On the Real-World Adversarial Robustness of Real-Time Semantic Segmentation Models for Autonomous Driving

Giulio Rossolini, Federico Nesti, Gianluca D'Amico, Saasha Nair, Alessandro Biondi, *Member, IEEE*, and Giorgio Buttazzo, *Fellow, IEEE*

***Abstract*— The existence of real-world adversarial examples (RWAEs) (commonly in the form of patches) poses a serious threat for the use of deep learning models in safety-critical computer vision tasks such as visual perception in autonomous driving. This article presents an extensive evaluation of the robustness of semantic segmentation (SS) models when attacked with different types of adversarial patches, including digital, simulated, and physical ones. A novel loss function is proposed to improve the capabilities of attackers in inducing a misclassification of pixels. Also, a novel attack strategy is presented to improve the expectation over transformation (EOT) method for placing a patch in the scene. Finally, a state-of-the-art method for detecting adversarial patch is first extended to cope with SS models, then improved to obtain real-time performance, and eventually evaluated in real-world scenarios. Experimental results reveal that even though the adversarial effect is visible with both digital and real-world attacks, its impact is often spatially confined to areas of the image around the patch. This opens to further questions about the spatial robustness of real-time SS models.**

***Index Terms*— Adversarial defenses, autonomous driving, real-world adversarial attacks, semantic segmentation (SS).**

I. Introduction

DEEP neural networks have reached super-human performance in many computer vision tasks. However, their results are typically threatened by a large set of inputs known as adversarial examples [1], [2]. The existence of adversarial examples is an empirical proof of the poor robustness of deep learning models, showing that imperceptible perturbations can easily corrupt the expected model behavior.

When dealing with cyber-physical systems such as autonomous cars and robots, several practical constraints should be taken into account when evaluating the relevance of digital perturbations injected in the image preprocessing system pipeline to the end of attacking deep learning algorithms. These constraints have raised several questions concerning the real criticality of adversarial perturbations in these application domains [3]. For this reason, in such domains, particular attention has been devoted to *real-world attacks*, which represent a more serious threat due to their capability of introducing adversarial effects by means of physically realizable objects.

In this field of study, most of the previous literature on adversarial robustness focused on image classification and object detection tasks, whereas fewer works have addressed semantic segmentation (SS). However, SS models play a key role in the visual perception pipeline used in autonomous driving, and thus evaluating their robustness is crucial. To this end, physically realizable adversarial attacks should be seriously considered in the design of robust vision models for autonomous driving: urban areas are typically filled with advertisement billboards, or road signs, each of which could become a potential attackable surface.

When dealing with these application domains, a proper evaluation of the system robustness should address several important factors, such as: 1) the concrete effectiveness of physically realizable adversarial objects for different targets and scenarios; 2) the transferability of the attack scheme from/to virtual (simulated) realistic environments; and 3) the design of efficient defenses with limited computational cost, which are crucial in self-driving applications that are notoriously subject to strict timing constraints.

This article faces all these challenges and extends a preliminary evaluation carried out in [6] with the following contributions. First, the work provides a flexible attack pipeline that enables the use of *multiple patches* and two different optimization paradigms to craft adversarial objects based on targeted or untargeted attack settings.

Second, based on the limitations of the standard pixelwise cross-entropy (CE) loss function for attacking regions of the image that are geometrically far from the patch, this work provides a proper definition and evaluation of a new loss function formulation, which was only briefly suggested in [6].

Finally, a novel fast adversarial patch detection method is proposed and evaluated in both the digital and real-world scenarios to understand the detectability of these physical attacks and the corresponding timing costs for SS tasks.

Manuscript received 16 January 2022; revised 7 October 2022 and 14 April 2023; accepted 3 September 2023. Date of publication 2 October 2023; date of current version 3 December 2024. This work was supported in part by the Project SERICS through the Ministero dell'Università e della Ricerca (MUR) National Recovery and Resilience Plan funded by the European Union—NextGenerationEU under Grant PE00000014. *(Corresponding author: Giulio Rossolini.)*

The authors are with the Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy (e-mail: giulio.rossolini@santannapisa.it; federico.nesti@santannapisa.it; gianluca.damico@santannapisa.it; saasha.nair@santannapisa.it; alessandro.biondi@sssup.it; giorgio@sssup.it).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2023.3314512, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2023.3314512

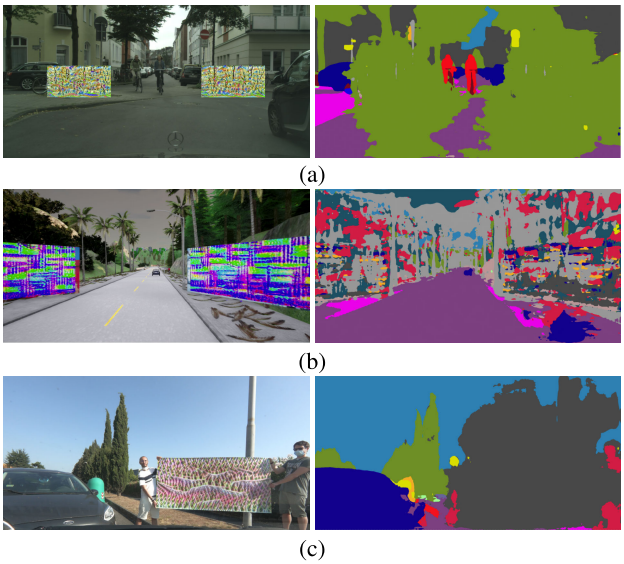


Fig. 1. Examples of effective patch-based attacks to scenes from (a) Cityscapes [4], (b) CARLA simulator [5], and (c) real-world environment. On the left-hand side, the figure shows three input images having adversarial patches, which are applied digitally, virtually, and physically. On the right–hand side, it reports the corresponding SS predictions obtained with DDRNet, BiSeNet, and ICNet, respectively.

Real-world datasets such as Cityscapes and custom realistic scenarios are hence considered for evaluation. Furthermore, to enable a scalable testing, similar as possible to real-world scenarios, we used the CARLA simulator [5], which provides fully controllable environments where billboards can be used as attackable surfaces. The geometric information derived from CARLA is used to perform a novel attack, named *scene-specific* attack, which improves the expectation over transformation (EOT) formulation [7]. Fig. 1 shows a few examples of successful attacks on these scenarios.

In summary, this work extends and refines the preliminary analysis in [6] by unifying the following list of contributions.

1) It proposes a general attack pipeline that enables *multipatch attacks* with both the targeted and untargeted schemes against SS models. This formulation also enables *scene-specific attacks* using geometric information available from the CARLA simulator.
2) It proposes a novel loss function formulation that improves the capabilities of attackers with respect to the use of the standard CE loss.
3) It extends a state-of-the-art adversarial patch detection method to cope with real-time constraints in SS tasks.
4) It presents an extensive evaluation of real-world attacks and defenses for a set of real-time SS models using: 1) realistic images from the Cityscapes dataset; 2) synthetic images from CARLA; and 3) real-world scenarios integrating physical adversarial objects.

The remainder of this article is organized as follows: Section II introduces the related work, Section III presents the proposed methods and algorithms, Section IV discusses the experimental results, and Section V concludes this article.

II. Related Work

The literature related to the topics addressed in this work is vast. Therefore, it has been categorized into: 1) real-time SS models; 2) adversarial studies on SS; 3) real-world adversarial attacks; and 4) defenses against adversarial patches.

A. Real-Time SS Models

Early works on SS architectures, such as the fully convolutional network (FCN) [8] and the UNet [9], were extended with methods that exploit spatial and context information of images: dilated convolutions, pyramid pooling modules, transformers, very deep layers, etc. [10], [11], [12], [13].

Although such models show high accuracy, they require powerful hardware platforms and imply large execution times when processing high-resolution images [14], making them not suited for real-time vision applications such as autonomous driving. To overcome this issue, several works [14], [15], [16], [17], [18] have recently proposed lightweight models that show acceptable accuracy on high-resolution images while taking into account real-time requirements. Among the vast number of available models, we restricted our attention to three popular real-time models: ICNet, DDRNet, and BiSeNet (see Section IV for the experimental settings and the supplementary materials for additional details).

The reason for selecting these models is to evaluate the robustness of several distinct approaches. As it is well-known in the literature, the majority of real-time SS models leverage two or more parallel paths for extracting context and spatial information [15], [19]. However, the internal operations they adopt are different and some of them can be more prone to adversarial attacks. ICNet is one of the first real-time SS models that merges multiple paths with cascade features fusion units [14]. Although nowadays several models reach faster performance, ICNet is considered an important milestone that is still taken as a reference in recent studies (e.g., [20], [21]). BiSeNet and DDRNet are more recent models with better real-time performance. Both are based on splitting the extraction of context and spatial information between two parallel branches, but BiSeNet exploits self-attention modules, while DDRNet involves multiple residual points between the branches.

B. Adversarial Studies on SS

Previous works [20], [22], [23], [24], [25], [26], [27], [28] proved that both the targeted and untargeted pixel-based perturbations easily fool SS models by extending well-known adversarial strategies (e.g., [1], [29]) from image classification. Consequently, Nakka and Salzmann [27] presented an interesting study on the robustness of SS models on autonomous driving datasets by showing that it is possible to perturb a precise area of pixels to change SS prediction corresponding to specific objects placed in the whole image.

Although the robustness of these convolutional architectures appears fragile at a first glance, the above works do not take two important aspects into consideration. First, they evaluated the adversarial effect against non-real-time networks only. From our point of view, this leads to unrealistic assessments,

since real-time architectures usually do not rely on the rich sequence of modules used by the evaluated networks, which could be practically prone to adversarial attacks [27]. Second, and most importantly, they did not consider real-world adversarial objects, which might represent a real threat for driving scenarios.

Differently from the strategies adopted by previous works, we believe that evaluating the robustness of a model for autonomous driving requires: 1) a precise selection of the tested models and 2) considering real-world adversarial examples (RWAEs). In fact, digital adversarial perturbations are not reproducible in the real-world, since they can be used to attack an autonomous driving perception pipeline only when the attacker has control on the digital representation of the image. If this is assumed, more effective system-level attacks may be used, hence reducing the relevance of adversarial attacks to SS models. Therefore, this work considers RWAEs as malicious objects that, once placed into a physical environment, are capable of externally injecting their adversarial effect.

C. Real-World Adversarial Attacks

Consistently with the latter observation, multiple works studied RWAEs for DNNs. Kurakin et al. [30] introduced a strategy for crafting physical-world adversarial pictures against image classification models, without the ability to manipulate the digital representation of inputs. Their work opened to a new class of adversarial examples called RWAEs. Athalye et al. [7] improved the robustness of RWAEs by proposing the EOT algorithm, which accounts for transformations that are typical of real-world scenarios (acquisition noise, changes in the point of view, etc.) during the optimization process. This allows crafting objects that retain the adversarial property when capturing images from different points of views.

Consequently, the EOT formulation opened to the development of adversarial patches [31], which are robust, localized, image-agnostic features that fool neural networks when placed in the input scene. Adversarial patches have been largely involved in the literature as a versatile tool to understand the practical robustness of DNNs. In fact, their real-world utilization involves printed objects that inject the adversarial effect directly from the external environment, rather than perturbing the digital pixel representation in the input image, as for common adversarial perturbations.

Although extensive prior work presented patch-based physical attacks for image classification [31], [32], [33], object detection [34], [35], [36], [37], optical flow [38], LiDAR object detection [39], and depth estimation [40], only a few focused on autonomous driving tasks. One reason might be that testing the adversarial robustness in autonomous driving context is more challenging, as it requires a certain control on the driving environments. Other works [35], [37] have shown autonomous driving simulators, and CARLA [5] in particular, to be a viable solution in alleviating this issue by crafting and evaluating adversarial situations in virtual 3-D environments.

Note that all the works mentioned above investigated the effect of adversarial objects on tasks different from SS. The first study addressing the evaluation of the robustness of real-time SS models against real-world adversarial attacks was proposed in [6], where both the Cityscapes dataset [4] and CARLA were used to investigate the effect of adversarial patches in the real world. However, no experiments were reported to evaluate the robustness of SS models against multipatch scenarios and targeted attacks. Also, no defense strategies were proposed to mitigate adversarial vulnerability at runtime. This present article fills these gaps by presenting novel attack objectives, additional experiments in multipatch scenarios, and a real-time adversarial detection mechanism.

D. Defense Mechanisms Against Adversarial Patches

The literature presents a wide set of defense mechanisms against adversarial attacks. While the methods used to detect generic adversarial examples are vast majority [3], [41], [42], [43], other works addressed the problem of detecting patches.

Some works [44], [45], [46], [47] exploit gradient-masking or adversarial training strategies to increase the robustness of the model against adversarial patches. Although such mechanisms help reduce the adversarial effect induced by patches, they do not provide any strategy to detect attacked images that are still dangerous for the tested models. As such, this class of defense methods is not considered for comparison.

Methods that detect images attacked with adversarial patches are based on masking and occlusion strategies [48], [49], [50], [51], [52] and run-time statistical analysis [53], [54]. Although both these approaches achieve high detection performance, most of them introduce significant latency during the model inference and hence are not suitable for real-time applications. Furthermore, note that most of the works mentioned in this section are designed to operate on image classification and object detection models, and only some of them can be adapted to SS architectures.

To the best of our knowledge, the only method that can be directly extended to cope with SS and is capable of providing real-time performance in detecting images affected by adversarial patches is HyperNeuron (HN) [53]. HN is based on detecting the overactivation of internal neurons, which is a symptom of the presence of an adversarial patch.

In this present article, we propose an extension of the HN algorithm that further improves the detection performance in the SS domain. First, a feature compression step is introduced to reduce the number of features to be processed at run time. This is of crucial importance when dealing with SS models, where the features' space may have high dimensions. Second, the selection of overactivated features is simplified using a threshold computed off-line rather than expensive operations performed at run-time. Based on these ideas, Section III presents a novel method that achieves similar performance to HN, but with significantly lower latency, making it more suitable for real-time applications.

E. This Work

This work provides a comprehensive study of the robustness of real-time SS models applied to autonomous driving scenarios. This is accomplished by defining and evaluating several

real-world adversarial patch attacks, a novel loss function formulation, and a real-time patch detection mechanism.

Such an extensive evaluation is missing in the literature and represents a promising milestone to better understand the practical robustness of SS models in real-world autonomous driving scenarios.

III. Patch-Based Attack Pipeline

This section presents the adopted notation, the attack pipeline used to generate real-time adversarial attack to SS models, the proposed loss function formulation, and the fast patch detection algorithm (FPDA).

A. Preliminaries

We consider input images with height H, width W, and C channels, denoted as $x \in [0,1]^{H\times W\times C}$. An SS model discriminating between N_c classes is thus represented by a function $f : x \mapsto [0,1]^{(H\cdot W)\times N_c}$, which gives the predicted class-probability scores associated with each image pixel i. More specifically, the predicted probability score for pixel i corresponding to class j is denoted by $f_i^j(x) \in [0,1]$.

The predicted SS of each pixel i, denoted by $\mathrm{SS}_i(x)$, is then computed by extracting the class with the highest probability score of the pixel, i.e., $\mathrm{SS}_i(x) = \mathrm{argmax}_{j\in\{1,\dots,N_c\}} f_i^j(x)$. The complete SS prediction $\mathrm{SS}(x)$ is then the collection $\{\mathrm{SS}_i(x),\ i = 1,\dots,H\times W\}$.

The ground truth for the SS of x is defined as $y \in \mathbb{N}^{H\times W}$, and it assigns the correct class (in $\{1,\dots,N_c\}$) to each pixel.

An adversarial patch of height $\tilde{H}$ and width $\tilde{W}$ is denoted by $\delta \in [0,1]^{\tilde{H}\times\tilde{W}\times C}$, where $\tilde{H} < H$ and $\tilde{W} < W$.

In a multipatch setting, we consider a set of patches $\Delta = \{\delta_k : k = 1,\dots,N_p\}$, where N_p is the number of patches used for the attack.

B. General Attack Pipeline

All the patch-based attacks considered in this article are generated with the pipeline illustrated in Fig. 2.

Inspired by the universal (i.e., image-agnostic) attacks [55] and the EOT-based attacks [7], the objective is to find an optimal patch set Δ^* by optimizing a certain loss function $\mathcal{L}$ for all the patched images in expectation, according to the distribution of transformations used to apply the patch set Δ on the image set $\mathbf{X}$.

In particular, we define the following.

1) 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 the selected transformation is directly applied to the patches Δ. The typical parameters of these transformations are randomized during the optimization to make the patches robust to illumination changes and acquisition noise.
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 III-D provides details of the patch placement function.
3) A patch application function $g(x,\Delta,\Gamma_a,\eta)$ that replaces the area(s) of the image x specified by η with transformed versions of the patches in Δ obtained by a transformation randomly selected from Γ_a, hence returning the patched image $\tilde{x}$.
4) A loss function $\mathcal{L}$, which is the objective function to be optimized. $\mathcal{L}$ consists of a weighted sum of multiple loss subfunctions. The adversarial effect is obtained through the optimization of the adversarial loss $\mathcal{L}_{\mathrm{adv}}$ (detailed in Section III-E). To ensure that the patch transfers well to the real world, two additional losses are considered to account for the *physical realizability* of the patch: smoothness loss $\mathcal{L}_S$ and nonprintability score $\mathcal{L}_N$. Refer to the supplementary material for details.

The optimization problem can therefore be written as

$$\Delta^* = \underset{\Delta}{\mathrm{argmin}}\ \mathbb{E}_{x\in\mathbf{X},\zeta_a\in\Gamma_\mathbf{a},\eta}\ \mathcal{L}\big(f(\tilde{x}),y\big) \tag{1}$$

whereas its practical iterative implementation at step t is

$$\delta_{k,t+1} = \mathrm{clip}_{[0,1]}\left(\delta_{k,t} + \epsilon\cdot\sum_{x\in\mathbf{X}}\nabla_{\delta_{k,t}}\mathcal{L}\big(f(\tilde{x}),y\big)\right) \tag{2}$$

where $\tilde{x} = g(x,\Delta,\Gamma_a,\eta)$, $k = \{1,\dots,N_p\}$, and ϵ represents the step size.

Sections III-C–III-E detail the optimization objectives, the patch placement functions considered for the EOT-based and scene-specific attacks, and the loss functions used.

C. Untargeted and Targeted Attacks

The attacker's objective is encoded in the performed optimization to craft adversarial patches. The attacker might want to maximize the prediction error of the network, regardless of the output classes (*untargeted* attack), or force the network prediction toward a specific output (*targeted* attack).

The objective of an untargeted attack is then to maximize the loss function $\mathcal{L}_{\mathrm{adv}}(f(\tilde{x}),y)$, defined later in Section III-E, where y is the ground-truth label. Hence, in (2), $\mathcal{L}(f(\tilde{x}),y) = -\mathcal{L}_{\mathrm{adv}}(f(\tilde{x}),y)$ (the additional losses for physical realizability of patches are neglected for simplicity).

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), 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 [27], as illustrated in Fig. 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 $y_t = \tau(y, c_{\mathrm{attacked}}, c_{\mathrm{target}})$, where c_{attacked} might indicate either a specific class or the nearest neighbor approach (specified as a pseudoclass NN). The targeted attack is then performed by considering, in (2), $\mathcal{L}(f(\tilde{x}),y) = \mathcal{L}_{\mathrm{adv}}(f(\tilde{x}),y_t)$.

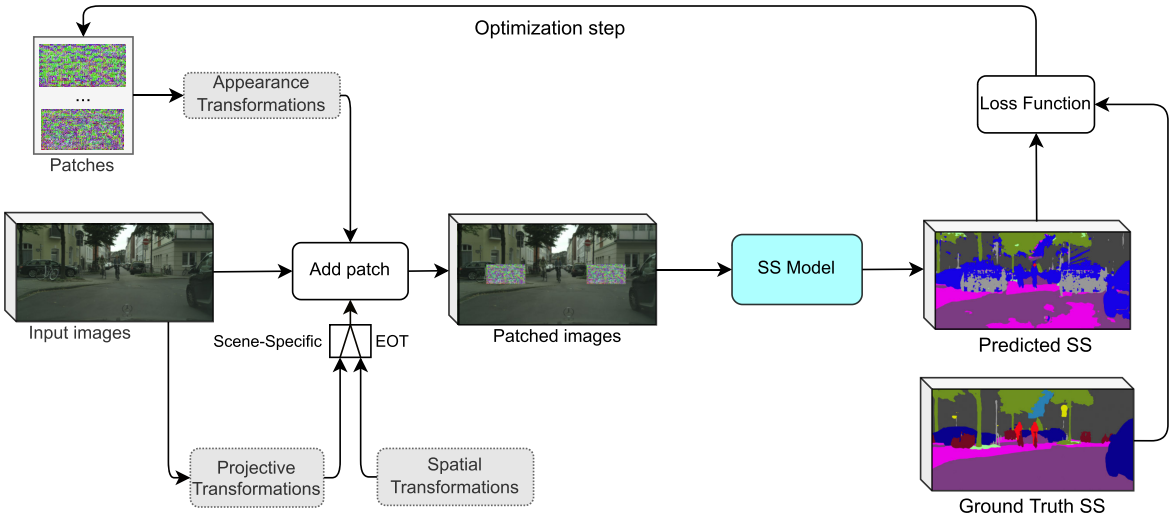


Fig. 2. Scheme of the proposed approach for crafting both the EOT-based and the scene-specific patches.

Fig. 3. Illustration of how the original label (left) can be modified to attack the class *pedestrian*: by changing *pedestrian* with *road* class (middle), or using the nearest neighbor approach (right).

D. EOT-Based and Scene-Specific Attack

The patch placement within the image might follow two different approaches: 1) randomizing the patch position, scale, and rotation at each iteration (i.e., using the EOT method) and 2) using accurate projective transformations.

While the first is a general-purpose method for real-world adversarial patch generation (hereby referred 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 3-D rototranslation (see supplementary materials) to warp the patch according to the point of view of the camera in each image of the dataset.

Formally, the function η in (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 it does not require randomization of the patch placement (saving time during optimization).

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 3-D 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. The transfer issues between CARLA and the real world will be investigated in a future work. Section IV provides a comparison of this method against the EOT-based attack.

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

E. Proposed Loss Function

The pixelwise CE loss, denoted by $\mathcal{L}_{\text{CE}}$, has been shown to work well for untargeted digital attacks by adding a perturbation r to pixels values [22], [27]. In this case, the loss is $\mathcal{L}_{\text{adv}}(f(x+r), y) = (1/|\mathcal{N}|) \cdot \sum_{i\in\mathcal{N}} \mathcal{L}_{\text{CE}}(f_i(x+r), y_i)$, where $f_i(x+r) \in [0,1]^{N_c}$ is the output of the model for each pixel (i.e., a probability distribution over N_c classes), and $\mathcal{N} = \{1, \ldots, H \times W\}$ is the entire set of pixels in $\tilde{x}$.

This formulation can be changed to generate stronger attacks against SS models. If $\tilde{\mathcal{N}}_k = \{1, \ldots, \tilde{H}_k \times \tilde{W}_k\} \subseteq \mathcal{N}$ denotes the pixels belonging to a patch δ_k only, the totality of the pixels belonging to the patches can be defined as $\tilde{\mathcal{N}} = \cup_{k=1}^{N_p} \tilde{\mathcal{N}}_k$.

The previous pixelwise CE loss computed on the subset of pixels $\mathcal{N} \setminus \tilde{\mathcal{N}}$ can be split into two terms

$$\mathcal{L}_M^{\tilde{x}} = \sum_{\substack{i\in\mathcal{N}\setminus\tilde{\mathcal{N}} \\ SS_i(\tilde{x})=y_i}} \mathcal{L}_{\text{CE}}\big(f_i(\tilde{x}), y_i\big), \quad \mathcal{L}_{\overline{M}}^{\tilde{x}} = \sum_{\substack{i\in\mathcal{N}\setminus\tilde{\mathcal{N}} \\ SS_i(\tilde{x})\neq y_i}} \mathcal{L}_{\text{CE}}\big(f_i(\tilde{x}), y_i\big) \tag{3}$$

where $\mathcal{L}_{\overline{M}}^{\tilde{x}}$ 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.

These 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:

$$\nabla_{\delta_k}\mathcal{L}_{\text{adv}}\big(f(\tilde{x}), y\big) = \gamma \cdot \frac{\nabla_{\delta_k}\mathcal{L}_M^{\tilde{x}}}{\left|\left|\nabla_{\delta_k}\mathcal{L}_M^{\tilde{x}}\right|\right|_2} + (1-\gamma) \cdot \frac{\nabla_{\delta_k}\mathcal{L}_{\overline{M}}^{\tilde{x}}}{\left|\left|\nabla_{\delta_k}\mathcal{L}_{\overline{M}}^{\tilde{x}}\right|\right|_2} \tag{4}$$

where $\gamma \in [0,1]$ is a balancing factor that determines whether the optimization should focus on decreasing the number of

correctly classified pixels or improving the adversarial strength for the currently misclassified pixels, and $k = \{1, \ldots, N_p\}$. In other words, γ balances between the importance of $\mathcal{L}_M$ and $\mathcal{L}_{\overline{M}}$ at each iteration t of the optimization problem in (2) depending on the number of pixels not yet misclassified.

To provide an automatic tuning of γ at each iteration, an adaptive value of $\gamma = (|\Upsilon|/|\mathcal{N} \setminus \tilde{\mathcal{N}}|)$ 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.

Note that this adaptive formulation requires knowing the ground-truth labels and the patch masks in $\tilde{x}$.

Section IV provides an extensive analysis of the proposed loss function by comparing multiple values of γ with the standard pixelwise CE measured both on $\mathcal{N} \setminus \tilde{\mathcal{N}}$ and $\mathcal{N}$ (which is used by [27]), suggesting that our formulation is indeed more general and effective for this kind of attacks.

F. Fast Patch Detection Algorithm

This section describes the FPDA designed to recognize SS predictions affected by adversarial patches. Inspired by the HN method [53], the basic idea of the proposed approach relies on identifying and counting the presence of overactivated internal features, commonly caused by adversarial attacks to induce erroneous predictions in deep learning models.

Before introducing the FPDA algorithm, some additional notation is required. As other deep learning models, the SS models are composed of a sequence of layers $(\ell_0, \ell_1, \ldots, \ell_L)$, from the input layer to the output layer, respectively.

Given an SS model f evaluated on an input image x, $v^{\ell}_{c_\ell, i_\ell, j_\ell}(x)$ represents the activation value of a neuron at layer ℓ, having position $(c_\ell, i_\ell, j_\ell) \in \{1, \ldots, C_\ell\} \times \{1, \ldots, H_\ell\} \times \{1, \ldots, W_\ell\}$, where C_ℓ, H_ℓ, and W_ℓ denote the number of channels, height, and width of the features at layer ℓ, respectively.

The pseudocode of FPDA is reported in Algorithm 1. The function $\texttt{getActivation}(f, x, \ell)$ (line 1) returns all the neuron activations at the layer ℓ, denoted by $A_\ell = \{v^{\ell}_{c_\ell, i_\ell, j_\ell}(x), \quad \forall c_\ell, i_\ell, j_\ell\}$. Since the number of neurons at layer ℓ might be large for SS models, making the next algorithmic steps computationally expensive and therefore not suited for real-time applications, a *feature compression* is performed at line 2, returning $\mathcal{C}_\ell = \{\max_{c_\ell \in C_\ell} |v^{\ell}_{c_\ell, i_\ell, j_\ell}(x)|, \quad \forall i_\ell, j_\ell\}$. This operation preserves the properties of the overactivated neurons by using the max operator.

Then, $\mathcal{C}_\ell$ 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). Here, $\hat{\mathcal{C}}_\ell$ denotes the normalized features.

To filter out all the common activations resulting from clean inputs, a subset of $\hat{\mathcal{C}}_\ell$ 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 $\hat{\mathcal{C}}_\ell$ of the previously defined clean dataset.

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

Algorithm 1 Fast Patch Detection Algorithm

Input: f, x, ℓ, μ_ℓ, σ_ℓ, θ_ν, ϱ
Output: Detection flag

$A_\ell = \texttt{getActivation}(f, x, \ell)$
$\mathcal{C}_\ell = \{ \max_{c_\ell \in C_\ell} |v^{\ell}_{c_\ell, i_\ell, j_\ell}(x)| , \quad v^{\ell}_{c_\ell, i_\ell, j_\ell}(x) \in A_\ell\}$
$\hat{\mathcal{C}}_\ell = (\mathcal{C}_\ell - \mu_\ell)/\sigma_\ell$
$\mathcal{S}_\ell = \hat{\mathcal{C}}_\ell.\texttt{where}(\hat{\mathcal{C}}_\ell > \theta_\nu)$
$score = \sum \mathcal{S}_\ell$
if $score > \varrho$ **then**
  Return **True**
end
Return **False**

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 detail in Section IV.

It is useful to highlight the main differences between the proposed method and HN. Although both achieve similar detection performance, our approach has a significantly smaller computation time, thanks to the feature compression step described above. Furthermore, while the feature 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.

Section IV presents exhaustive 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.

IV. Experimental Evaluation

This section presents the set of experiments carried out to evaluate the performance of the proposed attack and defense approaches. The code is available at https://github.com/retis-ai/SemSegAdvPatch.

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; the effects of untargeted single- and double-patch attacks are reported for the CARLA-generated images; finally, the detection results of FPDA are showed.

A. Experimental Setup

All the experiments were performed using PyTorch [56] and a set of eight NVIDIA-A100 GPUs, while the CARLA simulator was run on a system powered by an Intel Core i7 with 12-GB RAM and a GeForce GTX 1080 Ti GPU.

The optimizer of choice was Adam [57], with the learning rate empirically set to 0.5. The effect of the adversarial patches on the SS models was evaluated using the mean intersection-over-union (mIoU) and mean accuracy (mAcc) [58] on the subset of the image pixels not belonging to the patch.

1) Datasets: Several datasets were used for the experiments. The Cityscapes dataset [4] is one of the most common datasets of driving images for SS. 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 multipatch 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 multipatch), 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 rototranslations 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. 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 dataset and a validation dataset (denoted by "val") were also collected. Additional details on the fine-tuning process are in the supplementary material.

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 IV-G.

2) Models: Three real-time SS models suited for autonomous driving applications were used to evaluate the adversarial attacks investigated in this article, namely, DDRNet [15], BiSeNet [16], and ICNet [14]. These models were tested in two settings: one with Cityscapes, using the pretrained 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). Table I summarizes the performance of these models.

TABLE I

MIOU AND MACC OF THE TESTED MODELS ON CITYSCAPES (PRETRAINED) AND OUR CARLA DATASET (FINE-TUNED)

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

B. Effects of Untargeted Attacks on Cityscapes

In this section, 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 is the center of the left and right halves of the image. The patches were optimized over 200 epochs.

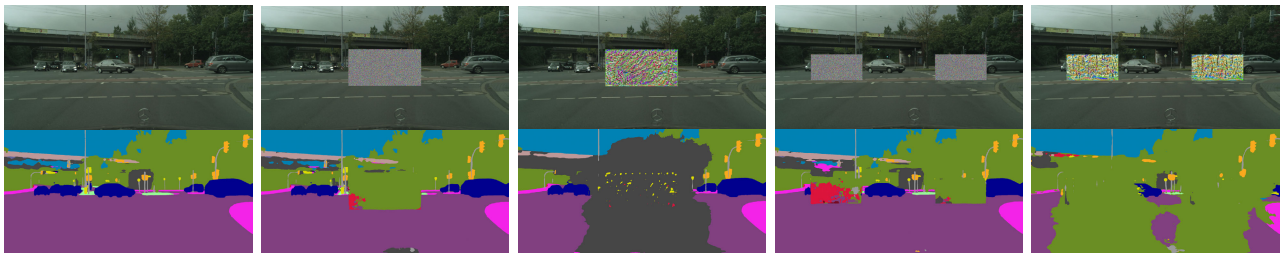
As shown in Table II, the double-patch formulation achieves, in general, higher attack performance with respect to the single-patch version. In particular, for DDRNet and ICNet, the double-patch attack gets a lower mIoU in all the tested sizes. Different considerations arise for BiSeNet, where patches 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 are provided in the supplementary material.

Fig. 4 shows the adversarial effect produced by the proposed attacks, illustrated for the single- (300 × 600) and double-patch formulation (212 × 424).

C. Effects of Targeted Attacks on Cityscapes

While the objective of an untargeted attack is to induce the maximum error in the network's prediction regardless of the classes that are predicted, a targeted attack must follow a much more constrained optimization process.

In particular, by pushing the network's prediction toward 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} (which is already more complex than an untargeted attack) but also to keep all the other pixels



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

Fig. 4. Images of the Cityscapes validation set and their SSs 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.

TABLE II

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 GRAY ROWS REFER TO DOUBLE PATCHES HAVING THE SAME AREAS OF THE SINGLE CONFIGURATION

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

untouched. 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).

Previous work [27] on targeted localized adversarial perturbations for SS models only deals 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 III 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.

TABLE III

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

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

We empirically assessed this argument by performing universal targeted attacks for the attacks defined above. Table III(b) presents the IoU of the attacked class for each attack and each network. 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. Fig. 5 illustrates some of the attacks performed during the evaluation.

D. 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 III-D, 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.

Based on preliminary experiments on the effect of the number of pixels and on the real-world dimension of the patch (described in the supplementary material), we decided to use

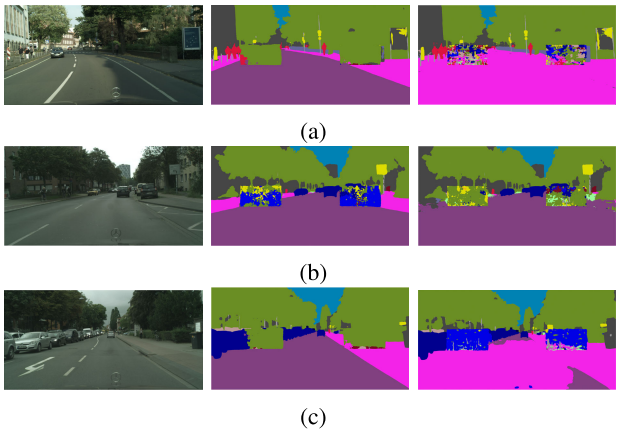


Fig. 5. Effect of targeted patches. (a) Image-specific road → sidewalk attack on DDRNet. (b) Image-specific sidewalk → **NN** attack on ICNet. (c) Universal road → sidewalk attack on DDRNet. First column is the original image, while middle and right columns are the predictions with random and adversarial patches, respectively.

patches of 150 × 300 pixels (corresponding to a real-world dimension of 3.75 × 7.5 m). 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 article 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 RWAEs in CARLA.

The third column of Table I reports the performance of the tested networks on the considered scenes (with no patch), while Table IV 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 IV, the scene-specific attack achieves better results than the EOT formulation from previous work. To better point out the higher effectiveness of the scene-specific attack formulation, which is not properly captured by the standard SS metrics, we provide further results in the supplementary material.

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 3.75 × 7.5 m). 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 3-D virtual scene of CARLA. Fig. 6 shows the effect of some representative attacks on CARLA images.

E. Evaluating the Proposed Loss Function and Parameters

The proposed loss function formulation presented in Section III-E was evaluated against the standard CE loss for several values of γ and for the different attacks.

Fig. 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 the cases, for each tested value of γ, our loss formulation outperforms the standard CE, both in terms of attack performance and convergence rate.

Fig. 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. 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 CE loss.

F. Detecting Robust Adversarial Patches

This section provides a set of experiments aimed at investigating the detection and timing performance of the FPDA.

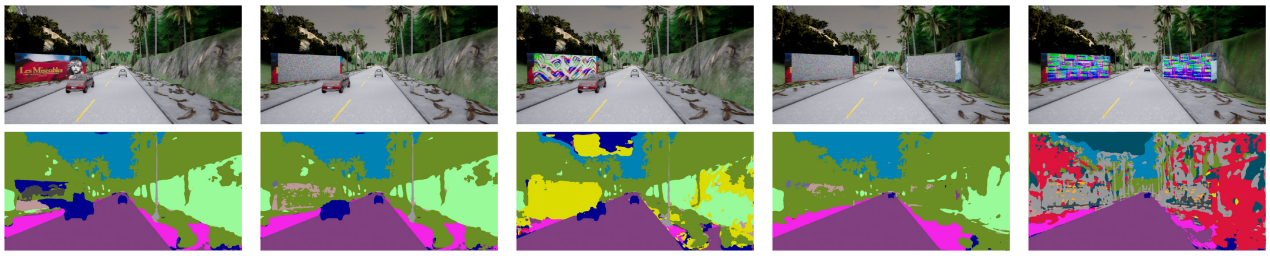
1) Tuning the Thresholds: The Cityscapes dataset is used to set the algorithm's parameters $\mu_\ell, \sigma_\ell, \nu$-*percentile*. In particular, a set of features $\mathcal{C}_\ell$ extracted from 1000 images of the original training set are used to compute μ_ℓ and σ_ℓ. Then, after normalization, the corresponding features $\hat{\mathcal{C}}_\ell$ 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. The choice of the layer has been the subject of many preliminary experiments. The real-time SS models considered in this article fuse information extracted from different parallel branches. The layer ℓ is chosen as the first layer that joins all the model's branches. In fact, 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. Moreover,

TABLE IV

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

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



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

Fig. 6. SSs 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.

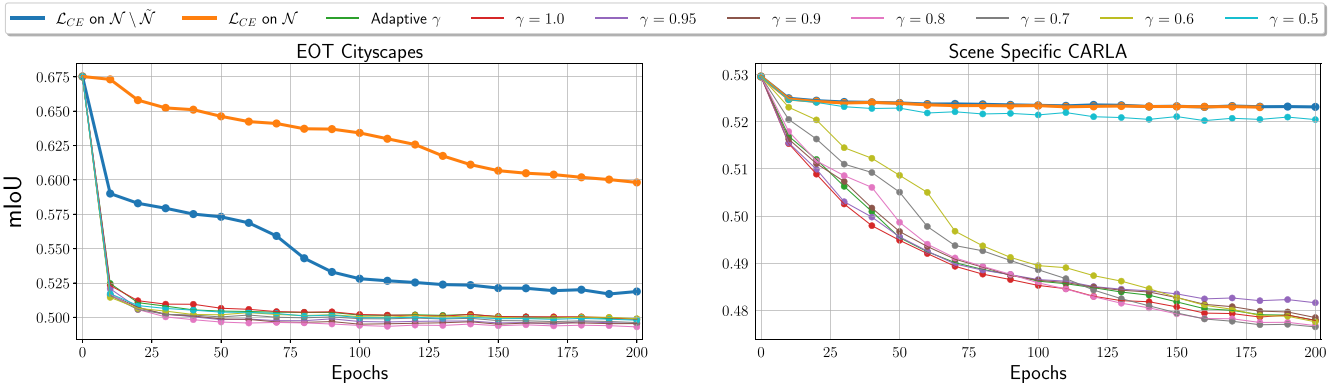


Fig. 7. Comparison of adversarial patch optimizations (200 × 400) on ICNet and Cityscapes using different loss functions: two versions of the standard pixelwise CE and our formulation with multiple values of γ. $\mathcal{L}_{\text{CE}}$ on $\mathcal{N}$ is the original version used by [27], while $\mathcal{L}_{\text{CE}}$ on $\mathcal{N} \setminus \tilde{\mathcal{N}}$ is an improved version based on the rationale presented in Section III-E.

we empirically found it to be the point with the best detection performance.

2) On the Validity of the Method: Fig. 9 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 Section IV-B are used (150 × 300, 200 × 400, and 300 × 600).

As shown in Fig. 9, 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 gray 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).

3) Timing Analysis: As already mentioned in Section III-F, the proposed method allows processing large-sized layer activations without affecting the timing performance of the model. Fig. 10 shows plots of 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 SS 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.

4) Defense-Aware Attack: Although the adversarial patches tested in Section IV-F2 are easily detectable, it is important to assess the goodness of the 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 article

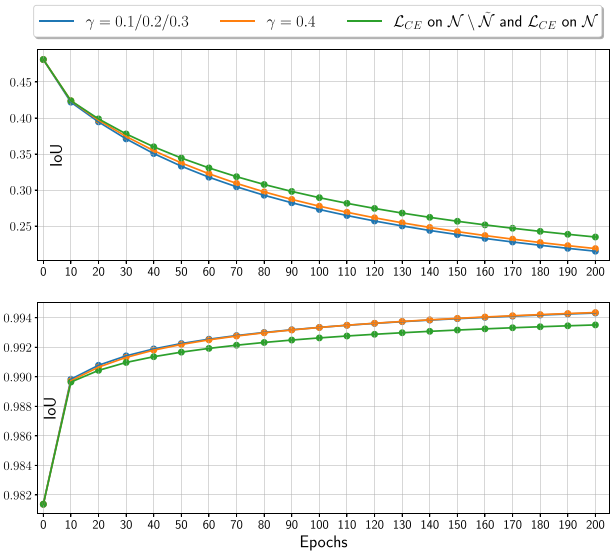


Fig. 8. Summary of the optimization process for an image-specific pedestrian → NN attack. The IoU values at each epoch are averaged over 100 different attacks. Top image shows the IoU of the class pedestrian, whereas bottom image shows the IoU of the class road. For each value of γ tested, our loss formulation outperforms the standard CE loss.

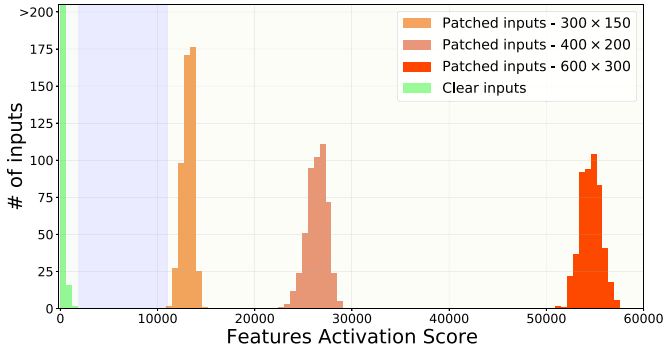


Fig. 9. Distribution of the clean and patched images as a function of the features overactivation *score* computed by Algorithm 1 with BiSeNet. The gray area highlights the set of ϱ thresholds that achieve a perfect detection accuracy with the tested images.

was conceived to craft untargeted adversarial patches with the intention of also deceiving the FPDA.

Since the FPDA is based on the detection of overactivated 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 overactivations). This is accomplished by solving the iterative optimization method discussed in Section III but using a loss function $\mathcal{L}(f(\tilde{x}), y)$ that includes both the adversarial loss function $\mathcal{L}_{\text{adv}}(f(\tilde{x}), y)$ (in short $\mathcal{L}_{\text{adv}}^{\tilde{x}}$) and an additional *activation loss* $\mathcal{L}_{\hat{\mathcal{C}}_\ell}^{\tilde{x}}$. The former is the loss function introduced in Section III, while the latter is a new loss function that returns a cost proportional to the squared overactivation of the normalized features $\hat{\mathcal{C}}_\ell$ (computed by performing the first operations of Algorithm 1 on $\tilde{x}$), i.e., $\mathcal{L}_{\hat{\mathcal{C}}_\ell}^{\tilde{x}} = ||\hat{\mathcal{C}}_\ell||_2^2$.

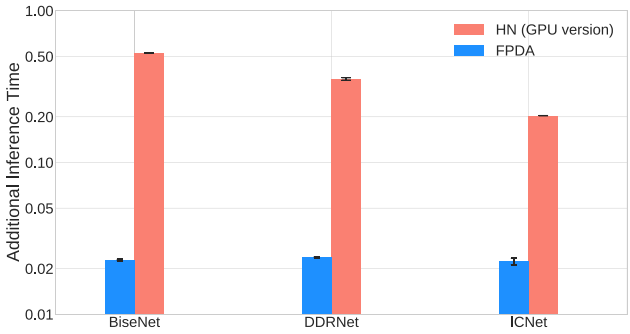


Fig. 10. Additional inference time computed as the ratio between the execution time of the detection algorithm and the original model inference time.

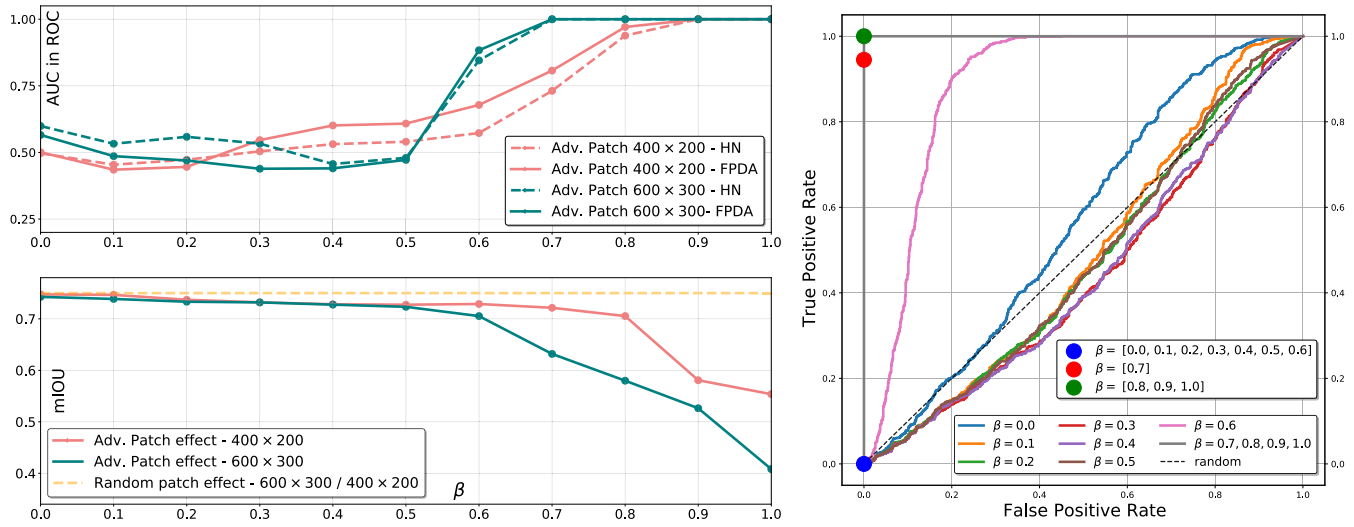
Therefore, the gradient of $\mathcal{L}(f(\tilde{x}), y)$ used during the optimization method was redefined as

$$\nabla_{\delta_k}\mathcal{L}\left(f\left(\tilde{x}\right), y\right) = \beta \cdot \frac{\nabla_{\delta_k}\mathcal{L}_{\text{adv}}^{\tilde{x}}}{\left|\left|\nabla_{\delta_k}\mathcal{L}_{\text{adv}}^{\tilde{x}}\right|\right|_2} - (1-\beta) \cdot \frac{\nabla_{\delta_k}\mathcal{L}_{\hat{\mathcal{C}}_\ell}^{\tilde{x}}}{\left|\left|\nabla_{\delta_k}\mathcal{L}_{\hat{\mathcal{C}}_\ell}^{\tilde{x}}\right)\right|\right|_2} \tag{5}$$

where β is a parameter introduced to balance the importance of $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\hat{\mathcal{C}}_\ell}$, and $k = \{1, \ldots, N_p\}$. 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 relationship between detectability and adversarial effect. To this end, several adversarial patches (300 × 600 and 200 × 400 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 (TPR) and false positive rate (FPR) 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.

Fig. 11(a) 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 there exists an intrinsic relationship between the adversarial effect and the overactivation of features, such that highly effective patches are more prone to be detected by the proposed strategy, as also observed by previous works [53].



(a) (b)

Fig. 11. (a) Comparison between the detectability and adversarial effect of multiple patches crafted changing the β parameter. (b) ROC curves corresponding to all the previous tested adversarial patches and their TFPs/FPRs corresponding to a unique threshold selected with $\beta = 0.8$. These results were obtained with the ICNet model.

Fig. 11(b) illustrates another important result, showing all the ROC curves computed for the 300 × 600 patches [their AUC score is the one shown in Fig. 11(a)]. By looking at these curves, it is possible to figure out the TPR and FPR with respect to a fixed threshold ϱ. The threshold ϱ is set as the optimal value (cutoff 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 the supplementary material).

G. Adversarial Patches in the Real World

This section evaluates the effectiveness of the proposed attack pipeline and the detection algorithm in the real world.

1) Effect of a Real-World Patch: To prove the effectiveness of the attack pipeline proposed in Section III, 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 (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 pretrained version of ICNet (since it showed good performance also on our personal real-world dataset). Fig. 12(a) shows a sequence of frames recorded while moving in the direction of the adversarial patch, printed as a 1 × 2-m poster.

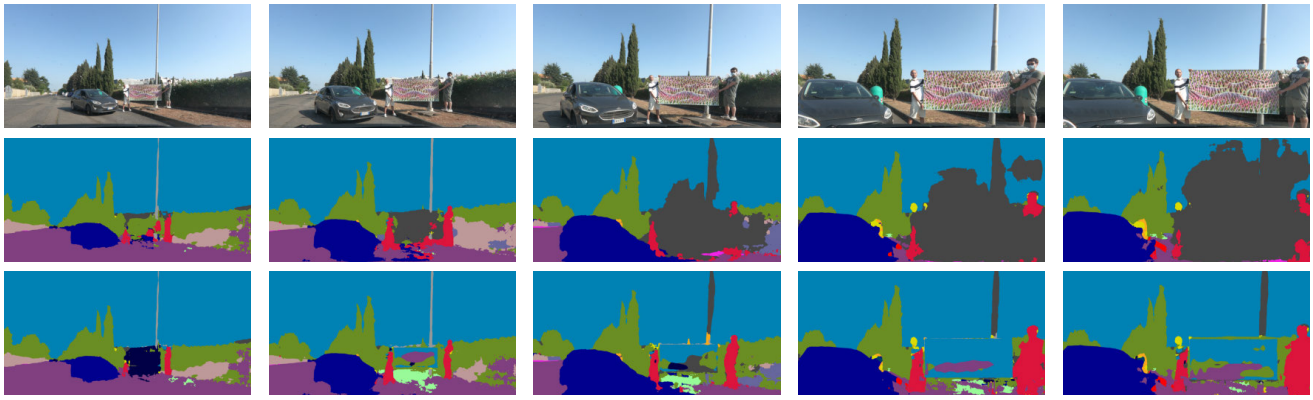
The results of further analyses are provided in the supplementary material, remarking how the optimized patch alters a significant area of the predicted SS, while the random patch does not with portions of the image far from its position that are not affected. Furthermore, the attack performance increases 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 urban corner that shows good performance and is not crowded with moving 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.

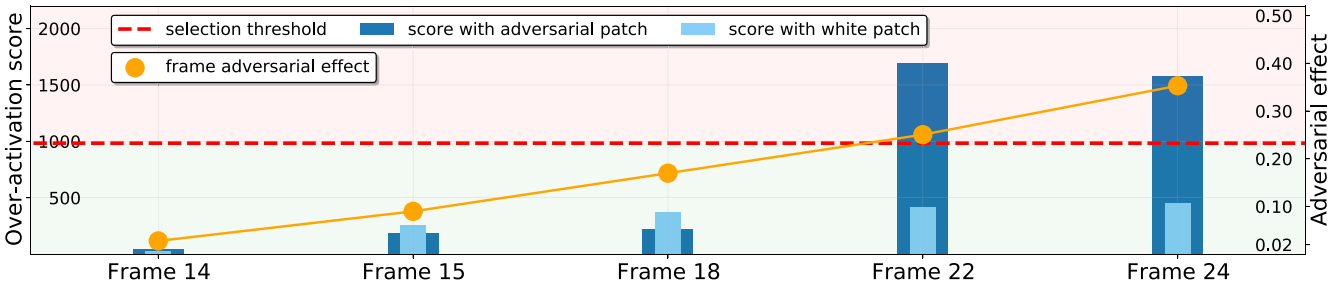
2) Detectability of a Real-World Patch: Fig. 12(b) reports the *overactivation score* corresponding to each frame shown in Fig. 12(a). The red dashed line represents the threshold ϱ, used to distinguish safe and unsafe predictions. To provide a clear visualization of the spatial effect caused by the printed adversarial patch [second row in Fig. 12(a)], we also report the SS obtained from the same images where the adversarial patch was masked with white pixels [third row in Fig. 12(a)] and their corresponding scores [light blue in Fig. 12(b)].

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 $1 - (1/|\mathcal{N} \setminus \tilde{\mathcal{N}}|) \sum_{i \in \mathcal{N} \setminus \tilde{\mathcal{N}}} (y_i == SS_i(x))$. 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 IV-B, 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).

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 Fig. 11(b)]. 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



(a)



(b)

Fig. 12. Real-world evaluation on ICNet of the attack method and the defense algorithm. (a) First row contains the original images with the printed adversarial patch, while the second row contains their obtained SSs. The third row instead shows the outcomes obtained by masking the patch areas of the previous inputs with white pixels. (b) Overactivation score of the corresponding frames for the SS predictions in rows 2 and 3, and which of them are detected 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.

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 overactivation score implied by patches transferred in the real world.

As can be noted from Fig. 12(b), 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 overactivations 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 overactivated 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. Therefore, the corresponding safe classifications could be acceptable. However, also some notable adversarial effects could overcome the defense mechanism, as, for instance, for frame 18, which has several areas affected by the patch effect. Improving the performance of the detection mechanism, while also providing deeper analysis on its transferability on real-world critical situations, is left as future work.

V. Conclusion

This article presented an extensive study of the real-world adversarial robustness of real-time SS models for autonomous driving scenarios. Preliminary results achieved in [6] have been extended in this work by generalizing the attack pipeline for EOT and *scene-specific attack* to both targeted and untargeted settings and double-patch configurations. All the proposed formulations leveraged a novel loss function that improves the state-of-the-art methods for optimizing adversarial patches. Finally, a novel real-time detection algorithm (FPDA) was presented and evaluated, proving its capability of quickly detecting dangerous adversarial patches on realistic driving scenarios.

The extensive experimentation performed in this article showed how real-time SS models present a certain robustness to real-world adversarial attacks for driving scenarios. Moreover, it showed that the proposed detection algorithm is able to drastically reduce the threat associated with those attacks.

Since many variables of a realistic driving environment are not controllable (e.g., weather, external objects), future work aims at studying the transferability of robust assessments derived from virtual scenarios (e.g., CARLA simulator) to real-world environments. Furthermore, the proposed detection mechanism highlights a strong relationship between the area attacked from an adversarial patch and the overactivation of internal features. This suggests the need for a deeper investigation on the spatial robustness of SS models.

Acknowledgment

This work was partially supported by project SERICS (PE00000014) under the MUR (Ministero dell'Università e della Ricerca) National Recovery and Resilience Plan funded by the European Union—NextGenerationEU.

REFERENCES

[1] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, Apr. 2014, pp. 1–10.

[2] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[3] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.

[4] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 3213–3223.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, Mountain View, CA, USA, Nov. 2017, pp. 1–16.

[6] 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," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 2826–2835.

[7] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 284–293.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI*. Cham, Switzerland: Springer, 2015.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.

[12] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K.: Springer, Aug. 2020, pp. 173–190.

[13] K. Sun et al., "High-resolution representations for labeling pixels and regions," 2019, *arXiv:1904.04514*.

[14] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2018, pp. 418–434.

[15] Y. Hong, H. Pan, W. Sun, and Y. Jia, "Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes," 2021, *arXiv:2101.06085*.

[16] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2018, pp. 334–349.

[17] R. Gao, "Rethink dilated convolution for real-time semantic segmentation," 2021, *arXiv:2111.09957*.

[18] X. Li et al., "Semantic flow for fast and accurate scene parsing," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Nov. 2020, pp. 775–793.

[19] I. Papadeas, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis, "Real-time semantic image segmentation with deep learning for autonomous driving: A survey," *Appl. Sci.*, vol. 11, no. 19, p. 8802, Sep. 2021.

[20] A. Arnab, O. Miksik, and P. H. S. Torr, "On the robustness of semantic segmentation models to adversarial attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 888–897.

[21] A. Pfeuffer and K. Dietmayer, "Robust semantic segmentation in adverse weather conditions by means of sensor data fusion," in *Proc. 22th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2019, pp. 1–8.

[22] A. Bar et al., "The vulnerability of semantic segmentation networks to adversarial attacks in autonomous driving: Enhancing extensive environment sensing," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 42–52, Jan. 2021.

[23] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2774–2783.

[24] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 1378–1387.

[25] X. Kang, B. Song, X. Du, and M. Guizani, "Adversarial attacks for image segmentation on multiple lightweight models," *IEEE Access*, vol. 8, pp. 31359–31370, 2020.

[26] C. Kamann and C. Rother, "Benchmarking the robustness of semantic segmentation models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 8828–8838.

[27] K. K. Nakka and M. Salzmann, "Indirect local attacks for context-aware semantic segmentation networks," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds. Glasgow, U.K.: Springer, 2020, pp. 611–628.

[28] G. Shen, C. Mao, J. Yang, and B. Ray, "AdvSPADE: Realistic unrestricted attacks for semantic segmentation," 2019, *arXiv:1910.02354*.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–27.

[30] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–14.

[31] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017, *arXiv:1712.09665*.

[32] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 1528–1540.

[33] K. Eykholt et al., "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1625–1634.

[34] Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein, "Making an invisibility cloak: Real world adversarial attacks on object detectors," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*. Glasgow, U.K.: Springer, Aug. 2020, pp. 1–17.

[35] T. Wu, X. Ning, W. Li, R. Huang, H. Yang, and Y. Wang, "Physical adversarial attack on vehicle detector in the Carla simulator," 2020, *arXiv:2007.16118*.

[36] M. Lee and Z. Kolter, "On physical adversarial patches for object detection," 2019, *arXiv:1906.11897*.

[37] Y. Zhang, H. Foroosh, P. David, and B. Gong, "CAMOU: Learning physical vehicle camouflages to adversarially attack detectors in the wild," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–20.

[38] A. Ranjan, J. Janai, A. Geiger, and M. J. Black, "Attacking optical flow," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct./Nov. 2019, pp. 2404–2413.

[39] J. Tu et al., "Physically realizable adversarial examples for LiDAR object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13713–13722.

[40] K. Yamanaka, R. Matsumoto, K. Takahashi, and T. Fujii, "Adversarial patch attacks on monocular depth estimation networks," 2020, *arXiv:2010.03072*.

[41] F. Nesti, A. Biondi, and G. Buttazzo, "Detecting adversarial examples by input transformations, defense perturbations, and voting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 3, pp. 1329–1341, Mar. 2023.

[42] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Adversarial defense by restricting the hidden space of deep neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3384–3393.

[43] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *IEEE Trans. Image Process.*, vol. 29, pp. 1711–1724, 2020.

[44] A. Saha, A. Subramanya, K. Patil, and H. Pirsiavash, "Role of spatial context in adversarial robustness for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3403–3412.

[45] P.-H. Chiang, C.-S. Chan, and S.-H. Wu, "Adversarial pixel masking: A defense against physical attacks for pre-trained object detectors," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 1856–1865.

[46] J. H. Metzen, N. Finnie, and R. Hutmacher, "Meta adversarial training against universal patches," in *Proc. ICML Workshop Adversarial Mach. Learn.*, 2021, pp. 1–23.

[47] S. Rao, D. Stutz, and B. Schiele, "Adversarial training against location-optimized adversarial patches," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, Jan. 2021, pp. 429–448.

[48] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "PatchGuard: A provably robust defense against adversarial patches via small receptive fields and masking," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, 2021, pp. 2237–2254.

[49] C. Xiang, S. Mahloujifar, and P. Mittal, "PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier," 2021, *arXiv:2108.09135*.

[50] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting localized universal attacks against deep learning systems," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2020, pp. 48–54.

[51] Z. Xu, F. Yu, and X. Chen, "LanCe: A comprehensive and lightweight CNN defense methodology against physical adversarial attacks on embedded multimedia applications," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 470–475.

[52] B. Li, J. Xu, S. Wu, S. Ding, J. Li, and F. Huang, "Detecting adversarial patch attacks through global-local consistency," in *Proc. 1st Int. Workshop Adversarial Learn. Multimedia*. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 35–41.

[53] K. T. Co, L. Muñoz-González, L. Kanthan, and E. C. Lupu, "Real-time detection of practical universal adversarial perturbations," 2021, *arXiv:2105.07334*.

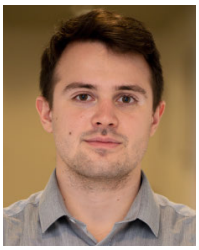
[54] G. Rossolini, A. Biondi, and G. Buttazzo, "Increasing the confidence of deep neural networks by coverage analysis," *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 802–815, Feb. 2023.

[55] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 86–94.

[56] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 1–12.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.

[58] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022.



Giulio Rossolini received the M.S. degree (cum laude) in computing engineering from the Scuola Superiore Sant'Anna of Pisa, University of Pisa, Pisa, Italy, in 2020. He is currently pursuing the Ph.D. degree with the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna, Pisa.

His current research interests include design and implementation of software tools to support and increase the trustworthiness of machine learning algorithms used in computer vision applications and safety-critical systems.

Federico Nesti received the M.S. degree (cum laude) in robotics and automation engineering from the University of Pisa, Pisa, Italy, in 2018, and the Ph.D. degree from the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna, Pisa, in 2022, where he investigated adversarial examples in the physical world and trustworthiness of AI-based computer vision systems.

Gianluca D'Amico received the M.S. degree in computer science and networking from the Scuola Superiore Sant'Anna, and University of Pisa, Pisa, Italy, in 2020. He is currently pursuing the Ph.D. degree with the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna.

His current research interests include computer graphics, especially the LiDAR technology, and design and implementation of visual simulation tools to test algorithms and neural network approaches in the railway environment and automotive.

Saasha Nair received the M.S. degree in informatics from the Technical University of Munich, Munich, Germany, in 2015.

She has been a Research Assistant with the Real-Time Systems (ReTiS) Laboratory, Scoula Superiore Sant'Anna, Pisa, Italy. She also explores topics in improving safety and trustworthiness of machine learning in cyber-physical systems, specifically autonomous vehicles.

Alessandro Biondi (Member, IEEE) received the M.S. degree (cum laude) in computer engineering from the University of Pisa, Pisa, Italy, in 2013, within the excellence program, and the Ph.D. degree in computer engineering from the Scuola Superiore Sant'Anna, Pisa, in 2017, under the supervision of Prof. Giorgio Buttazzo and Prof. Marco Di Natale.

In 2016, he has been a Visiting Scholar with the Max Planck Institute for Software Systems, Saarbrücken, Germany. He is currently an Associate Professor with the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna. His research interests include design and implementation of real-time operating systems and hypervisors, schedulability analysis, cyber-physical systems, synchronization protocols, and safe and secure machine learning.

Dr. Alessandro was a recipient of the six best paper awards and one outstanding paper award, the ACM SIGBED Early Career Award 2019, and the EDAA Dissertation Award 2017.

Giorgio Buttazzo (Fellow, IEEE) received the degree in electronic engineering from the University of Pisa, Pisa, Italy, in 1985, the M.S. degree in computer science from the University of Pennsylvania, Philadelphia, PA, USA, in 1987, and the Ph.D. degree in computer engineering from the Scuola Superiore Sant'Anna, Pisa, in 1991.

He is currently a Full Professor of computer engineering with the Sant'Anna School of Advanced Study of Pisa. He has authored ten books in real-time systems and more than 300 articles in the field of real-time systems, robotics, and neural networks.

Dr. Buttazzo received 13 best paper awards in 2013, the Outstanding Technical Achievement and Leadership Award by the IEEE Technical Committee on Real-Time Systems for the relevance of the scientific contributions in the real-time systems research field. He is the Editor-in-Chief of *Real-Time Systems* and an Associate Editor of the *ACM Transactions on Cyber-Physical Systems*.