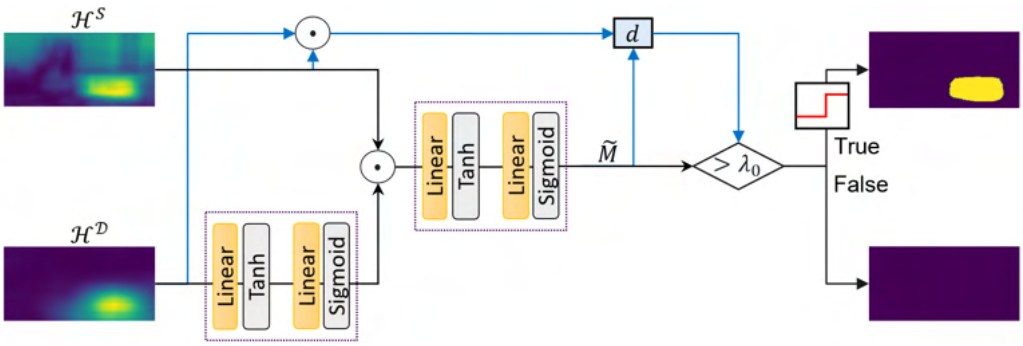


Figure 5.8: Fusion and Detection Block.

Figure 5.8 illustrates the operations performed by the Fusion and Detection Block. The merging process leverages two *soft-thresholding blocks*. The first block extracts a region of interest from $\mathcal{H}_\mathcal{D}$, which is then multiplied by $\mathcal{H}_\mathcal{S}$ to pose attention only to over-activated areas in the deeper layers. The second block extracts $\tilde{M}$, a soft version of the final mask with real pixel values in $[0, 1]$. Each *soft-thresholding block* consists of two sequential linear layers (both with 1-dimensional weight and bias), activated by a *tanh* and a *sigmoid* function, respectively.

Finally, to apply the masking only when an adversarial region is detected, we measure the over-activation as $d = \frac{\|\mathcal{H}^\mathcal{S} \odot \mathcal{H}^\mathcal{D} \odot \tilde{M}\|_1}{\|\tilde{M}\|_1}$ and compute the mask $\mathcal{M}$ as follows:

$$\mathcal{M}_\delta = \begin{cases} 1 - \hbar(\tilde{M}), & d > \lambda_0 \\ 1, & \text{otherwise}, \end{cases} \tag{5.5}$$

where $\hbar$ is the Heaviside function centered in 0.5 and λ_0 is a given threshold. The soft-thresholding parameters (eight in total) are fitted by supervised learning, while the threshold λ_0 is configured through an ROC analysis. The main motivation of using this module is its high deterministic behavior, since it mimics a soft-threshold operation in a differentiable manner. This allows testing against gradient-based defense-aware attacks and offers a transparent robustness by constraining an attacker to reduce the over-activation values to fool the defense (see experimental part).

5.3.2 Experiments

This section presents a set of experiments carried out on several convolutional models for object detection (OD) and semantic segmentation (SS) to evaluate the effectiveness of the proposed defense. All the experiments were implemented using PyTorch [131] on a server with 8 NVIDIA-A100 GPUs. For both SS and OD tasks, the effectiveness of an adversarial attack was measured by evaluating the drop of the model performance with a task-dependent metric. For SS models, the mIoU was used on the subset of pixels not belonging to the applied patch, as done by [24]. For OD models, the performance was measured by the COCO mAP.

Datasets. The Cityscapes dataset [156] and COCO 2017 dataset [166] were considered for these tests. Furthemore, to assess the proposed approach on real-world scenarios, we

considered APRICOT [58], which is a COCO-like dataset including more than 1000 images, each containing a physical adversarial patch for one between Faster R-CNN, RetinaNet, and SSD. Concerning the models, we tested the defense mechanisms for ICNet, BiseNet and DDRNet for semantic segmentation, while for object detection we considered R-CNN, RetinaNet and SSD.

Attack and defense strategies. Different attack methodologies were used to craft adversarial patches. For SS models, we leveraged the untargeted attack pipeline used in [24], while, for OD models, we performed an untargeted attack on the classes, similarly to [177]. The patches from APRICOT dataset rely on a false-detection attack [58].

Concerning defense strategies, we compared *Z-Mask* against different approaches for both adversarial pixel masking and detection. For the masking task, we re-implemented the Local Gradient Smoothing method (LGS) [103] and MaskNet [106], both with the original settings described by the authors. For the adversarial detection, we considered for comparisons FPDA [24] and HN [105].

Activation-aware patch optimization. We crafted adversarial patches while controlling the over-activation to understand its relation with the induced adversarial effect, as well train the defense to properly scale into real-world scenarios. To do that, we proposed the following optimization:

$$\hat{\boldsymbol{\delta}}_\beta = \underset{\boldsymbol{\delta}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\boldsymbol{\Gamma}}\big[(1-\beta)\cdot\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x},\boldsymbol{\delta})) + \beta\cdot\mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x},\boldsymbol{\delta})),\mathbf{y})\big], \tag{5.6}$$

where $\beta \in [0,1]$ is a control parameter and $\mathcal{L}_{OZ}$ is a loss function that measures the magnitude of over-activation of internal layers (details are available in the supplementary material). The rationale behind this optimization problem is that a low value of β reduces the importance assigned to the adversarial effect, while forcing the attack to generate less over-activation in the internal layers, hence simulating real-world patches. Figure 5.9 illustrates the over-activation of these patches (computed with the SPR) both in shallow and deep layers, remarking the relation with the induced adversarial effect. Furthermore, Figure 5.12 (discussed later) provides a measure of the adversarial effect as a function of the parameter β.

Details on the over-activation loss. Formally, we defined the loss function $\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x},\boldsymbol{\delta}))$ used to control the over-activation, as follows:

$$\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x},\boldsymbol{\delta})) = \frac{1}{|L_A|\cdot|\bar{M}|}\sum_{l\in L_A}\frac{1}{C^l}\sum_c\left|\sum_{i,j}(\boldsymbol{z}^{(l)}_{c,i,j}\odot\bar{(}1-\mathcal{M}_\delta))\right|, \tag{5.7}$$

where $\boldsymbol{z}^{(l)}$ is the channel-wise Z-score obtained through the forward pass of $g_\gamma(\mathbf{x},\boldsymbol{\delta})$ in f and l is a network layer selected between the set of layers under analysis L_A, each having a channel dimension C^l. In short, $\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x},\boldsymbol{\delta}))$ computes the average over-activation in each layer $l \in L_A$ corresponding only to those neurons that are spatially associated to the adversarial patch δ.

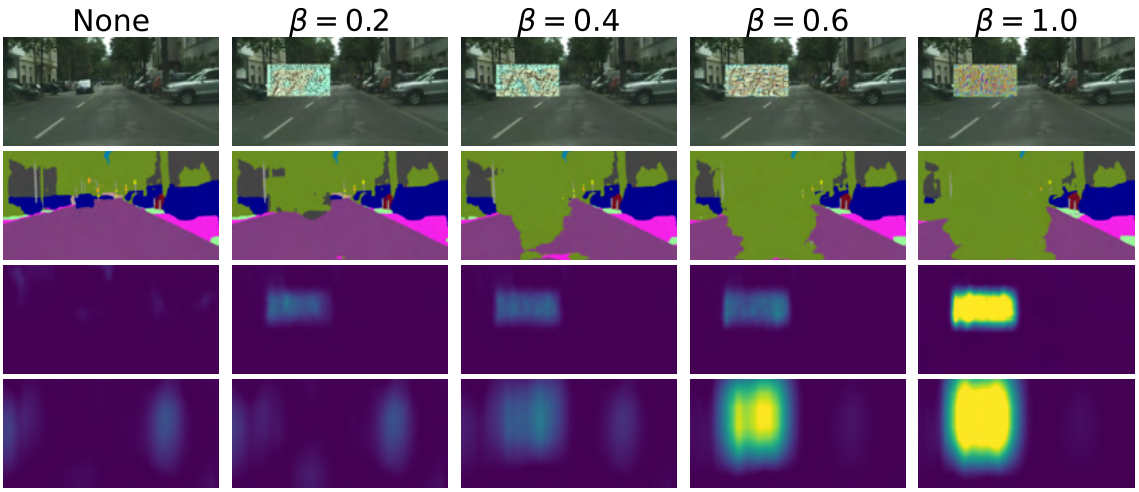


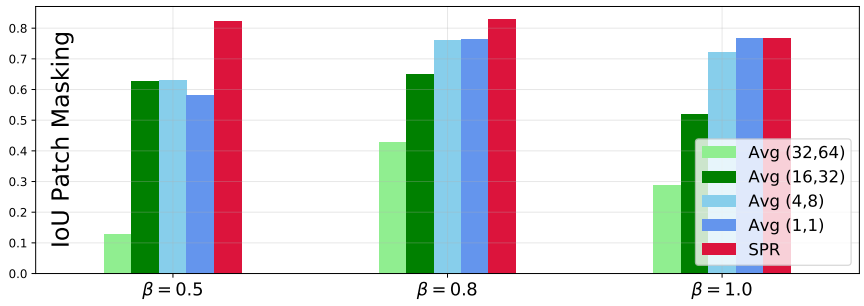
Figure 5.9: Visualization of the predictions and SPR heatmaps obtained from several β activation-aware patches. The rows report x, $f(x)$, $\mathcal{H}^{\mathcal{S}}$ and $\mathcal{H}^{\mathcal{D}}$, respectively.

***Z-Mask* settings and training.** For semantic segmentation models, the heatmaps in $\mathcal{S}$ were generated with a SPR composed of four pooling operations, with kernel sizes $k_1 = (64, 128)$, $k_2 = (32, 64)$, $k_3 = (16, 32)$, $k_4 = (8, 16)$. Instead, the heatmaps in $\mathcal{D}$ were generated using two pooling operations with kernel sizes $k_1 = (64, 128)$, $k_2 = (32, 64)$. After each pooling operation, the heatmaps were resized to $(H^{\mathcal{R}} \times W^{\mathcal{R}}) = (150 \times 300)$. Please note that all the resulting heatmaps have a 1:2 aspect ratio, keeping the same aspect ratio of the input images. For OD models, the SPR used $k_1 = (40, 40)$, $k_2 = (25, 25)$, $k_3 = (10, 10)$ to build $\mathcal{S}$, and $k_1 = (80, 80)$, $k_2 = (40, 40)$ to build $\mathcal{D}$. The resizing dimension was set to (400×500). For all the tests, pooling operations were applied with stride 1. These kernel settings were motivated by preliminary tests performed to analyse the internal activations. To illustrate the benefits of the SPR, Figure 5.10 provides the results of ablation studies by comparing the performance of the Fusion and Detection block with different pooling settings and patches crafted with different β. The SPR block always achieves a better IoU Patch Masking, which is computed as the IoU between the predicted mask and its ground truth.

For completness, we summarize all the layers and the parameters addressed in the experiments in Table 5.2, which also reports further models that will be included in the evaluation performed in the Section 5.4 through CARLA-GEAR.

The parameters of the *soft-thresholding* operations inside the Fusion and Detection block were trained in a supervised fashion by considering a set of patches crafted with Equation 5.6 and minimizing the pixel-wise binary cross-entropy loss $\mathcal{L}_{\text{BCE}}(\tilde{M}, \bar{M})$, where $\bar{M}$ is the ground-truth binary mask. To this end, we collected a set of adversarial patches $\Delta = \{\hat{\boldsymbol{\delta}}_\beta : \beta \in [\beta_0, 1]\}$, where we set $\beta_0 = 0.5$ to avoid generating patches with scarce adversarial effect. These patches were used to craft the set $\tilde{\boldsymbol{X}}$, which was obtained by adding the patches in Δ to each image of $\boldsymbol{X}$. Set $\tilde{\boldsymbol{X}}$ was used to train the Fusion and Detection Block and make it robust to a wide spectrum of over-activations.

In our tests, $\boldsymbol{X}$ contained 500 images randomly sampled from the original training dataset. The ADAM optimizer [127] was used for this purpose, with a learning-rate of



(a) IoU Patch Masking comparison

(b) No Avg (1,1) (c) Avg (16,32) (d) SPR

Figure 5.10: Ablation studies on the masking accuracy with different pooling strategies on 100 images of the Cityscapes validation set and BiseNet (a). Bottom figures are the predicted masks of a same input, using a patch with $\beta = 0.5$.

0.01 and training for 15 epochs. The channel-wise std and mean of each selected layer was computed on a different subset of the training set containing 500 clean (i.e., non-patched) images. The detection threshold λ_0 was deduced after the soft-thresholding training as the *cut-off* threshold of the ROC curve. The ROC was generated by computing the measure d on each input of a dataset, including the clean set $\boldsymbol{X}$ and the patched set $\tilde{\boldsymbol{X}}$, labeled as negative and positive samples, respectively.

Net	Patch	Defense Method (mAP Val)			
		Z-Mask	MaskNet	LGS	None
FR-CNN	None	**0.357**	0.353	0.350	0.357
	Rand	0.301	0.295	**0.320**	0.308
	S	0.335	0.333	**0.354**	0.337
	M	**0.302**	0.289	0.246	0.140
	L	**0.300**	0.289	0.244	0.164
SSD	None	**0.253**	0.180	0.243	0.264
	Rand	**0.208**	0.132	0.198	0.215
	S	**0.237**	0.159	0.233	0.245
	M	**0.202**	0.125	0.144	0.065
	L	**0.205**	0.113	0.163	0.072
RetinaNet	None	**0.355**	0.269	0.337	0.359
	Rand	0.305	0.227	**0.312**	0.308
	S	**0.339**	0.245	0.339	0.335
	M	**0.326**	0.222	0.306	0.304
	L	**0.305**	0.212	0.297	0.283

(a)

Net	Patch	Defense Method (mIoU Val)			
		Z-Mask	MaskNet	LGS	None
DDRNet	None	**0.778**	0.739	0.777	0.778
	Rand	0.731	0.710	**0.769**	0.761
	S	**0.741**	0.701	**0.741**	0.702
	M	**0.723**	0.699	0.719	0.663
	L	**0.691**	0.689	0.642	0.532
BiseNet	None	0.684	0.622	**0.685**	0.687
	Rand	0.650	0.569	**0.668**	0.653
	S	**0.663**	0.560	0.522	0.475
	M	**0.653**	0.550	0.413	0.323
	L	**0.621**	0.535	0.320	0.220
ICNet	None	**0.785**	0.783	0.782	0.785
	Rand	**0.768**	0.736	0.764	0.746
	S	**0.748**	0.737	0.657	0.625
	M	**0.729**	0.718	0.593	0.549
	L	**0.747**	0.725	0.528	0.430

(b)

Table 5.3: Robustness performance evaluated for different patch sizes for Object detection on COCO (a) and Semantic Segmentation on Cityscapes (b).

Evaluation for Digital Attacks

Masking performance. The benefits of the proposed defense mechanism were evaluated by attacking the validation sets of COCO and Cityscapes with different adversarial

Net	Pooling param	Name	Resize factor (H,W)
DDRNet	(64,128), (32,64), (16,32), (8,16)	layer5	(150,300)
ICNet	(64,128), (32,64), (16,32), (8,16)	res-block5, convbnrelu1-2	(150,300)
BiseNet	(64,128), (32,64), (16,32), (8,16)	context_path_2	(150,300)
FRCNN	(40,40),(25,25), (10,10)	backbone.body.conv1 backbone.body.layer1[0].conv1, backbone.body.relu	(400,400)
RetinaNet	(40,40),(25,25), (10,10)	backbone.body.conv1 backbone.body.layer1[0].conv1, backbone.body.relu	(400,400)
Ababins	(32,128), (16,64), (8,32)	encoder.original_model.blocks[1][0] encoder.original_model.blocks[0][2]	(100,400)
GLPDepth	(8,8), (4,4)	encoder.patch_embed1.proj	(100,400)
Stereo-FRCNN	(16, 32)	RCNN_smooth1	(100,400)

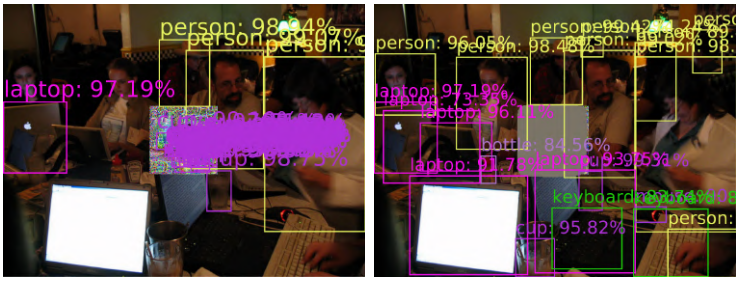
Net	Pooling param	Name	Resize factor (H,W)
DDRNet	(64,128), (32, 64)	layer5	(150,300)
ICNet	(64,128), (32, 64)	res-block5, conv5-4-k1, convbnrelu1-2	(150,300)
BiseNet	(64,128), (32, 64)	context_path_2	(150,300)
FRCNN	(80,80), (40,40)	backbone.body.conv1 backbone.body.layer1[0].conv1, backbone.body.layer1[1].conv1	(400,400)
RetinaNet	(80,80), (40,40)	backbone.body.conv1 backbone.body.layer1[0].conv1, backbone.body.layer4	(400,400)
Ababins	(32,128), (16, 64)	encoder.original_model.blocks[1][0]	(100,400)
GLPDepth	(16, 16), (8,8)	encoder.patch_embed1.proj	(100,400)
Stereo-FRCNN	(16,32), (8,16)	RCNN_smooth1	(100,400)

Table 5.2: Pooling settings and selected shallow (above) and deep (below) layers for Z-Mask. For reproducibility, the layer names refer to github.com/retis-ai/CARLA-GeAR

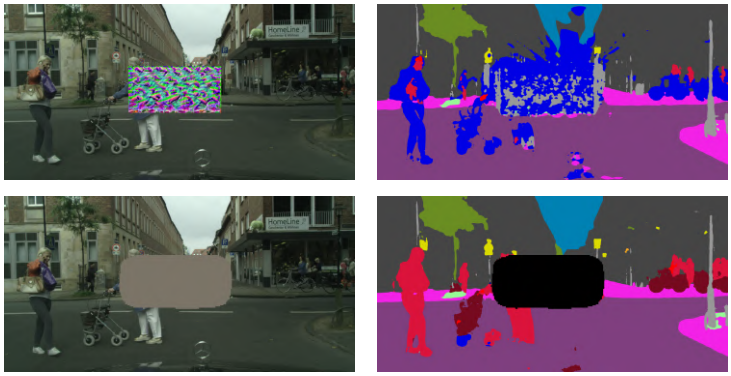
patch sizes. For Cityscapes, we used patches with size 600x300 (L), 400x200 (M) and 300x150 (S) pixels, whereas for COCO, due to the different image aspect ratio, we used 200x200 (L), 150x150 (M), and 100x100 (S). Also, an L-size random patch was evaluated to test the case in which a portion of the image is occluded without the intent of generating an adversarial attack.

As shown in Table 5.3, *Z-Mask* outperformed the other defense strategies, achieving scores similar to the random case, when tested against adversarial attacks, and close to the original model without applying patches, meaning that does not affect the nominal model performance. Figure 5.11 illustrates the benefits of *Z-Mask*: attacked areas are identified and covered without affecting other portions.

Detection performance. All the adversarial patches evaluated in Table 5.3 were perfectly detected by both *Z-Mask*, HN, and FPDA. To better assess the performance of these adversarial detection methods, we used the optimization described in Equation (5.6) to generate a set of patches with a wider range of over-activation values, selecting the values of $\beta \in \{0.1, 0.2, ..., 0.9, 1.0\}$. Please note that $\beta = 1.0$ corresponds to a regular adversarial attack, while lower β values decrease the importance of adversarial effect to reduce the magnitude of over-activation. An L-sized patch was generated for each β. Figure 5.12 shows the detection and masking accuracy against this set of patches as a function of β for DDRNet. The top part of the figure shows the detection accuracy, evaluated using the AUC of ROC on a dataset, including both the clean and the attacked validation set



(a) Faster R-CNN - COCO dataset



(b) BiseNet - Cityscapes dataset

Figure 5.11: *Z-Mask* effects (comparison w/ and w/o defense)

(as negative and positive samples, respectively). Note that *Z-Mask* achieved better results than the other adversarial patch detectors, providing good detection performance also to patches that do not retain much adversarial effect. The bottom part of the Figure 5.12 reports the performance of *Z-Mask*, MaskNet, LGS, and the original model (without defense). Again, our method achieved higher mIoU among almost all the β values.

Evaluation for Physical Attacks

The masking and detection performance of *Z-mask* was evaluated in real-world scenarios with images containing printed adversarial patches. For this test, we adopted the same *Z-mask* settings and parameters used for digital attacks on COCO, which generalize well also for real-world patches. The detection performance was assessed with the APRICOT dataset, as positive samples, and 1000 images of the COCO validation set, as negative samples. Figure 5.13 (a) reports the corresponding ROC, where *Z-Mask* obtained the best AUC with respect to FPDA and HN on both RetinaNet and Faster R-CNN. The analysis on SSD was omitted since the large rescaling factor on the input image required by the pretrained network neutralize the adversarial effect of APRICOT patches. Figure 5.13 (b) shows the effect of *Z-Mask* on a sample of APRICOT.

5.3.3 Defense-Aware Attacks

Since the *Z-Mask* pipeline is fully differentiable up to $\tilde{M}$, an attacker might exploit that knowledge to craft defense-aware attacks, i.e., optimize patches that are adversarial for

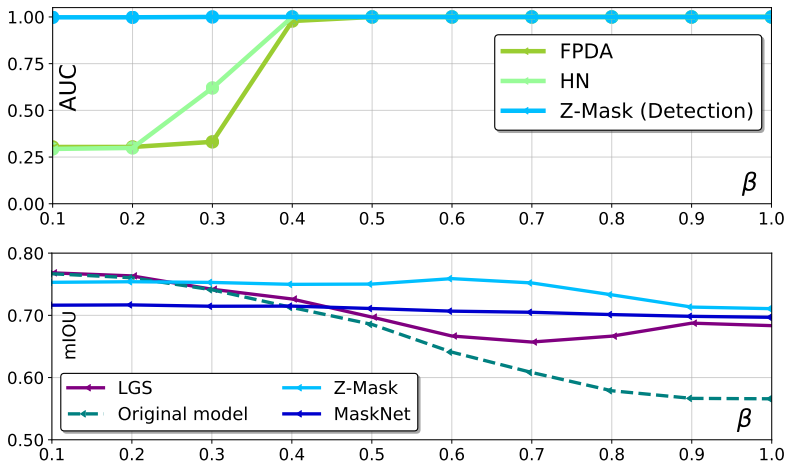


Figure 5.12: Comparison of the Detection accuracy (top plot) and task mIoU performance (bottom plot) using DDRNet on the validation set of Cityscapes.

the model and the defense together. To this end, we propose two defense-aware attacks.

The first attack, denoted as *Mask-Attack*, is designed to induce errors in the mask output to yield an incorrect input masking operation. This would allow the adversarial patch to pass without being masked or induce additional occlusion in the image. This attack is obtained by solving the following problem with $\alpha \in \{0, 0.1, 0.2, \ldots, 1.0\}$:

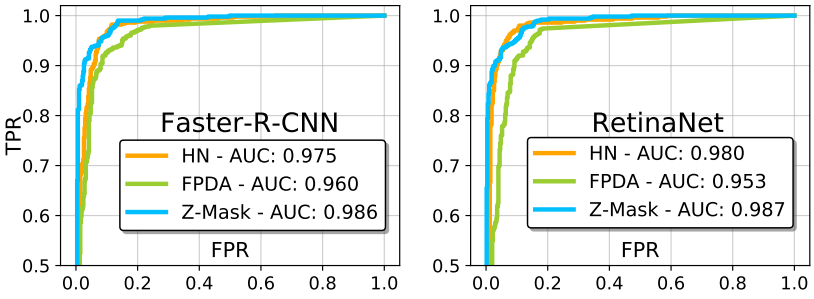
$$\hat{\boldsymbol{\delta}}_\alpha = \underset{\boldsymbol{\delta}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\boldsymbol{\Gamma}}\left[(1-\alpha)\cdot(-\mathcal{L}_{BCE}(\tilde{M},\bar{M})) + \alpha\cdot\mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x},\boldsymbol{\delta})),\mathbf{y})\right]. \tag{5.8}$$

Recall that $\mathcal{L}_{\text{BCE}}(\tilde{M},\bar{M})$ is the pixel-wise binary cross-entropy loss between the defense mask $\tilde{M}$ and the ground-truth patch mask $\bar{M}$ (which is known).

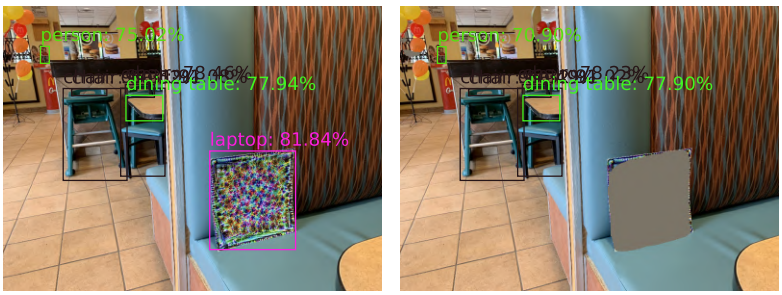
A second attack formulation, denoted as *Flag-Attack*, targets the detection flag aiming at causing false negatives in the detector. This attack is performed by replacing $\mathcal{L}_{\text{BCE}}(\tilde{M},\bar{M})$ with $\mathcal{L}_{\text{BCE}}(\mathit{Sigmoid}(d-\lambda_0),1)$. This is done to force $d < \lambda_0$ in the optimization, hence resulting in a mask $M(\mathbf{x}) = 1$.

Figure 5.14 shows the results of *Z-Mask* against these attacks on DDRNet (results on other networks in the supplementary material). A mask defense-aware attack was also tested on *MaskNet* to provide a comparison, while the results of LGS are not reported, since other works already addressed its weaknesses under defense-aware attacks [106].

Note that, even exploiting the knowledge of the defense, the proposed attacks were not able to reduce the performance of *Z-Mask* more than what obtained for the digital evaluation, as reported in Table 5.3. Indeed, observe from Figure 5.14 that, when *Z-Mask* does not detect the attack (TPR=0), the attack is not effective (maximum mIoU). Practically speaking, the robustness of *Z-Mask* comes from the fact that it directly exploits the over-activation values. In fact, recalling that physical attacks are strictly related to over-activations, the attacker is required to reduce their magnitude to bypass the defense, thus inevitably yielding less effective attacks. Conversely, for MaskNet, certain values of α induce larger performance degradation.



(a)



(b)

Figure 5.13: (a) ROC analysis performed on the dataset including APRICOT images and 1000 COCO images. (b) The effect of *Z-Mask* on an APRICOT image.

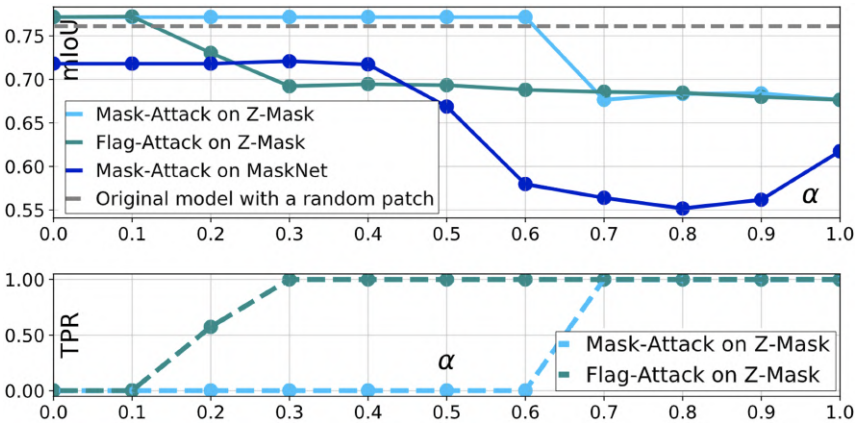


Figure 5.14: Evaluation and comparison of defense benefits (mIoU) and detection performance (TPR) against defense-aware attacks as a function of α. The results refer to DDRNet evaluated on the validation set of Cityscapes.

5.4 Defenses Evaluation on CARLA

This section presents an extensive comparison of different defense methods when applied to the dataset generated by CARLA-GEAR for each of the four computer vision tasks considered in this proposed tool. The settings used for each defense have been detailed in the above sections, see Tables 5.2 and 5.1.

More specifically, the training of Z-Mask was performed using the Adam optimizer with $lr = 0.05$ for 1000 epochs and 50 as the batch size. The training steps were performed through a positive dataset (containing physical patches in the CARLA scenario) and a negative dataset (without patches). Each of these datasets consist of 100 images selected from generated dataset splits. The negative set is the original *no_billboard* dataset (see Section 4.3), while the positive set is composed of 100 images randomly extracted from extra *test_net* splits (from billboard 1 to 9), which were generated using a different seeds with respect the original ones. As carried out in the ZMask section and [107], the detection threshold was tuned with the two previous datasets by computing the cut-off point obtained from the ROC curve. The *mean* and *std*, which are necessary to compute the z-score, are extracted from the layers activations of inputs belonging to the previous negative dataset.

Adversarial Detection. Table 5.4 compares the performance of Z-Mask, FPDA, and HyperNeuron in terms of AUROC. Please note that the performance of adversarial detection algorithms is evaluated with the area under the receiver operating characteristic (AUROC). Each ROC is computed on two "twin" datasets, one including the adversarial patches and the other containing the same images, but without any patch.

All the methods present good performance for most of the networks and scenarios, with Z-Mask being the most consistent detection method overall. Please note that, in some scenarios (*bilboard08, bilboard09 and truck*), all the tested algorithms surprisingly get to low detection performance. Such values clearly differ from the detection performance expected by the tested algorithms, showing that such scenarios depict critical situations that question the reliability of the addressed defenses.

Adversarial Masking. Table 5.5 reports the performance of each defense method considered for each task. Notably, there are a few attack situations that do not induce a large adversarial effect with respect to the addressed metrics (e.g., Billboard01, 02, 03, 05, 09, and Truck on DDRNet), where the defense is actually harming the performance of the network. Nevertheless, it is worth noting how the mAP metric for 2D object detection indicates a consistent low adversarial effect and bad defense performance, while in reality the effect of the adversarial patch is present (see Figure 5.15 for an example) and the defense is masking the patch with fair accuracy. As it is also remarked in Section 4.4, this observation highlights the need for a more accurate metric to evaluate the effect of these patches (and related defenses) and highlights one of the limitations to the realization of a proper real-world robustness benchmark.

Regarding the achieved results, also these tests on CARLA reported different outcomes among all the addressed scenarios and defense mechanisms, thus helping understand potential critical situations uncovered during the design of the defense. Figure 5.15 provides

some illustrations of the masking effect of Z-Mask among some generated images for semantic segmentation and object detection.

		SS		2DOD		Depth		3DOD
	Def	DDRNet	BiseNet	FRCNN	RetinaNet	AdaBins	GLPDepth	Stereo RCNN
TruckBillboardD01	ZM	0.99	0.95	0.50	0.49	0.27	0.00	0.82
	FPDA	1.00	0.97	0.53	0.49	0.20	0.49	0.50
	HN	1.00	1.00	0.52	0.53	0.09	0.12	0.89
TruckBillboardD02	ZM	1.00	1.00	1.00	1.00	0.89	0.26	0.92
	FPDA	1.00	1.00	0.97	0.66	0.90	0.00	0.50
	HN	1.00	1.00	0.97	0.78	0.98	0.51	0.96
TruckBillboardD03	ZM	0.98	0.90	0.99	0.99	0.99	0.33	0.74
	FPDA	0.86	0.92	0.64	0.85	0.89	0.43	0.50
	HN	0.95	0.87	0.88	0.82	0.97	0.23	0.75
TruckBillboardD04	ZM	1.00	1.00	0.95	0.96	0.97	0.39	0.97
	FPDA	0.99	1.00	0.58	0.56	0.92	0.49	0.50
	HN	1.00	1.00	0.70	0.66	0.97	0.40	0.95
TruckBillboardD05	ZM	1.00	1.00	0.86	0.90	0.31	0.05	0.92
	FPDA	0.99	1.00	0.62	0.64	0.30	0.50	0.78
	HN	0.99	1.00	0.74	0.72	0.27	0.39	0.95
TruckBillboardD06	ZM	1.00	1.00	1.00	1.00	0.96	0.59	0.98
	FPDA	1.00	1.00	0.87	0.61	0.86	0.07	0.50
	HN	1.00	1.00	0.85	0.85	0.88	0.94	0.99
TruckBillboardD07	ZM	1.00	0.60	1.00	1.00	1.00	0.61	0.93
	FPDA	1.00	1.00	0.89	0.60	0.97	0.49	0.50
	HN	1.00	1.00	0.96	0.73	0.98	0.27	0.92
TruckBillboardD08	ZM	1.00	0.97	0.66	0.67	0.29	0.00	0.98
	FPDA	0.87	1.00	0.58	0.55	0.23	0.50	0.50
	HN	0.93	1.00	0.59	0.67	0.21	0.10	0.99
TruckBillboardD09	ZM	0.68	0.65	0.84	0.86	0.91	0.48	0.59
	FPDA	0.62	1.00	0.58	0.46	0.90	0.51	0.50
	HN	0.61	1.00	0.76	0.51	0.99	0.51	0.65
TruckBillboardE01	ZM	0.87	0.68	0.58	0.56	0.60	0.42	0.63
	FPDA	0.87	0.72	0.52	0.54	0.52	0.46	0.50
	HN	0.86	0.77	0.53	0.52	0.58	0.43	0.62

Table 5.4: Comparison of detection performance (measured in AUROC) across various defense strategies (Z-Mask (ZM), FPDA, and HyperNeuron (HN)), computed for each attack situation, model, and task. The evaluated tasks include Semantic Segmentation (SS), 2D and 3D Object Detection (2D/3D OD), and Depth Estimation.

		SS (mIOU)				2DOD (mAP)				Depth (RMSE)				3DOD (mAP)	
	Def	DDRNet		BiseNet		FRCNN		RetinaNet		AdaBins		GLPDepth		Stereo RCNN	
Billboard01	-	**33.04**	0.65	**28.35**	-10.20	**19.89**	-0.59	**20.12**	-2.66	**15.51**	6.00	**13.52**	3.22	**51.43**	-2.84
	ZM	0.00	-2.31	-0.15	-5.32	0.02	-0.49	-0.25	-2.95	-0.36	1.72	-0.10	3.22	-0.45	-0.86
	LGS	-0.07	0.45	0.11	-8.63	0.02	-1.38	-0.25	-4.41	-0.36	5.94	-0.10	3.12	-0.45	-3.09
Billboard02	-	**34.11**	-1.74	**29.66**	-4.32	**19.66**	-0.93	**19.65**	-0.45	**16.04**	4.93	**12.33**	5.61	**38.29**	3.70
	ZM	-0.04	-1.76	-0.21	-2.35	0.06	-1.32	-0.17	-1.12	-0.02	0.07	-0.03	3.04	-0.41	0.85
	LGS	-0.20	-3.51	0.07	-5.09	0.06	-2.90	-0.17	-2.63	-0.02	3.54	-0.03	4.51	-0.41	6.23
Billboard03	-	**34.39**	-0.60	**28.88**	-2.81	**19.30**	-0.89	**18.92**	-0.69	**14.23**	3.22	**12.05**	3.33	**77.54**	-11.56
	ZM	0.00	-0.60	0.00	-2.81	0.00	-0.77	0.00	-0.70	-0.00	1.03	0.00	3.33	-8.86	-16.59
	LGS	1.60	1.01	1.92	-4.09	0.00	-2.27	0.00	-1.60	-0.00	4.24	0.00	3.11	-8.86	-27.77
Billboard04	-	**34.22**	-6.53	**24.71**	-6.40	**25.01**	1.80	**27.83**	0.99	**18.62**	6.11	**15.73**	5.32	**47.75**	0.86
	ZM	0.00	-5.20	0.00	-3.01	0.00	1.79	0.00	0.92	-0.01	0.71	0.00	5.32	-7.97	0.03
	LGS	-0.14	-8.30	-0.71	-6.43	0.00	0.17	0.00	-2.32	-0.01	5.80	0.00	5.23	-7.97	1.11
Billboard05	-	**31.01**	-1.01	**25.10**	-5.06	**22.45**	-0.21	**22.17**	-0.53	**11.75**	10.41	**12.54**	7.01	**46.14**	-4.10
	ZM	0.00	-1.56	0.00	-1.70	0.00	-0.21	0.00	-0.53	0.36	4.67	0.03	7.01	-8.30	-9.69
	LGS	-0.20	-0.87	-0.24	-4.04	0.00	-1.01	0.00	-1.55	0.36	9.74	0.03	6.39	-8.30	-2.17
Billboard06	-	**33.26**	-2.31	**26.51**	-9.50	**24.04**	-0.62	**22.86**	-0.23	**11.47**	9.71	**8.11**	8.08	**64.95**	-27.01
	ZM	0.00	-4.12	0.00	-5.87	0.01	-1.22	-0.09	-0.11	0.00	4.27	0.04	5.40	-0.08	-11.60
	LGS	-2.07	-4.06	-0.40	-6.65	0.01	-1.98	-0.09	-2.03	0.00	10.28	0.04	7.73	-0.08	-25.48
Billboard07	-	**30.73**	-12.69	**20.56**	-7.00	**12.76**	-1.01	**9.43**	-0.15	**13.35**	13.65	**13.35**	8.84	**45.41**	-33.17
	ZM	0.00	-12.69	-0.91	-4.84	0.00	-0.31	0.00	-0.17	0.00	6.12	-0.08	8.68	-0.33	-30.72
	LGS	0.87	-11.97	-0.43	-6.01	0.00	-0.95	0.00	-0.51	0.00	12.73	-0.08	8.54	-0.33	-32.83
Billboard08	-	**31.86**	-3.03	**24.46**	-2.93	**37.18**	0.78	**38.59**	-0.55	**12.14**	8.48	**12.31**	6.19	**43.15**	-8.01
	ZM	-0.11	-2.38	-1.02	-3.00	-0.02	0.31	-0.20	-0.29	0.55	1.30	0.29	6.19	-8.49	-8.93
	LGS	-3.28	-1.22	-0.00	-1.60	-0.02	-0.26	-0.20	-0.74	0.55	9.21	0.29	6.04	-8.49	-8.58
Billboard09	-	**37.41**	-1.11	**23.68**	-6.75	**14.29**	-0.32	**14.53**	-3.39	**19.81**	1.74	**15.84**	2.61	**61.73**	-2.55
	ZM	0.00	-1.11	-4.76	-5.89	0.00	-0.22	-0.00	-1.97	-0.03	0.47	-0.40	2.15	-0.94	-1.50
	LGS	-0.18	-1.25	0.19	-5.83	0.00	-0.83	-0.00	-3.72	-0.03	2.17	-0.40	2.34	-0.94	-4.17
Truck	-	**27.89**	0.09	**22.75**	-4.51	**10.22**	0.29	**12.68**	0.49	**14.08**	1.21	**11.97**	0.15	**42.88**	-0.11
	ZM	-0.35	-0.20	0.46	-0.22	-0.08	-0.22	-0.00	0.47	0.01	0.42	0.08	0.11	-0.40	-8.23
	LGS	-0.44	-0.39	-0.56	-4.62	-0.08	0.05	-0.00	-0.84	0.01	0.77	0.08	-0.18	-0.40	-9.12

Table 5.5: Performance of Z-Mask and LGS for each attack situation, for each task. White columns shows the results in the case without patch, whereas greyed columns shows the performance in the adversarial case. All the cells report the relative performance with respect to the undefended, non-adversarial case (in bold). The value is colored in red [green] if the defense is worsening [improving] the performance of the model, or blue if there is no effect.

(a)

(b)

Figure 5.15: Illustration of the effectiveness of adversarial billboards (column one and two) and of Z-Mask [178] (column three and four) for Semantic Segmentation (a) and Object Detection (b). Images are extracted from the generated datasets.

5.5 Attention-based Real-Time Attack Tracking

Although recent adversarial masking strategies have shown a promising performance to face real-world attacks, even on complex scenarios, previous work mainly focused on single-image (i.e., single-frame) cases and without paying particular attention at the computational cost of the proposed defense method, resulting in more inference passes or additional expensive neural models.

As discussed in Section 2.4, recent techniques as [60, 106, 59] introduced a secondary encoder-decoder model to compute the adversrial mask, which is then applied to the image before it is passed to the DNN. These approaches significantly increase the overall computational cost for each input (even for the non-attacked ones). In contrast, Z-Mask addresses a different paradigm, based on internal analysis of DNN layers to identify anomalous over-activations at run-time. However, as shown in Section 5.3, it requires two inference (i.e., forward) passes as the attacks are detected when analyzing deep layers of the model, where the effects of the attacks are not anymore recoverable. The second pass is hence needed to process the input image with the pixel mask applied.

Overall, although many approaches presented in previous works achieve promising performance, these limitations pose challenges for implementing state-of-the-art methods in real-time, safety-critical applications.

To face these challenges, in this third defense work [179] we delve deep into understanding the over-activation phenomenon by observing the presence of specific over-activated channels even in the first layers of DNNs, which are predominantly targeted by the real-world attack for propagating adversarial effects. We systematically identified this attack pattern through channel-wise weights, denoted as *adversarial trace*, that enable a significantly faster and more accurate identification of attacks by means of a proper attention strategy designed in this work. This allows for the immediate removal of adversarial features before their spatial propagation in the deep layers, hence detecting and masking attacks in a single inference pass.

After presenting the results of our analysis and providing insights into the nature of the adversarial trace, we propose a defense algorithm for multi-frame vision applications named *Adversarial-Channel Attention Tracing* (ACAT). To enable the efficient tracing of adversarial physical objects in a video stream, ACAT requires to know a starting spatial position of the objects, which can be extracted using a single inference pass of state-of-the-art single-frame defense methods. As witnessed by the experimental results reported in the paper, this improves both efficiency in terms of running times and computing load, as well as the attack detection effectiveness. The proposed approach is illustrated in Figure 5.16.

5.5.1 Adversarial-Channel Attention

Inspired by prior research on real-world attacks and internal over-activation analysis for neural networks, we observed that attacks can be detected by even analyzing shallow network layers only (as opposed to deep layers as done by previous work). In the following, we provide insights into the existence of over-activation patterns within shallow layers and then address the definition of *adversarial trace*, which is later used to enable the implementation of an adversarial attention mechanism for multi-frame scenarios.

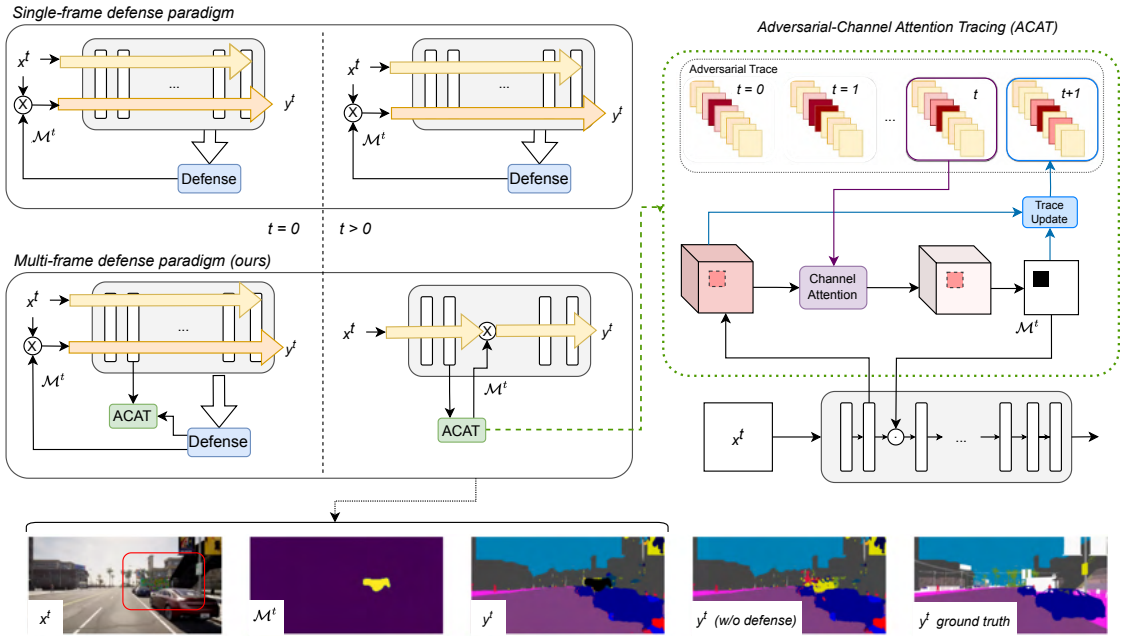


Figure 5.16: Schematic and simplified overview of the proposed multi-frame defense paradigm compared to state-of-the-art, single-frame defense paradigms. At frame $t = 0$, with a first inference pass (yellow arrow in the figure), a single-frame defense mechanism extracts a mask to inhibit the detected attack. Another inference pass is required at frame $t = 0$ to apply the defense mask (orange arrow in the figure). With the proposed approach, the mask is used to implement pattern analysis in the shallow layers for the next frames $t > 0$, which is the core task performed by ACAT. This allows extracting an *adversarial trace* that allows for a quick identification of the shape of the adversarial object, hence efficiently generating and applying defense masks (right side of the figure). At the bottom, we show illustrations of the defense mechanism in a simulated attacked Carla driving scenario [165] with the BiseNet model [2], where the adversarial object is highlighted in the red area. For completeness, we also report the output of the same frame without any defense mechanism and the ground truth.

Single-frame Adversarial Attention Analysis

We start by providing insights that link abnormal activations induced by adversarial objects with a particular pattern of channels in the shallow layers.

Observation 1 *An adversarial object δ is designed to minimize a specific adversarial loss function by influencing certain network features (see Eq.(2.3)). As for instance shown in Figure 5.17(a), we argue that in any layer L_l with activations $\boldsymbol{h}_l$, there exists a subset of channels targeted by the adversarial object. The channels can be identified by leveraging some channel weights $\boldsymbol{\sigma} \in [0,1]^{C^l}$ that, if applied to $\boldsymbol{h}_l$, amplify the adversarial features, i.e.,*

$$\mathcal{L}_{Adv}(f^{l \to N_L}(\boldsymbol{\sigma} \cdot \boldsymbol{h}_l), \mathbf{y}_{Adv}) \;\leq\; \mathcal{L}_{Adv}(f^{l \to N_L}(\boldsymbol{h}_l), \mathbf{y}_{Adv}).$$

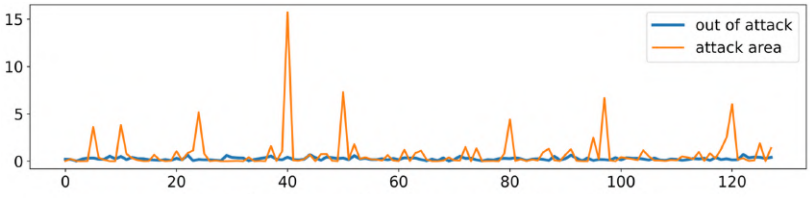
Observation 2 *As known from previous work [116], the adversarial features introduced by physical attacks are characterized by over-activations. Therefore, from the perspective of an internal over-activation analysis (as done with the steps performed by ZMask 5.3),*

a proper definition of $\boldsymbol{\sigma}$ also allows focusing on the channels that are more subject to over-activation within the attacked area. This results in a better separation of the attacked area from all the others in the heatmap $\mathcal{H}$, which can be interpreted as a more accurate computation of defense masks. Formally, this observation can be formulated by means of the intersection-over-union (IoU) [180], which provides the amount of overlap between two masks. If $\mathcal{M}_{GT}$ is the ground-truth mask capable of perfectly masking the attacked area in the input image, this observation can be written as a function of the IoU between the predicted attacked area and the ground-truth mask, i.e.,

$$IoU\left(\Lambda^{\xi'}(\boldsymbol{\sigma}\cdot\boldsymbol{h}_l),\ \mathcal{M}_{GT}\right) \geq IoU\left(\Lambda^{\xi''}(\boldsymbol{h}_l),\ \mathcal{M}_{GT}\right),$$

where ξ' and ξ'' are two thresholds used to extract the masks from activation values $(\boldsymbol{\sigma}\cdot\boldsymbol{h}_l)$ and $(\boldsymbol{h}_l)$, respectively.[2]

Channel weights $\boldsymbol{\sigma}$ play a pivotal role in efficiently identifying over-activated areas associated with adversarial features, even in shallow layers. In fact, while the over-activation phenomenon may look straightforward to detect, our preliminary experiments revealed that simple operations directly applied to all channels, e.g., a channel-wise sum compared to a threshold, do not allow detecting the presence of adversarial objects within shallow layers.



(a) y-axis: mean activation; x-axis: channel-index

(b) Heatmaps with (right) and without (center) the channel-attention

Figure 5.17: (a) Mean channel-wise activation from the first spatial BiSeNet [2] layer during the inference of the attacked image; (b) representation of the heatmap w/ and w/o the attention mechanism.

Computing the Adversarial Trace

Following the above observations, we propose a practical usage and update of channel weights $\boldsymbol{\sigma}$, which are used to track an adversarial object over time. As anticipated in the paper introduction (see Figure 5.16), the proposed implementation is conceived to be complemented with a defense method capable of providing a starting mask $\mathcal{M}^0_{\boldsymbol{\delta}}$. When

[2]Note that the thresholds must be different because $\boldsymbol{\sigma}$ scales $\boldsymbol{h}_l$.

and how this starting mask needs to be computed will be discussed in Section 5.5.2, where the complete defense framework is presented.

The *adversarial trace* is defined as a sequence of weights that highlight the channels over-activated by adversarial attacks. Formally, given a layer L_l, the adversarial trace at time t, denoted by $\boldsymbol{\sigma}^t$, is a vector of C^l elements in $[0, 1]$ that enables the computation of an accurate heatmap $\mathcal{H}_l^t$ at time $t > 0$ as follows:

$$\mathcal{H}_l^t = \sum_{c=1}^{C^l} (\boldsymbol{\sigma}^t)^\tau \cdot \boldsymbol{h}_l^t, \tag{5.9}$$

where parameter τ is introduced to amplify the attention pattern within the heatmap. Figure 5.17(b) shows the benefits of using attention based on the adversarial trace. Once the heatmap is obtained, a threshold parameter ξ^t (defined below) can be used to devise the binary mask $\mathcal{M}_\delta^t$.

In this work, we proposed a per-frame update of the adversarial trace, so that the next element for time $t + 1$ can be computed as a function of the mask and activations computed at time t:

$$\boldsymbol{\sigma}^{t+1} = \mathcal{N}\left(\frac{\sum_{i,j=1}^{H,W} \left(\boldsymbol{h}_l^t \odot \bar{\mathcal{M}}_{\boldsymbol{\delta}}^t\right)_{c,i,j}}{|\bar{\mathcal{M}}_{\boldsymbol{\delta}}^t|} - \frac{\sum_{i,j=1}^{H,W} \left(\boldsymbol{h}_l^t \odot \mathcal{M}_{\boldsymbol{\delta}}^t\right)_{c,i,j}}{|\mathcal{M}_{\boldsymbol{\delta}}^t|} \right), \tag{5.10}$$

where $\mathcal{M}_{\boldsymbol{\delta}}^t$ is the predicted mask at time t, $\bar{\mathcal{M}}_{\boldsymbol{\delta}}^t$ denotes a complementary mask to address all other tensor values not interested by $\mathcal{M}_{\boldsymbol{\delta}}^t$, and $\mathcal{N}$ represents a normalization function that scales the values to the $[0, 1]$ range. In our experiments, we implemented $\mathcal{N}$ as a ReLU function followed by a channel-wise min-max normalization.

In particular, the first fractional term in Eq. (5.10) provides attention to over-activated patterns within the area of the adversarial object, while the second term provides negative attention to activations outside the same.

The effectiveness of using information obtained from the current frame to compute the next adversarial trace element $\boldsymbol{\sigma}^{t+1}$ was verified by means of experiments.

Summary of the approach. Figure 5.18 provides a schematic representation that illustrates the use and update of the adversarial trace for a frame at time t. Note that a noise filter (e.g., a Gaussian filter) can be introduced into the pipeline for computing the heatmap. As highlighted in [107], noise filters help mitigate the effects of small spurious activations.

Threshold definition. Differently from previous work, which adopted a static threshold computed offline on a calibration dataset, this work adopts an adaptive threshold that is dynamically computed frame by frame. This is necessary due to the attack-specific channel weighting of the attention mechanism, which makes not effective thresholds computed a priori. In ACAT, the threshold is updated at each frame as follows:

$$\xi^{t+1} = \max(\bar{\mathcal{H}}_l^t) + \psi(\bar{\mathcal{H}}_l^t). \tag{5.11}$$

In the above equation, $\bar{\mathcal{H}}_l^t = \mathcal{H}_l^t \odot D(\bar{\mathcal{M}}_{\boldsymbol{\delta}}^t)$, where $D(\cdot)$ is an operator that expands $\bar{\mathcal{M}}_{\boldsymbol{\delta}}^t$ by means of an unitary kernel convolution. This expansion is designed to account for *uncertainty* in the areas around the mask, coping with potential spurious activations close

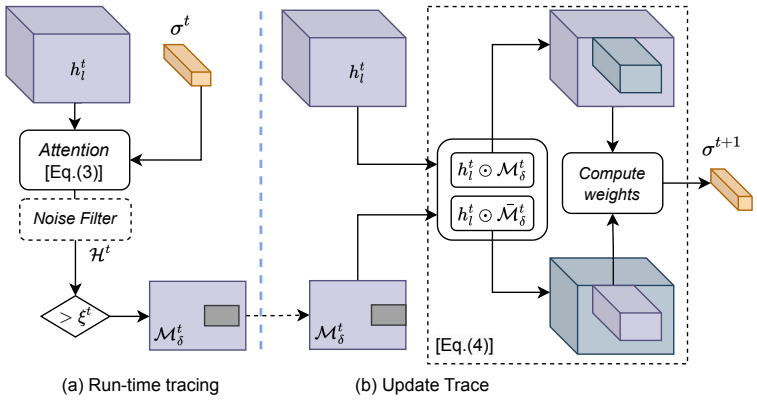


Figure 5.18: Illustration of the operations to implement adversarial attention mechanism performed at time t. The resulting output is the next element $\boldsymbol{\sigma}^{t+1}$ of the adversarial trace.

to its border. After applying an unitary convolution, non-integer values can be obtained: hence, the operator $D(\cdot)$ eventually binarizes all values using a threshold equal to 0.5.

To further reduce false positives, an extra safety margin $\psi(\bar{\mathcal{H}}_l^t)$ is included in Eq. (5.11). It is computed as the difference between the v-th percentile of the values in $\bar{\mathcal{H}}_l^t$ and the mean value of the same. In our experiments, we used $v = 70$, which proved to offer effective resilience to uncertainty.

5.5.2 ACAT Framework

This section shows how to integrate adversarial-channel attention within the continuous processing loop of vision applications. Algorithm 8 reports the pseudocode of the operations to be performed at each frame (retrieved with function capture_frame()). To improve readability, the discrete-time notation with the superscript t is omitted in the pseudocode, as all variables are updated to be used at the next cycle.

For each frame, it checks if the adversarial trace $\boldsymbol{\sigma}$ exists. If not, it means no adversarial attack was detected at the previous frame. In this case, a state-of-the-art attack detection method, e.g., [107], is executed (line 5) with a single inference pass. If the latter detects an attack, the algorithm initializes the adversarial trace $\boldsymbol{\sigma}$, computes the threshold ξ, and leverages the mask compute by the state-of-the-art method to defend from the attack (lines 8-10). The processing of the current frame can hence end.

Otherwise, when the adversarial trace $\boldsymbol{\sigma}$ is available from the previous frame, the algorithm leverages it to compute the defense mask following the results of Sec. 5.5.1 (lines 14-16). If the mask is meaningful (details provided next), it also computes the next adversarial trace and threshold (lines 21-22), still based on Sec. 5.5.1, and continues the inference process by applying the mask at the inner layer L_l to defend from the attack (line 23).

Reset criterion Knowing about the connection between the mask size and the induced adversarial effect by the masked attack [107], we use the number of pixels detected in the predicted complementary mask to decide whether to reset adversarial tracing or not. This

Algorithm 8 Adversarial-Channel Attention Tracing

1: $\boldsymbol{\sigma} \leftarrow$ None
2: **while** True **do**
3: $x \leftarrow$ capture_frame()
4: **if** $\boldsymbol{\sigma}$ is None **then**
5: $(y, \mathcal{M}_\delta, \boldsymbol{h}_l) \leftarrow$ inference_with_SoA_method(x, f)
6: **if** $\mathcal{M}_\delta$ is not None **then**
7: *#Attack notified*
8: $\boldsymbol{\sigma} \leftarrow$ ACAT_update$(\boldsymbol{h}_l, \mathcal{M}_\delta)$ *#Eq. (5.10)*
9: $\xi \leftarrow$ compute_threshold$(\boldsymbol{h}_l, \mathcal{M}_\delta)$ *#Eq. (5.11)*
10: $y = f(x \odot \mathcal{M}_\delta)$ *# Inference with masked input*
11: **end if**
12: **Continue** *#Wait for next frame*
13: **end if**
14: $\boldsymbol{h}_l = f^{[0 \rightarrow l]}(x)$
15: $\mathcal{H} =$ noise_filter $\left(\sum_{c=1}^{C^l} (\boldsymbol{\sigma})^\tau \cdot \boldsymbol{h}_l\right)$ *#Eq. (5.9)*
16: $\mathcal{M}_\delta \leftarrow \Lambda^\xi(\mathcal{H})$ #Apply threshold to get mask
17: **if** $|\bar{\mathcal{M}}_\delta| < \lambda_\mathcal{M}$ **then**
18: $\boldsymbol{\sigma} \leftarrow$ None *#Stop adv. tracing*
19: $y = f^{[l \rightarrow L]}(\boldsymbol{h}_l)$
20: **else**
21: $\boldsymbol{\sigma} \leftarrow$ ACAT_update$(\boldsymbol{h}_l, \mathcal{M}_\delta)$ *#Eq. (5.10)*
22: $\xi \leftarrow$ compute_threshold$(\boldsymbol{h}_l, \mathcal{M}_\delta)$ *#Eq. (5.11)*
23: $y = f^{[l \rightarrow L]}(\boldsymbol{h}_l \odot \mathcal{M}_\delta)$ *#Inference with masked layer*
24: **end if**
25: **end while**

could mean that the adversarial object is either too small or far away from the camera. Specifically, we disable adversarial tracing when the computed mask has less than $\lambda_\mathcal{M}$ pixels (line 17), where the latter is a configurable parameter.

Timing performance State-of-the-art approaches require *two* inference passes to defend from adversarial attack while, as it can be noted from Algorithm 8, once an attack has been detected at a certain frame, ACAT allows defending from the same with just *one* inference pass (completed in two stages at lines 14 and 23, respectively) for the next frame. This holds until tracing is active, i.e., the reset criterion is not reached. Once a new attack will be detected the same will hold for the next frames, and so on and so forth. Overall, ACAT allows significantly improving the timing performance of the defense mechanism (quantitative results provided in the next section) by halving inference times in general, except for the very first frame in which the attack manifests.

5.5.3 Experiments

The experimental evaluation is focused on semantic segmentation models designed for autonomous driving, which have recently garnered attention due to the need to address real-world adversarial attacks in outdoor scenarios [24, 90]. Please note however that defense mechanisms based on over-activation also work for different computer vision tasks, where the connection between over-activation and adversarial effect persists [24, 116].

In the following, we first provide details on the experimental settings. Then, we present and discuss different tests and ablation studies conducted to validate the design and ben-

efits of the proposed defense algorithm. All the experiments were implemented using PyTorch [131] on a machine with 8xNVIDIA-A100 GPUs.

Experimental settings

Complete multi-frame benchmarks to evaluate the effectiveness of defense methods against real-world adversarial attacks are not available from previous work.

For this reason, we addressed two evaluation approaches: **(i)** attack scenarios generated with the CARLA simulator [74], used to test the attention mechanism of ACAT only, and **(ii)** digitally attacked video generated with Cityscapes [156], which instead allow testing the whole ACAT framework.

Attacks in CARLA-simulated scenarios With the intent of facing with realistic settings, we utilized the Carla-Gear framework, which offers 9 photo-realistic scenarios (50 test images each) collected in areas of Carla-town 10 [74], integrating adversarial billboards specifically designed for each model in use. Please note that the framework only provides random viewpoints of the area next to the adversarial billboards, which are not sequential videos. For this reason, this setting allows evaluating the benefits of adopting adversarial-channel attention only, i.e., improving the capabilities of state-of-the-art defense mechanisms when used on a single frame, while not enabling meaningful tests to evaluate ACAT as a whole.

Digitally attacked video datasets To address the lack of a dedicated video dataset featuring attacked driving scenes, we generated custom videos that include digital adversarial attacks. Three extended sequences from Cityscapes [156] videos[3] were utilized with images sized at 2048x1024 pixels. Within each video, a dynamic adversarial patch was digitally introduced in the frames, which, at every frame, changes its position and scaling factor, following a sinusoidal trend. The patch position and scale were computed as follows:

$$\begin{bmatrix} x_{pos} \\ y_{pos} \\ s \end{bmatrix} = \begin{bmatrix} c_x + A_x \sin(\alpha_x \cdot k + \omega_x) \\ c_y + A_y \sin(\alpha_y \cdot k + \omega_y) \\ 1 + A_s \sin(\alpha_s \cdot k + \omega_s) \end{bmatrix}, \tag{5.12}$$

where t is the frame index, x_{pos} and y_{pos} are coordinates of the position of the patch, c_x, c_y are the center coordinates of the frame, and s is the scaling factor of the patch. In our experiments we set $(A_x, A_y, A_s, \alpha_x, \alpha_y, \alpha_s) = (500, 300, 0.3,\ 0.05, 0.05, 0.05)$. The ω values represent a phase used to randomize tests among different initial positions. With these settings, the patch can partially go beyond the image boundaries while holding a size that is sufficient for producing an adversarial effect [24, 181]. The α values provide a smooth trend of the patch among subsequent frames.

The attack mechanism used to generate the patch was the Over-Activation-aware Expectation Over Transformation (EOT) optimizatin (presented in 5.6), where the parameter $\beta \in [0, 1]$ is used to regulate the over-activation level of the patch within the internal layers while reducing the adversarial effect (the lower the β the lower the over-activation, and so the adversarial effect). This approach is particularly useful for evaluating the robustness

[3] https://www.cityscapes-dataset.com, *leftImg8bit_demoVideo.zip*

of our approach when the attacker tries to limit over-activation to mount attacks that are difficult to detect.

Network models and defense methods Following related work on semantic segmentation [165], we considered real-time high-performance DNN models: DDRNet-Slim23 version [3] and BiSeNetX39 [2]. We use the pre-trained versions available from [165].

We compared our method ACAT with two lightweight single-frame approaches designed to mask real-world attacks. The first is LGS [103], which applies gradient-based filtering of the image to mask adversarial pixels. The second is ZMask [107] (presetend in Section 5.3)

For ACAT, we set $\tau = 2$, and the kernel size to $5, 3$ and $31, 11$ for the Gaussian filter and dilatation operators for Bisenet and DDRNet, respectively. The different sizes are due to the different spatial dimensions of the features. The layers analyzed by ACAT are in the shallower blocks of the considered model, specifically the output of the second block of DDRNet and the output of the first context layer of BiSeNet. Ablation studies were also performed to understand the selection process of these layers (see Sec. 5.5.3).

Metrics Different metrics were used to assess the performance of the addressed mechanisms. Given the unavailability of annotations for the Cityscapes videos, we use the binary Intersection-over-Union (IoU), referred to as Mask-IoU, to measure the overlap between the predicted complementary mask $\bar{\mathcal{M}}^t_{\boldsymbol{\delta}}$ (whose values equal to 1 denote the predicted adversarial region) and the corresponding ground-truth mask $\bar{\mathcal{M}}^t_{GT}$. Intuitively, Mask-IoU quantifies the quality of the predicted defense mask: the higher the better.

For the tests conducted on the Carla-Gear dataset, as indicated in the benchmark, we measured the effectiveness of adversarial attacks by addressing the original multi-class MIoU [165, 156] of the task, since annotations are available.

Performance Evaluation on Carla

Table 5.6 highlights the advantages of our approach across nine scenarios of the Carla-Gear dataset on BiSeNet (top part) and DDRnet (bottom). Regarding ACAT, which is designed to integrate with state-of-the-art defenses, we conducted analyses under two settings: $ACAT_{ZM}$ and $ACAT_{GT}$. The former utilizes ZMask [107], reflecting a realistic scenario built upon an already available approach. In the second setting, $ACAT_{GT}$ assumes the knowledge of an ideal, ground-truth mask at first frame in which the attack is detected. While this setting depicts a less realistic scenario, it serves to highlight the intrinsic performance of ACAT, independently from the defense method with which it is integrated.

In the table, the first line for each scenario depicts the task MIoU without an adversarial billboard, while subsequent lines show the drop in MIoU with the adversarial billboard and/or without the related defenses. The value between the brackets for the ACAT results depicts the number of times that the reset criterion takes effect, necessitating the extraction of a new starting mask. As also mentioned in [165], there are instances where certain attacks can be particularly challenging for a specific model and scenario, leading to a poor reduction in the MIoU. To assist the reader, in Table 5.6 we highlighted in gray the scenarios that have resulted in a more pronounced adversarial effect.

	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Scene 9
No Attack	28.35	29.66	28.88	24.71	25.10	26.51	20.56	24.46	23.68
No Def	-10.2	-4.32	-2.81	-6.4	-5.06	-9.5	-7.00	-2.93	-6.75
$ACAT_{GT}$	**-2.9** [1]	**+0.59** [1]	**-2.67** [1]	**-2.11** [2]	**-0.9** [1]	-5.4 [1]	**-2.87** [1]	**-0.63** [1]	**-0.78** [1]
$ACAT_{ZM}$	**-2.9** [2]	-3.4 [1]	-3.15 [12]	-2.82 [2]	-1.1 [7]	**-5.33** [1]	-2.89 [21]	-5.92[1]	-5.12[1]
ZMask	-5.32	-2.35	-2.81	-3.01	-1.7	-5.87	-4.84	-3.0	-5.89
LGS	-8.63	-5.09	-4.09	-6.43	-4.04	-6.65	-6.01	-1.60	-5.83
No Attack	33.04	34.11	34.39	34.22	31.01	33.26	30.73	31.86	37.41
No Def	+0.65	-1.74	-0.6	-6.53	-1.01	-2.31	-12.69	-3.03	-1.11
$ACAT_{GT}$	**+1.59** [1]	**-1.58** [1]	**+0.66** [3]	**-3.29** [1]	**+0.4** [1]	**-0.83** [1]	**-1.52** [1]	**-1.11** [1]	**-0.86** [1]
$ACAT_{ZM}$	+0.59 [2]	-1.73 [3]	-0.67 [15]	-3.52 [6]	+0.35 [5]	-2.85 [1]	-4.25 [33]	-2.22 [16]	-1.01 [5]
ZMask	-2.27	-3.2	-1.32	-5.2	-0.97	-5.55	-3.58	-0.98	-1.01
LGS	+0.45	-3.51	+1.01	-8.30	-0.87	-4.06	-11.97	-1.22	-1.25

Table 5.6: Variation of the multi-class mIoU w/ and w/o defense mechanisms across the 9 driving scenarios of CarlaGear [165]. Results for both BiseNet [2] (top) and DDRNet [3] (bottom) are reported. The values inside square brackets denote the number of times ACAT required to be re-initialized (the reset criterion in line 17 occurs). The results of ACAT are averaged across 5 random shuffling of each scene dataset.

As it can be noted from the table, ACAT consistently outperforms the other methods, significantly reducing the number of extra inference passes, reaching the reset conditions only a few times. Note also that $ACAT_{ZM}$ generally improves the performance of ZMask. However, when ZMask fails to return an accurate mask, it may jeopardize the initialization of ACAT, resulting in lower performance (e.g., scene 8 - BiSeNet and scene 7 on DDRNet). This is not the case for $ACAT_{GT}$, confirming that the lower performance is not due to ACAT. Concerning LGS, as known from previous work, it loses accuracy in real-world scenarios [107, 106].

Please note that, in these tests only, we did not update the trace and threshold of ACAT (lines 21-22 in Algorithm 8). As anticipated above, this is because the tested images do not pertain to sequential video. The whole ACAT framework is instead addressed by the following experiments.

It is however interesting to also observe the number of times ACAT required a re-initialization (reset criterion) during these tests, even if updates are disabled. As one may expect, we found scenarios in which the mask provided by ZMask was frequently required (e.g., note the numbers between square brackets in Table 5.6 for Scenes 3 and 7), while surprisingly, in other cases, it was not at all. This means that the attention mechanism offered by ACAT is sometimes effective even with sporadic updates (see also the other experiments below). Conversely, in the former case, we found that the reset criterion was prominently triggered because the mask provided by ZMask was not particularly accurate, as $ACAT_{GT}$ almost never requires to be re-initialized.

Performance Evaluation on Digital Attacks

In Figure 5.19, we studied the Mask-IoU for the digital attacked video. To show that ACAT provides high robustness even when the adversarial trace and thresholds are not updated at every frame as mandated by Alg. 8, we measured the average Mask-IoU under $ACAT_{ZM}$ on attacked video streams from Cityscapes, varying the period with which the trace and thresholds are updated. The period is expressed in number of frames and is reported on the x-axis of the figure (e.g., value 1 on the x-axis means that the update

occurs at each frame). In the analysis, we tested two digital adversarial patches, with $\beta = 0.6$ and $\beta = 0.8$, to better investigate on the robustness of ACAT. We also evaluated ZMask, which achieves $(0.66, 0.75)$ and $(0.70, 0.72)$ of Mask-IoU with ($\beta = 0.6$, $\beta = 0.8$) for Bisenet and DDRNet, respectively. The results for LGS are not reported since it does not provide a binary defense mask, but rather a soft filtering of the input image, for which it is not possible to compute the Mask-IoU.

The figure shows that ACAT surprisingly works well even with sporadic updates of the adversarial trace and thresholds. This was also due to the fact that the Cityscapes videos are related to rather static scenarios. In fact, despite some changes in the appearance of adversarial patches and their background, the over-activated pattern of the patch in these cases continuously insist on a similar set of channels to induce the adversarial effect. An update of the parameters is anyway required in more dynamic scenarios with more frequent changes of the background and appearance of the adversarial object. The figure also shows that sporadic updates always provide better performance than ZMask.

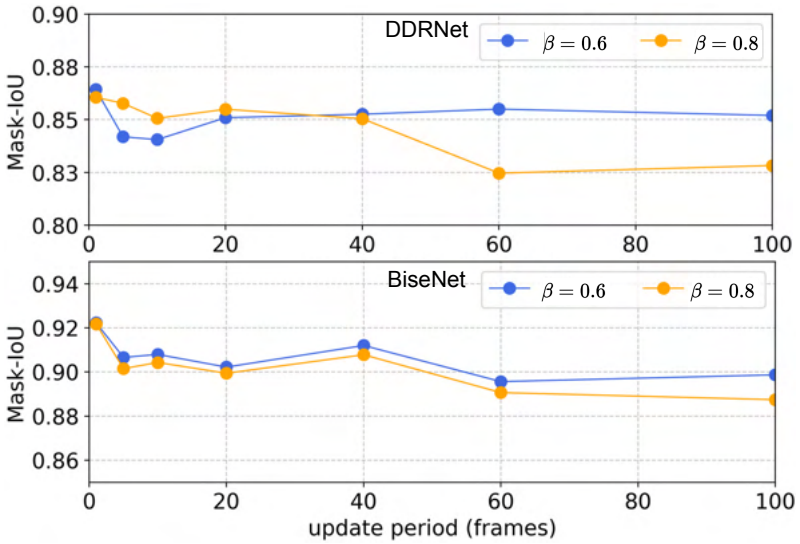


Figure 5.19: Mask-IoU performance by varying the update period (in frames, x-axis of the figure) of the adversarial trace and thresholds. The upper plot refers to the DDRNet architecture, while the second one pertains to BiseNet. The tests evaluate the performance of $ACAT_{ZM}$ for two distinct digital patches ($\beta = 0.6$ and $\beta = 0.8$). The results are the average of five different initializations of the ω parameters in Eq. (5.12).

Ablation Studies

To better understand the contribution of each operation performed by ACAT to its overall performance, Table 5.7 reports the Mask-IoU of $ACAT_{GT}$ on the attacked videos under different settings. With the aim of acquiring a deeper understanding about the attention mechanism of ACAT, we independently examined the two fractional terms defined in Equation (5.10) to update the adversarial trace. The first term provides positive attention within the attacked area, which is the most important part of the attention mechanism. We hence introduce a flag Att^{+} to indicate a setting of ACAT that uses this term. The second

term refines the previous operation by introducing negative attention to the elements outside the attacked area. Another flag Att^- is also introduced to denote if this second term is used by ACAT.

As shown in the table, it is clear that using both Att^+ and Att^- leads to better results, hence motivating the construction of Equation (5.10). In general, it is evident that the use of the attention mechanism significantly improves the Mask-IoU when compared to not using attention (both Att^+ and Att^- disabled, first rows of the table). Its benefits are especially notable in the results obtained with DDRNet, where adversarial over-activations in the shallow layers proved to be very difficult to detect without attention. These observations are also illustrated with an example frame in Figure 5.20.

The ablation studies also tested ACAT with and without the update of the adversarial trace and the threshold of Eq. (5.11) (flag *Upd* in Table 5.7), and with and without the noise filter (flag NF).

				Bisenet		DDRNet	
Att^+	Att^-	*Upd*	NF	$\delta_{0.6}$	$\delta_{0.8}$	$\delta_{0.6}$	$\delta_{0.8}$
				11.9	16.2	0.00	0.01
			✓	89.0	88.7	7.2	0.6
✓			✓	90.7	90.2	76.91	84.90
✓		✓	✓	91.3	90.8	72.05	83.05
✓	✓		✓	**92.24**	91.9	85.56	84.22
✓	✓	✓	✓	92.23	**92.18**	**86.12**	**86.42**

Table 5.7: Experimental results of ablation studies with respect to the different components used to update the adversarial trace. The results are in terms of Mask-IoU and related to the digitally-attacked Cityscape videos using $ACAT_{\text{GT}}$ as a defense mechanism. Two model-specific patches were utilized, one with $\beta = 0.6$ and another with $\beta = 0.8$.. In the table, Att^+ and Att^- denote two flags to enable the two attention terms of Equation 5.10, respectively, while *Upd* and the NF are other two flags to enable the update of the trace and threshold, and the usage of the noise filter, respectively.

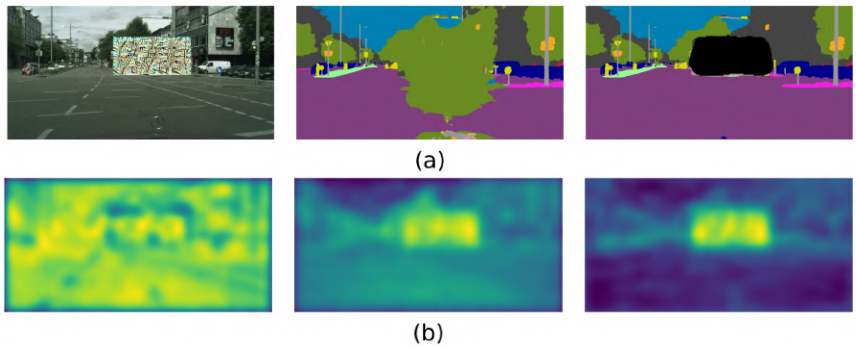


Figure 5.20: (a) Comparison of the adversarial effect of a patch with $\beta = 0.6$ (left), with ACAT_{GT} mechanism (right), and without the ACAT_{GT} mechanism (middle). (b) Illustration of the heatmap among different settings, from left to right: **(i)** only NF enabled, **(ii)** only Att^+ and NF enabled, **(iii)** Att^+, Att^-, NF, and *Upd* enabled.

Layer-wise Ablation

Figure 5.21 reports the Mask-IoU by varying the layer of the DDRNet model with which ACAT operates (parameter l in Alg. 8). As observed, the more shallow the layer the better the performance. In fact, if addressing deeper layers, the mask based on the over-activation extends beyond the ground-truth position (in the figure, only the yellow parts denote a complete overlap of the ground-truth and the predicted mask). This is attributed to the fact that the features of shallow layers are less spatially compressed (i.e., they have a higher spatial size) than those in deeper layers.

Note that, for fair comparisons, in layer $l = 3$, we used the same kernel size as layer $l = 2$ (i.e., 3), which provided better performance than kernel size 1 (i.e., no Gaussian filter). While, for layer $l = 5$, we did not use the Gaussian filter due to the high compression of the spatial dimension. These results highlight how ACAT allows focusing on shallow layers so that attacks can be masked within a single inference pass, as opposed to previous work that analyzes deep layers and hence requires another inference pass to mask attacks.

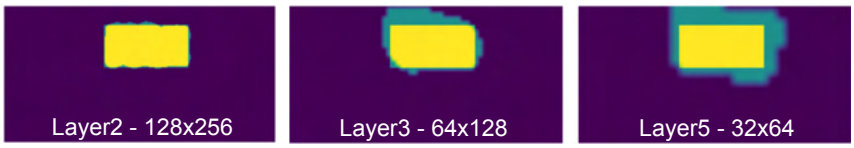


Figure 5.21: Mask-IoU (in black) for the digital adversarial patch with $\beta = 0.6$ and 0.8 on attacked cityscapes video using the $ACAT_{GT}$ on different layers of DDRNet. The figures show the overlapping between the predicted mask and the ground truth for $\beta = 0.6$, with the highest color indicating the degree of overlap. The depth of the DDRNet layer and the spatial dimension are denoted in white.

Timing Evaluation

To demonstrate the improvements provided by ACAT in terms of running times, we measured the inference times when testing the attacked Cityscapes videos. Figure 5.22 reports the overall inference time required on average to process a frame by the tested defense mechanisms, with the baseline labeled by *No_Def*, denoting the original model without defenses. Two inference times are reported for ZMask: when an attack is not detected and when an attack is detected, which are separated by a slash in the figure. As expected, when ZMask detects an attack, its inference time is approximately twice the one of the baseline model. Conversely, when no adversarial attacks are detected, ZMask is particularly efficient and hence represents an excellent choice to work in conjunction with ACAT, which activates only when an attack is first detected (see Alg. 8).

The figure also reports the results for another state-of-the-art defense mechanism, named MaskNet [106], which incorporates a secondary model. It is relatively more expensive due to the necessity of always running an encoder-decoder model in tandem with the original model.

Note that LGS exhibits comparable timing performance with respect to ACAT, since it focuses on specific filters that are directly applied to the input image. However, as shown

by the results in Table 5.6 and other studies in previous work [106, 107], LGS tends not to perform well in detecting adversarial attacks that can be carried out in real world, i.e., by means of physical adversarial objects.

In summary, these results remark on how ACAT provides a well-balanced trade-off between defense performance and overall inference time.

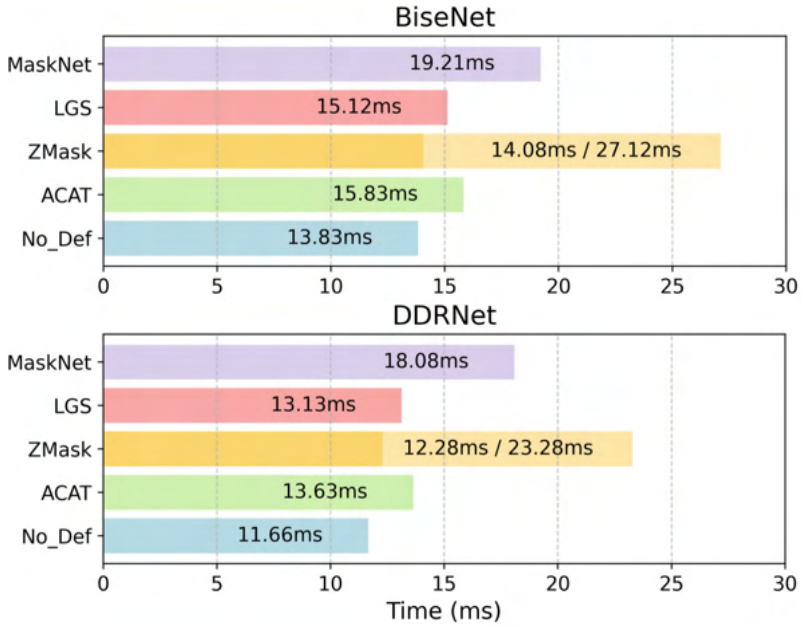


Figure 5.22: Overall inference time with and without defense mechanisms for DDRNet and BiseNet.

5.6 Summing up Defense Methods and Future Steps

In the previous sections, we addressed three defense strategies aimed at: (i) enhancing the performance of adversarial detection and (ii) adversarial masking, while (iii) reducing the computational cost of adversarial masking strategies in multi-frame applications.

Beyond the performance evaluation in both detection and masking, the presented approaches provided a sense of explainable robustness, taking advantage of the insights on the over-activation presented in Section 5.1. This includes high robustness against defense-aware attacks and an understanding of when the defense mechanism might fail, i.e., where the attacker is constrained to craft patches with low over-activation, thus reducing their effectiveness according to the insights discussed above.

The addressed studies and achieved results open up several future research directions:

- *Investigating over-activation in recent transformer models.* While the research in this thesis primarily focuses on convolutional neural networks, it is important to note that recent studies have also explored the generation of adversarial patches specifically for transformer neural networks [182]. As original vision transformers do not leverage convolutional blocks, which is important assumption behind the over-activation analysis presented, this suggests interesting investigation of over-activations in this new domain.

- *Deepening the understanding of universal adversarial perturbations.* Another intriguing area of study involves universal adversarial perturbations. Further research, including over-activation studies, could be valuable in this domain to assess the over-activation phenomenon and its implications (as preliminary discussed in Section 5.1).

- *Integrating over-activation awareness in model architecture or training.* Knowing the relationship between over-activation and induced adversrial effects, it is worth addressing the integration of training-aware approaches and specific blocks that could mitigate adversarial effects during the design phase.

Chapter 6

Conclusions

To summarize, this thesis provides comprehensive literature reviews, introduces novel works, and outlines future research directions in multiple areas related to trustworthy AI.

Coverage Testing. Section 2.2 reviews the literature concerning structural coverage criteria and more recent techniques. Then, in Section 3.1, we introduced a new approach aimed at applying coverage criteria in building a real-time monitoring tool. This tool uses coverage metrics to determine if a new input aligns with safe patterns computed offline. Finally, in Section 3.2, building on the extensive literature reviewed, we highlighted future research directions in coverage testing, emphasizing the need to deepen understanding of explainable methods, statistical analysis, and scalability for larger and more complex models.

Real-World Attacks. The thesis reviews the literature on real-world attacks in Section 2.3.2, inspiring the research on spatial robustness concepts for dense prediction tasks. In Chapter 4, we presented, to the best of our knowledge, the first real-world adversarial attack for semantic segmentation models in driving scenarios, proposing a new optimization method designed to attack the spatial robustness of these models. We also addressed the use of different transformations in the adversarial optimization process, leveraging virtual scenarios with the Carla simulator. This opens up the development of a schematic tool for evaluating robustness with customizable parameters. Lastly, Section 4.4 outlines future research investigations in real-world attacks, addressing new attack scenarios and improvements needed for the effective use of simulators in driving scenarios.

Defense Mechanisms Against Real-World Attacks. We faced the problem of designigin defense strategies to mitigate the effects of real-world attacks on different vision tasks. We first reviewed the literature concerning real-world defense strategies in Section 2.4.1, offering new perspectives to distinguish current works and highlighting crucial requirements for practical and feasible defenses in safety-critical systems. Then, in Chapter 5, we delved into an exploration of various works conducted to mitigate the effects of adversarial patches through an over-activation analysis, which has been demonstrated and discussed in Section 5.1. We also proposed future directions to further explore over-activation analysis and enhanced the applicability of defense mechanisms in cyber-physical scenarios.

Concluding Remarks

As stated in the introduction, Section 1.1, this thesis presents contributions and findings from publications conducted during the Ph.D. studies. The work focuses on expanding and disucssing the literature, encouraging a continuous dialogue on these subjects and setting the stage for future endeavors.

Bibliography

[1] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[2] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision*. Springer, 2018.

[3] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv:2101.06085*, 2021.

[4] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[5] HuggingFace. open llm leaderboard, 2023.

[6] Fahmida Chowdhury, Puneet Wahi, Ramesh Raina, and Suneetha Kaminedi. A survey of neural networks applications in automatic control. pages 349 – 353, 04 2001.

[7] Kai Hu, Xu Chen, Liguo Weng, Lang Tian, and Yongzan Hu. A survey of deep neural network sliding mode control in robot application. In *2020 Chinese Automation Congress (CAC)*, pages 6659–6662, 2020.

[8] Roberto Gozalo-Brizuela and Eduardo C Garrido-Merchán. A survey of generative ai applications. *arXiv preprint arXiv:2306.02781*, 2023.

[9] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[10] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7374–7383, 2019.

[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[13] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *in 35th International Conference on Machine Learning*, 2018.

[14] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[15] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[16] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[17] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 2154–2156, New York, NY, USA, 2018. Association for Computing Machinery.

[18] Nick Frosst Toronto Machine Learning Summit. Certifiable robustness to adversarial attacks; what is the point?, 2020.

[19] Donghua Wang, Wen Yao, Tingsong Jiang, Guijian Tang, and Xiaoqian Chen. A survey on physical adversarial attack in computer vision. *arXiv preprint arXiv:2209.14262*, 2022.

[20] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys*, 55(13s):1–39, 2023.

[21] Vanlalruata Hnamte and Jamal Hussain. An extensive survey on intrusion detection systems: Datasets and challenges for modern scenario. In *2021 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, pages 1–10, 2021.

[22] Minh Pham and Kaiqi Xiong. A survey on security attacks and defense techniques for connected and autonomous vehicles. *Computers & Security*, 109:102269, 2021.

[23] Mahdi Dibaei, Xi Zheng, Kun Jiang, Robert Abbas, Shigang Liu, Yuexin Zhang, Yang Xiang, and Shui Yu. Attacks and defences on intelligent connected vehicles: a survey. *Digital Communications and Networks*, 6(4):399–421, 2020.

[24] Giulio Rossolini, Federico Nesti, Gianluca D'Amico, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. On the real-world adversarial robustness of real-time semantic segmentation models for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023.

[25] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[26] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial Patch. *arXiv:1712.09665 [cs]*, May 2018.

[27] Qian Yang, J. Jenny Li, and David M. Weiss. A survey of coverage-based testing tools. *The Computer Journal*, 52(5):589–597, 2009.

[28] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. *ACM Trans. Embed. Comput. Syst.*, 18(5s), oct 2019.

[29] Zhiyang Zhou, Wensheng Dou, Jie Liu, Chenxin Zhang, Jun Wei, and Dan Ye. Deepcon: Contribution coverage testing for deep learning systems. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 189–200, 2021.

[30] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. Deephunter: A coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2019, page 146–157, New York, NY, USA, 2019. Association for Computing Machinery.

[31] Siqi Li, Xiaofei Xie, Yun Lin, Yuekang Li, Ruitao Feng, Xiaohong Li, Weimin Ge, and Jin Song Dong. Deep learning for coverage-guided fuzzing: How far are we? *IEEE Transactions on Dependable and Secure Computing*, pages 1–13, 2022.

[32] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18, page 120–131, New York, NY, USA, 2018. Association for Computing Machinery.

[33] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 1–18, New York, NY, USA, 2017. Association for Computing Machinery.

[34] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

[35] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. ICSE '19, page 1039–1049. IEEE Press, 2019.

[36] Pengcheng Zhang, Bin Ren, Hai Dong, and Qiyin Dai. Cagfuzz: Coverage-guided adversarial generative fuzzing testing for image-based deep learning systems. *IEEE Transactions on Software Engineering*, 48(11):4630–4646, 2022.

[37] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*, pages 4901–4911. PMLR, 2019.

[38] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 109–119, 2018.

[39] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Deepconcolic: Testing and debugging deep neural networks. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 111–114, 2019.

[40] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142, 2018.

[41] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111, 2018.

[42] Qiang Hu, Lei Ma, Xiaofei Xie, Bing Yu, Yang Liu, and Jianjun Zhao. Deepmutation++: A mutation testing framework for deep learning systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1158–1161, 2019.

[43] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[44] Yuanyuan Yuan, Qi Pang, and Shuai Wang. Revisiting neuron coverage for dnn testing: A layer-wise and distribution-aware criterion. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1200–1212. IEEE, 2023.

[45] Zhenlan Ji, Pingchuan Ma, Yuanyuan Yuan, and Shuai Wang. Cc: Causality-aware coverage criterion for deep neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1788–1800, 2023.

[46] Marco Di Natale, Kelly J. Hayhurst, Dan S. Veerhusen, John Chilenski, and L.K. Rierson. A practical tutorial on modified condition/decision coverage. 2001.

[47] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. Npc: Neuron path coverage via characterizing decision logic of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(3):1–27, 2022.

[48] Moritz Böhle, Fabian Eitel, Martin Weygandt, and Kerstin Ritter. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. *Frontiers in aging neuroscience*, 11:194, 2019.

[49] Muhammad Usman, Youcheng Sun, Divya Gopinath, Rishi Dange, Luca Manolache, and Corina S Păsăreanu. An overview of structural coverage metrics for testing neural networks. *International Journal on Software Tools for Technology Transfer*, 25(3):393–405, 2023.

[50] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. Is neuron coverage a meaningful measure for testing deep neural networks? In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 851–862. ACM, 2020.

[51] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[52] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[53] Ashutosh Chaubey, Nikhil Agrawal, Kavya Barnwal, Keerat K Guliani, and Pramod Mehta. Universal adversarial perturbations: A survey. *arXiv preprint arXiv:2005.08087*, 2020.

[54] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[55] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[56] Krishna Kanth Nakka and Mathieu Salzmann. Indirect local attacks for context-aware semantic segmentation networks. In *16th European Conference Computer Vision ECCV*, volume 12350. Springer, 2020.

[57] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. {PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier. In *31st USENIX Security Symposium*, 2022.

[58] A Braunegg, Amartya Chakraborty, Michael Krumdick, Nicole Lape, Sara Leary, Keith Manville, Elizabeth Merkhofer, Laura Strickhart, and Matthew Walmer. Apricot: A dataset of physical adversarial attacks on object detection. In *European Conference on Computer Vision*, 2020.

[59] Ke Xu, Yao Xiao, Zhaoheng Zheng, Kaijie Cai, and Ram Nevatia. Patchzero: Defending against adversarial patch attacks by detecting and zeroing the patch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4632–4641, 2023.

[60] Jiang Liu, Alexander Levine, Chun Pong Lau, Rama Chellappa, and Soheil Feizi. Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[61] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA*, pages 14242–14251. IEEE, 2020.

[62] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1028–1035, Jul. 2019.

[63] Yang Zhang, Hassan Foroosh, Philip David, and Boqing Gong. CAMOU: Learning physical vehicle camouflages to adversarially attack detectors in the wild. In *International Conference on Learning Representations*, 2019.

[64] Donghua Wang, Tingsong Jiang, Jialiang Sun, Weien Zhou, Xiaoya Zhang, Zhiqiang Gong, Wen Yao, and Xiaoqian Chen. Fca: Learning a 3d full-coverage vehicle camouflage for multi-view physical adversarial attack. *ArXiv*, abs/2109.07193, 2021.

[65] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *16th European Conference*, volume 12350. Springer, 2020.

[66] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6898–6907, 2019.

[67] Bingyao Huang and Haibin Ling. Spaa: Stealthy projector-based adversarial attacks on deep image classifiers. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 534–542. IEEE, 2022.

[68] Abhiram Gnanasambandam, Alex M Sherman, and Stanley H Chan. Optical adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 92–101, 2021.

[69] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. On success and simplicity: A second look at transferable targeted attacks. *Advances in Neural Information Processing Systems*, 34:6115–6128, 2021.

[70] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[71] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan Loddon Yuille. Adversarial attacks beyond the image space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4297–4306, 2017.

[72] David A. Forsyth and Jean Ponce. *Computer Vision - A Modern Approach, Second Edition.* Pitman, 2012.

[73] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 665–681. Springer, 2020.

[74] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: an open urban driving simulator. In *1st Annual Conference on Robot Learning*, volume 78. PMLR, 2017.

[75] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540, Vienna Austria, October 2016. ACM.

[76] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18)*, 2018.

[77] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. IEEE Computer Society, 2018.

[78] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 1–17. Springer, 2020.

[79] Tong Wu, Xuefei Ning, Wenshuo Li, Ranran Huang, Huazhong Yang, and Yu Wang. Physical adversarial attack on vehicle detector in the carla simulator. *ArXiv*, abs/2007.16118, 2020.

[80] Mark Lee and Zico Kolter. On Physical Adversarial Patches for Object Detection. *arXiv:1906.11897*, June 2019.

[81] Wenjun Zhu, Xiaoyu Ji, Yushi Cheng, Shibo Zhang, and Wenyuan Xu. TPatch: A triggered physical adversarial patch. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 661–678, Anaheim, CA, August 2023. USENIX Association.

[82] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1989–2004, 2019.

[83] Amira Guesmi, Ioan Marius Bilasco, Muhammad Shafique, and Ihsen Alouani. Advart: Adversarial art for camouflaged object detection attacks. *arXiv preprint arXiv:2303.01734*, 2023.

[84] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J Black. Attacking optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2404–2413, 2019.

[85] J. Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Binh Yang, Richard Du, Frank Cheng, and R. Urtasun. Physically realizable adversarial examples for lidar object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13713–13722, 2020.

[86] Koichiro Yamanaka, Ryutaroh Matsumoto, Keita Takahashi, and Toshiaki Fujii. Adversarial Patch Attacks on Monocular Depth Estimation Networks. *arXiv e-prints*, page arXiv:2010.03072, October 2020.

[87] Sangjun Kim, Kyung-Joon Park, and Chenyang Lu. A survey on network security for cyber–physical systems: From threats to resilient design. *IEEE Communications Surveys & Tutorials*, 24(3):1534–1573, 2022.

[88] Abhijith Sharma, Yijun Bian, Phil Munz, and Apurva Narayan. Adversarial patch attacks and defences in vision-based tasks: A survey. *arXiv preprint arXiv:2206.08304*, 2022.

[89] Jung Im Choi and Qing Tian. Adversarial attack and defense of YOLO detectors in autonomous driving scenarios. In *2022 IEEE Intelligent Vehicles Symposium, IV 2022, Aachen, Germany, June 4-9, 2022*, pages 1011–1017. IEEE, 2022.

[90] Jindi Zhang, Yang Lou, Jianping Wang, Kui Wu, Kejie Lu, and Xiaohua Jia. Evaluating adversarial attacks on driving safety in vision-based autonomous vehicles. *IEEE Internet Things J.*, 9(5):3443–3456, 2022.

[91] Mohammad R. Alam and Chris M. Ward. Adversarial examples in self-driving: A review of available datasets and attacks. In *51st Applied Imagery Pattern Recognition Workshop, AIPR 2022, Washington, DC, USA, October 11-13, 2022*, pages 1–6. IEEE, 2022.

[92] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8561–8570, 2021.

[93] Hiroharu Kato, Y. Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2017.

[94] L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu. Universal physical camouflage attacks on object detectors. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 717–726, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[95] Aniruddha Saha, Akshayvarun Subramanya, Koninika Patil, and Hamed Pirsiavash. Role of spatial context in adversarial robustness for object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3403–3412. IEEE, 2020.

[96] Samer Y. Khamaiseh, Derek Bagagem, Abdullah Al-Alaj, Mathew Mancino, and Hakam W. Alomari. Adversarial deep learning: A survey on adversarial attacks and defense mechanisms on image classification. *IEEE Access*, 10:102266–102291, 2022.

[97] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[98] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54. IEEE, 2020.

[99] Zitao Chen, Pritam Dash, and Karthik Pattabiraman. Jujutsu: A two-stage defense against adversarial patch attacks on deep neural networks. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 689–703, 2023.

[100] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. {PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2237–2254, 2021.

[101] Chong Xiang and Prateek Mittal. Detectorguard: Provably securing object detectors against localized patch hiding attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, page 3177–3196, New York, NY, USA, 2021. Association for Computing Machinery.

[102] Zirui Xu, Fuxun Yu, and Xiang Chen. Lance: A comprehensive and lightweight cnn defense methodology against physical adversarial attacks on embedded multimedia applications. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 470–475. IEEE, 2020.

[103] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[104] Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. Minority reports defense: Defending against adversarial patches. In *International Conference on Applied Cryptography and Network Security*, pages 564–582. Springer, 2020.

[105] Kenneth T Co, Luis Muñoz-González, Leslie Kanthan, and Emil C Lupu. Real-time detection of practical universal adversarial perturbations. *arXiv:2105.07334*, 2021.

[106] Ping-Han Chiang, Chi-Shen Chan, and Shan-Hung Wu. Adversarial pixel masking: A defense against physical attacks for pre-trained object detectors. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21. ACM, 2021.

[107] Giulio Rossolini, Federico Nesti, Fabio Brau, Alessandro Biondi, and Giorgio Buttazzo. Defending from physically-realizable adversarial attacks through internal over-activation analysis. In *in AAAI Conference on Artificial Intelligence*, volume 37, 2023.

[108] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing.

[109] Jan Hendrik Metzen, Nicole Finnie, and Robin Hutmacher. Meta adversarial training against universal patches. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[110] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *European Conference on Computer Vision ECCV*, pages 429–448. Springer, 2020.

[111] Franziska Boenisch, Philip Sperl, and Konstantin Böttinger. Gradient masking and the underestimated robustness threats of differential privacy in deep learning. *arXiv preprint arXiv:2105.07985*, 2021.

[112] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[113] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. Certified defenses for adversarial patches.

[114] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[115] Alexander Levine and Soheil Feizi. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4585–4593, 2020.

[116] Cheng Yu, Jiansheng Chen, Youze Xue, Yuyang Liu, Weitao Wan, Jiayu Bao, and Huimin Ma. Defending against universal adversarial patches by clipping feature norms. In *in IEEE/CVF International Conference on Computer Vision*, 2021.

[117] Aaron Kane. Runtime Monitoring for Safety-Critical Embedded Systems. 2 2015.

[118] Nikita Bhardwaj Haupt and Peter Liggesmeyer. A runtime safety monitoring approach for adaptable autonomous systems. In Alexander Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security*, pages 166–177, Cham, 2019. Springer International Publishing.

[119] Geoffrey Nelissen, Humberto Carvalho, David Pereira, and Eduardo Tovar. Demo abstract: Runtime monitoring environments for real-time and safety critical systems. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–1, 2016.

[120] Geoffrey Nelissen, David Pereira, and Luís Miguel Pinho. A novel run-time monitoring architecture for safe and efficient inline monitoring. In Juan Antonio de la Puente and Tullio Vardanega, editors, *Reliable Software Technologies – Ada-Europe 2015*, pages 66–82, Cham, 2015. Springer International Publishing.

[121] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv: abs/1511.08458*, 2015.

[122] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA, 2014. Association for Computing Machinery.

[123] Nicolas Papernot and Patrick D. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *ArXiv: abs/1803.04765*, 2018.

[124] Lei Sun, Yuehan Wang, and Leyu Dai. Convolutional neural network protection method of lenet-5-like structure. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, CSAI '18, page 77–80, New York, NY, USA, 2018. Association for Computing Machinery.

[125] Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database, 2010.

[126] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv: abs/1708.07747*, 2017.

[127] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[128] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[129] Yiannis Kantaros, Taylor Carpenter, Sangdon Park, Radoslav Ivanov, Sooyong Jang, Insup Lee, and James Weimer. VisionGuard: Runtime Detection of Adversarial Inputs to Perception Systems. *arXiv e-prints*, February 2020.

[130] Giulio Rossolini, Alessandro Biondi, and Giorgio Buttazzo. Increasing the confidence of deep neural networks by coverage analysis. *IEEE Transactions on Software Engineering*, 49(2):802–815, 2023.

[131] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[132] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

[133] Carlos Ferreira and Gil Gonçalves. Remaining useful life prediction and challenges: A literature review on the use of machine learning methods. *Journal of Manufacturing Systems*, 63:550–562, 2022.

[134] Filip Došilović, Mario Brcic, and Nikica Hlupic. Explainable artificial intelligence: A survey. 05 2018.

[135] John X Morris, Chandan Singh, Alexander M Rush, Jianfeng Gao, and Yuntian Deng. Tree prompting: Efficient task adaptation without fine-tuning. *arXiv preprint arXiv:2310.14034*, 2023.

[136] Jie Song, Haofei Zhang, Xinchao Wang, Mengqi Xue, Ying Chen, Li Sun, Dacheng Tao, and Mingli Song. Tree-like decision distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13488–13497, June 2021.

[137] Sangwon Kim and Byoung Chul Ko. Neural tree decoder for interpretation of vision transformers. *IEEE Transactions on Artificial Intelligence*, pages 1–12, 2023.

[138] Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14933–14943, June 2021.

[139] Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. ViT-NeT: Interpretable vision transformers with neural tree decoder. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11162–11172. PMLR, 17–23 Jul 2022.

[140] Tsun-Hsuan Wang, Wei Xiao, Tim Seyde, Ramin Hasani, and Daniela Rus. Interpreting neural policies with disentangled tree representations. *arXiv preprint arXiv:2210.06650*, 2022.

[141] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E Gonzalez. Nbdt: neural-backed decision trees. *arXiv preprint arXiv:2004.00221*, 2020.

[142] Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

[143] Caglar Aytekin. Neural networks are decision trees. *arXiv preprint arXiv:2210.05189*, 2022.

[144] Gesina Schwalbe and Bettina Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery*, pages 1–59, 2023.

[145] Haoling Li, Jie Song, Mengqi Xue, Haofei Zhang, Jingwen Ye, Lechao Cheng, and Mingli Song. A survey of neural trees. *arXiv preprint arXiv:2209.03415*, 2022.

[146] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[147] Francesco Crecchi, Marco Melis, Angelo Sotgiu, Davide Bacciu, and Battista Biggio. Fader: Fast adversarial example rejection. *Neurocomputing*, 470:257–268, 2022.

[148] Ertunc Erdil, Krishna Chaitanya, Neerav Karani, and Ender Konukoglu. Task-agnostic out-of-distribution detection using kernel density estimation. In *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Perinatal Imaging, Placental and Preterm Image Analysis: 3rd International Workshop, UNSURE 2021, and 6th International Workshop, PIPPI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 3*, pages 91–101. Springer, 2021.

[149] Angelo Sotgiu, Ambra Demontis, Marco Melis, Battista Biggio, Giorgio Fumera, Xiaoyi Feng, and Fabio Roli. Deep neural rejection against adversarial examples. *EURASIP Journal on Information Security*, 2020:1–10, 2020.

[150] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Testing deep neural networks. *arXiv preprint arXiv:1803.04792*, 2018.

[151] Anonymous. Neuron activation coverage: Rethinking out-of-distribution detection and generalization. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review.

[152] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy*, pages 2774–2783. IEEE Computer Society, 2017.

[153] Andreas Bär, Jonas Löhdefink, Nikhil Kapoor, Serin Varghese, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. The vulnerability of semantic segmentation networks to adversarial attacks in autonomous driving: Enhancing extensive environment sensing. *IEEE Signal Process. Mag.*, 38(1):42–52, 2021.

[154] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[155] Shervin Minaee, Yuri Y. Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[156] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[157] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420. Springer, 2018.

[158] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 321–331, 2020.

[159] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[160] Norman Mu and Justin Gilmer. MNIST-C: A Robustness Benchmark for Computer Vision. *arXiv e-prints*, page arXiv:1906.02337, June 2019.

[161] Maura Pintor, Daniele Angioni, Angelo Sotgiu, Luca Demetrio, Ambra Demontis, Battista Biggio, and Fabio Roli. Imagenet-patch: A dataset for benchmarking machine learning robustness against adversarial patches. *Pattern Recognition*, 134, 2023.

[162] Brett Jefferson and Carlos Ortiz Marrero. Robust assessment of real-world adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 792–793, 2020.

[163] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, page 293–308, New York, NY, USA, 2020. Association for Computing Machinery.

[164] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358. IEEE, 2020.

[165] Federico Nesti, Giulio Rossolini, Gianluca D'Amico, Alessandro Biondi, and Giorgio Buttazzo. Carlagear: a dataset generator for a systematic evaluation of adversarial robustness of vision models. *arXiv preprint arXiv:2206.04365*, 2022.

[166] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[167] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[168] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[169] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2980–2988, 2017.

[170] Doyeon Kim, Woonghyun Ga, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth. *arXiv preprint arXiv:2201.07436*, 2022.

[171] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4009–4018, June 2021.

[172] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN Based 3D Object Detection for Autonomous Driving. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7636–7644, Long Beach, CA, USA, June 2019. IEEE.

[173] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks.

[174] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14521–14530, 2020.

[175] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[176] Piyush Satti, Nikhil Sharma, and Bharat Garg. Min-max average pooling based filter for impulse noise removal. *IEEE Signal Processing Letters*, 27:1475–1479, 2020.

[177] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng (Polo) Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2019.

[178] Giulio Rossolini, Federico Nesti, Fabio Brau, Alessandro Biondi, and Giorgio Buttazzo. Defending from physically-realizable adversarial attacks through internal over-activation analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:15064–15072, Jun. 2023.

[179] Giulio Rossolini, Alessandro Biondi, and Giorgio Buttazzo. Attention-based real-time defenses for physical adversarial attacks in vision applications. *arXiv preprint arXiv:2311.11191*, 2023.

[180] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[181] Mark Lee and J. Zico Kolter. On physical adversarial patches for object detection. *CoRR*, abs/1906.11897, 2019.

[182] Yonggan Fu, Shunyao Zhang, Shang Wu, Cheng Wan, and Yingyan Lin. Patch-fool: Are vision transformers always robust against adversarial perturbations? In *International Conference on Learning Representations*, 2021.

Ringraziamenti

Ho sempre creduto che la ricerca possa essere vissuta come un'espressione di creatività scientifica. Gli anni trascorsi hanno rafforzato questa mia convinzione, formandomi sia professionalmente che caratterialmente. Mi hanno permesso di visitare nuovi luoghi e conoscere nuove persone, di chiarire i miei obiettivi e il mio futuro, e di insegnarmi ad affrontare sfide anche in contesti difficili. Quindi, desidero ringraziare tutti coloro che, in questo cammino, hanno creduto in me e mi hanno offerto occasioni per migliorarmi e per mettermi alla prova.

Oltre agli obiettivi professionali raggiunti, desidero esprimere un grande ringraziamento agli amici e al gruppo con cui ho condiviso questi anni. Anche nei momenti di stress lavorativo, hanno reso le mie giornate ricche di momenti felici e indimenticabili, facendomi sentire orgoglioso del percorso intrapreso. Ho avuto la fortuna di far parte di un gruppo che mi ha permesso di essere semplicemente me stesso, sia dentro che fuori dal laboratorio, facendomi sentire parte di qualcosa che va oltre i semplici concetti di 'ambiente di lavoro' e 'colleghi'.

Un grande ringraziamento va anche alla mia famiglia, per avermi sempre sostenuto, offrendomi la libertà di scegliere la mia strada e supportandomi anche nei momenti più particolari e distanti che, purtroppo, spesso mi caratterizzano.

Pubblicato nel dicembre 2025
Società Italiana di Intelligence
SOCINT Press
https://press.socint.org

Direzione editoriale: Alice Felli

979-12-80111-73-9