Proceedings of the 37th AAAI Conference on Artificial Intelligence, Washington, DC, USA, February 7-14, 2023.

DEFENDING FROM PHYSICALLY-REALIZABLE ADVERSARIAL ATTACKS THROUGH INTERNAL OVER-ACTIVATION ANALYSIS

Giulio Rossolini, Federico Nesti, Fabio Brau, Alessandro Biondi and Giorgio Buttazzo
Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa, Italy

name.surname@santannapisa.it

ABSTRACT

This work presents *Z-Mask*, a robust and effective strategy to improve the adversarial robustness of convolutional networks against physically-realizable adversarial attacks. The presented defense relies on specific *Z-score* analysis performed on the internal network features to detect and mask the pixels corresponding to adversarial objects in the input image. To this end, spatially contiguous activations are examined in shallow and deep layers to suggest potential adversarial regions. Such proposals are then aggregated through a multi-thresholding mechanism. The effectiveness of *Z-Mask* is evaluated with an extensive set of experiments carried out on models for both semantic segmentation and object detection. The evaluation is performed with both digital patches added to the input images and printed patches positioned in the real world. The obtained results confirm that *Z-Mask* outperforms the state-of-the-art methods in terms of both detection accuracy and overall performance of the networks under attack. Additional experiments showed that *Z-Mask* is also robust against possible defense-aware attacks.

Keywords Adversarial Robustness, Real-world adversarial attacks, Semantic Segmentation, Object Detection, Adversarial Defense.

1 Introduction

Nowadays, deep neural networks yield impressive performance in computer vision tasks such as semantic segmentation and object detection. These remarkable results have encouraged the use of deep learning models also in critical *cyber-physical systems* (CPS), such as autonomous robots and cars. However, the trustworthiness of neural networks is often questioned by the existence of adversarial attacks [13], especially those performed in the physical world [1, 41, 35, 2, 16], which are most relevant to CPS. Such attacks are usually performed by means of adversarial objects, most often in the form of *patches* [3] which are capable of corrupting the model outcome when processed as a part of the input image.

Several techniques were proposed in the literature to defend neural networks from adversarial attacks. Many of them are based on adversarial training to regularize the behavior of the model (therefore, changing its parameters), which does not immunize a model against other adversarial examples and does not provide a means for detecting adversarial inputs. Other methods tackle this problem by masking or detecting physical adversarial attacks, without altering the model parameters. However, such approaches (discussed in Section 2) are often expensive or do not transfer well in realistic scenarios.

The defense method proposed in this paper is based on the experimental evidence that physical adversarial attacks yield anomalous activation patterns in the internal network layers [34, 47, 9, 35, 4]. To deepen the understanding of the effect of physical adversarial attacks, an extensive set of experiments have been carried out, showing that (1) shallow layers contain high/medium over-activations in the spatial image areas

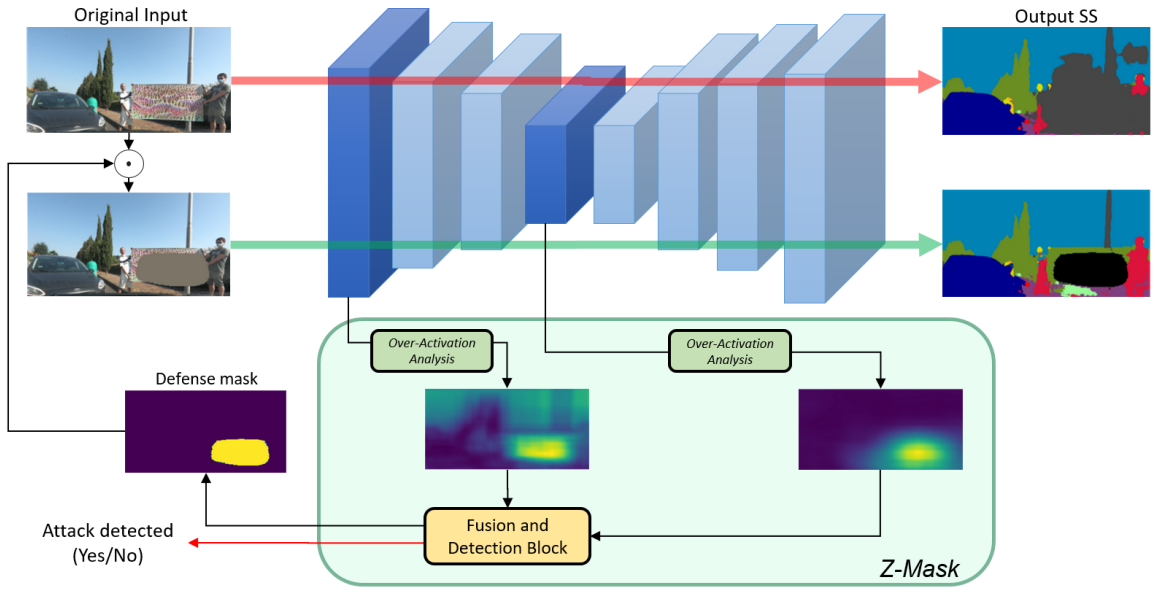


Figure 1: Illustration of the proposed approach.

corresponding to adversarial objects, and (2) in deeper layers such over-activations grow, both spatially and in magnitude, allowing to identify a region of interest that contains the adversarial patch. Based on such evidences, this paper proposes *Z-Mask*, a novel defense mechanism aimed at detecting and masking potential adversarial objects. Being the method task-agnostic, it can be applied to any convolutional model, without altering its original parameters. Figure 1 illustrates the proposed defense approach for the case of semantic segmentation.

To extract preliminary adversarial region proposals, *Z-Mask* runs an over-activation analysis (presented in Section 3.1) on a set of selected layers. This analysis exploits a *Spatial Pooling Refinement* to filter out high-frequency noise in over-activated regions. For each layer, 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 Section 3.2) that flags the presence of an adversarial object and generates the corresponding defense mask that serves the purpose of covering the detected adversarial patch. The mask generation algorithm leverages a trainable *multi-threshold mechanism* to fuse the information provided by $\mathcal{H}^{\mathcal{D}}$ and $\mathcal{H}^{\mathcal{S}}$.

A set of experimental results (reported in Section 4) highlights the effectiveness and the robustness of the proposed defense approach, which outperforms the state-of-the-art methods both in adversarial objects detection and masking. Furthermore, the experiments show that *Z-Mask* performs well against adversarial patches that are either digitally added to the input image or physically printed and placed in the real world.

In summary, this paper provides the following novel contributions:

- It proposes *Z-Mask*, a novel adversarial defense method designed to detect and mask the pixels belonging to adversarial objects;
- It shows the effectiveness of a *Z-score*-based defense by improving a naive neuron-wise approach with a *Spatial Pooling Refinement* that removes high-frequency noise and helps extract proper contiguous maps;
- It evaluates and compares the benefits of the proposed approach on semantic segmentation and object detection models against digital and real-world attacks.

The remainder of the paper is organized as follows: Section 2 introduces the related work, Section 3 presents the *Z-Mask* pipeline, Section 4 reports the experimental results, and Section 5 states the conclusions.

2 Related work

Physical adversarial attacks. Adversarial attacks are widely studied methods capable of easily fooling the outcomes of neural models by adding input perturbations [26, 23, 44, 39]. However, in recent years, particular interest has been devoted to adversarial attacks aimed at controlling the output of deep learning models through physical adversarial objects or patches that lay in the environment.

In this context, Athalye et al. [1] presented the Expectation Over Transformations (EOT) paradigm, which allows crafting adversarial objects robust against real-world transformations, as scaling, translation, orientation, and illumination changes. Later, Brown et al. [3] proposed an attack method based on adversarial patches, which achieved great success as a means to study the real-world robustness of deep neural networks and generate new effective physical attacks [28, 2, 41, 17, 12].

Defense methods. To tackle the problem of physical attacks and digital adversarial patches, several defense methods have been proposed in the literature. For the sake of clarity, defense methods are divided in two main categories: *adversarial training* and *external tools*. The former aim at making a model more robust by re-training the network using an augmented training set that includes attacked images. The latter aim at supporting the original model through an external tool (or methodology), without modifying the internal parameters of the model.

Adversarial training methods, such as the ones proposed in [36, 22, 31, 40], perform regularization operations that significantly increase the training and testing efforts without making the model immune to new adversarial examples. Furthermore, they do not provide any mechanism to notify the presence of an adversarial input or mask the pixels that realize the attack, e.g., those belonging to an adversarial patch.

Conversely, the methods based on external tools preserve the original model parameters and complement the model output with additional information that typically consists in an attack detection flag [9, 35, 43, 45] and/or defense masks [6, 27, 8, 20, 42] that remove the adversarial parts of the image.

Attack detection flags notify the presence of a possible adversarial object without localizing it in the image, and thus can only raise an alert to reject the predicted outcome. Defense masks specify a set of pixels that replaces those of the input image, allowing cleaning the attacked image areas. Some works [6, 20] proposed a defense strategy based on an additional deep learning model specialized on generating the mask, while other works [27, 49] directly generated the defense mask by computing the image gradient to filter out regions with high-frequency intensity variations.

Although all such methods do not alter the model parameters, only a few of them are task-agnostic and capable of working for both object detection and semantic segmentation models. Moreover, some of these works unfortunately lack of extensive evaluations on large datasets and realistic scenarios.

The role of internal activations. Among the large plethora of methods that study the internal behavior of a model under adversarial attacks, some works [34, 47, 9, 35, 4] showed that adversarial inputs cause large and abnormal activations in the internal network layers. In particular, two of them [9, 35] exploited this fact to detect adversarial patches by computing the cumulative sum of all the activated neurons in a certain layer. Such a score is deemed as *safe* or *unsafe* by comparing it to a threshold. Although this approach achieves good performance in detecting adversarial patches, it is applied to a single layer only using a neuron-wise over-activation analysis. Furthermore, it has only been used for detection purposes, without generating a defense mask, and it has been proved to be ineffective against defense-aware attacks [35].

This work. Inspired by previous approaches, this work provides a more comprehensive study of the over-activation caused by physically-realizable adversarial attacks. It consists in a defense method that performs a multi-layer and a multi-neuron analysis. First, a *spatial pooling refinement* is introduced in the over-activation analysis to better identify the image regions that determine the over-activations. Second, shallow and deep layer analysis are combined to generate an aggregated defense mask. *Z-mask* is a fully task-agnostic defense that outputs both a precise pixel mask and an attack detection flag, improving the adversarial robustness of convolutional models in the context of a large-scale evaluation that also targets realistic attacks (i.e., through physically-printed patches). Furthermore, the aggregation of information derived by multiple layers improve the robustness of the defended model, as shown in Section 4, since acting on multiple layers reduces the attack capabilities.

3 Proposed defense

This section presents the *Z-Mask* defense strategy, which is formulated to be task agnostic, i.e., applicable on any convolutional model. In this work, we consider the case of semantic segmentation and object detection models. In both cases, the input consists of an image with $H \times W$ pixels and C channels, denoted by $\mathbf{x} \in [0,1]^{C \times H \times W}$, while the form of the output $f(\mathbf{x})$ depends on the task. For a semantic segmentation model with N classes, the output $f(\mathbf{x}) \in [0,1]^{N \times H \times W}$ is an image that encodes the semantic context of each pixel. For an object detection model, the output $f(\mathbf{x})$ is a tensor encoding the class and the bounding box of each detected object. Without loss of generality, a task-specific loss function $\mathcal{L}(f(\mathbf{x}), \mathbf{y})$ is used to quantify the quality of a prediction $f(\mathbf{x})$ against the ground-truth output $\mathbf{y}$.

A real-world adversarial attack can be simulated by applying an *adversarial patch* in a specific region of the input image $\mathbf{x}$. A patch $\boldsymbol{\delta}$ is a $\tilde{H} \times \tilde{W}$ image within C channels, where $\tilde{H} \leq H$ and $\tilde{W} \leq W$. Crafting an adversarial patch requires solving an optimization problem that aims at minimizing a specific attack loss function while making patch features more robust against real-world transformations in the input image [1].

In detail, given an input image $\mathbf{x}$ and a patch $\boldsymbol{\delta}$, an additional function γ is randomly sampled from a set $\boldsymbol{\Gamma}$ of compositions of appearance-changing and placement transformations. The appearance-changing transformations include brightness, contrast change and noise addition; the patch placement transformations include random translation and scaling for defining the position of the patch in the image. Then, a patch $\boldsymbol{\delta}$ is applied to $\mathbf{x}$, according to γ, through a patch application function $g_\gamma(\mathbf{x}, \boldsymbol{\delta})$. Formally, an adversarial patch $\hat{\boldsymbol{\delta}}$ can be crafted by solving the following optimization problem:

$$\hat{\boldsymbol{\delta}} = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x} \sim \mathbf{X}, \gamma \sim \boldsymbol{\Gamma}} \, \mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x}, \boldsymbol{\delta}), \mathbf{y}_{Adv}), \tag{1}$$

where $\mathbf{X}$ is a set of known inputs, $\mathbf{y}_{Adv}$ is the adversarial target, and $\mathcal{L}_{Adv}$ is the adversarial loss that specifies the objective of the attacker. In the case of untargeted attacks, the adversarial target is the regular ground truth $\mathbf{y}$ and the adversarial loss function is $-\mathcal{L}(f(\tilde{\mathbf{x}}), \mathbf{y})$, to maximize the task-specific loss. To enhance the physical realizability of the patches, the adversarial loss includes additional terms that are described in the supplementary material for space limitations.

A defense masking strategy obscures a portion of the input image (supposedly containing the adversarial patch) through a pixel-wise product $\odot$ with a binary mask having the same size of the image. Formally, for each perturbed image $\tilde{\mathbf{x}} = g_\gamma(\mathbf{x}, \hat{\delta})$, a binary mask $M(\tilde{\mathbf{x}})$ is computed with the intent of satisfying the following property:

$$\mathcal{L}(f(\tilde{\mathbf{x}} \odot M(\tilde{\mathbf{x}})), \mathbf{y}) \approx \mathcal{L}(f(\mathbf{x}), \mathbf{y}). \tag{2}$$

Equation (2) states that the objective of a masking defense is to mitigate the effectiveness of a physical adversarial perturbation while preserving a correct behavior outside the region of the mask. Mask $M(\tilde{\mathbf{x}})$ is generated by leveraging the over-activation analysis described in Section 3.1 and aggregating multiple heatmaps through a *Fusion Block* mechanism, as described in Section 3.2.

3.1 Layer-wise over-activation analysis

Let $\boldsymbol{h}^{(l)} \in \mathbb{R}^{C^{(l)} \times H^{(l)} \times W^{(l)}}$ be the output features of layer l, obtained during the forward pass of $f(\mathbf{x})$, 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 2).

First, for a layer l, the channel-wise *Z-score* $\boldsymbol{z}^{(l)} = \frac{\boldsymbol{h}^{(l)} - \mu^{(l)}}{\sigma^{(l)}}$ of $\boldsymbol{h}^{(l)}$ is computed, where $\mu^{(l)}$ and $\sigma^{(l)}$ 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 $(A_1, \ldots, A_i, \ldots A_m)$ as follows:

$$\begin{cases} \boldsymbol{a}_i^{(l)} = \mathcal{R}(A_i(\mathcal{R}(\boldsymbol{z}^{(l)}))) \odot \frac{\boldsymbol{a}_{i-1}^{(l)}}{\|\boldsymbol{a}_{i-1}^{(l)}\|_\infty}, & i = 1, \ldots, m \\ \boldsymbol{a}_0^{(l)} \equiv 1, \end{cases} \tag{3}$$

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^{th} kernel is larger than the $(i+1)^{\text{th}}$ 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.

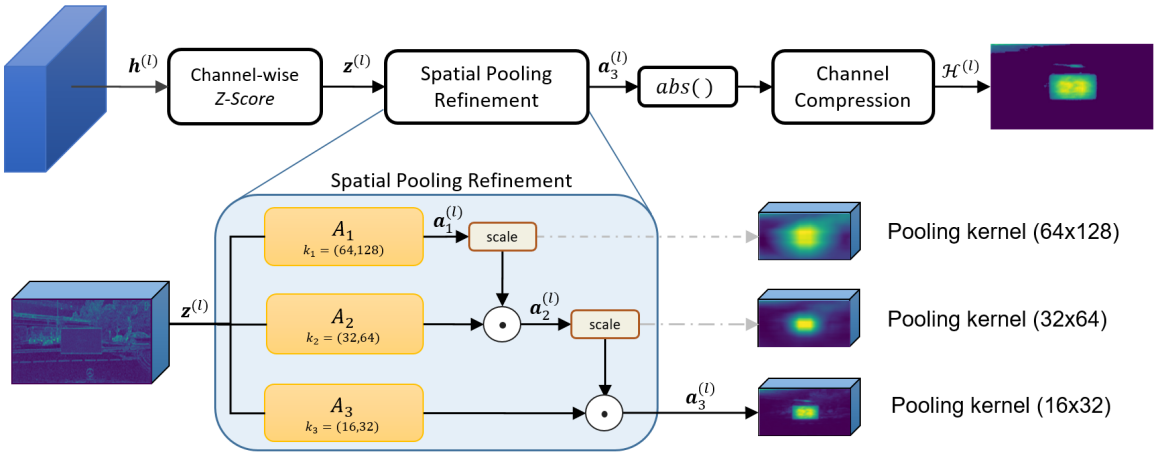


Figure 2: Over-activation pipeline performed by *Z-Mask* on a given layer with $m = 3$ Average-Pooling stages. The *scale* blocks used in the *Spatial Pooling Refinement* refer to the ∞-norm used in Equation 3. Resizing operations are omitted in the figure.

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 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 Spatial Pooling Refinement implements a cascade filtering [37] 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. This process yields a heatmap of the over-activated region with sharper borders.

3.2 Fusion and detection mechanism

This section explains how the mask $M(\mathbf{x})$ 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 3 illustrates the operations performed by the Fusion and Detection Block shown in Figure 1 to flag the presence of an adversarial object and generate the defense mask. The merging process leverages two *soft-thresholding blocks*, represented by dotted boxes. 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.

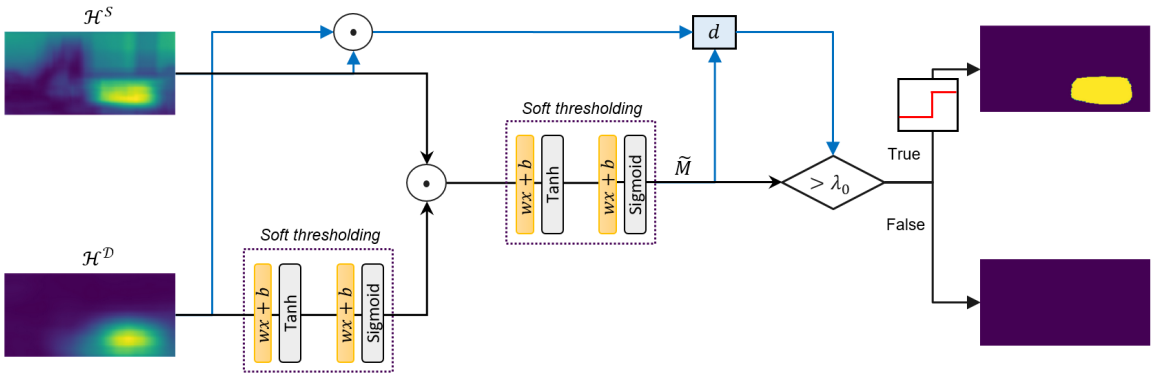


Figure 3: Fusion and Detection Block.

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

$$M(\mathbf{x}) = \begin{cases} 1 - \hbar(\tilde{M}), & d > \lambda_0 \\ 1, & \text{otherwise}, \end{cases} \tag{4}$$

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. More details about the datasets used and learning parameters are provided in Section 4.

4 Experimental evaluation

This section presents a set of experiments carried out on several convolutional models for object detection and semantic segmentation to evaluate the effectiveness of the proposed defense method.

4.1 Experimental settings

All the experiments were implemented using PyTorch [30] on a server with 8 NVIDIA-A100 GPUs. For both semantic segmentation and object detection tasks, the effectiveness of an adversarial attack was measured by evaluating the drop of the model performance with a task-dependent metric. For semantic segmentation models, the *mean Intersection-over-Union* (mIoU) [24] was used on the subset of pixels not belonging to the applied patch, as done in [35]. For object detection models, the performance was measured by the *COCO mean Average Precision* (mAP) [24, 19] on the whole image. Note that, for object detection, the patch area was not removed for the computation of the mAP. Therefore even a random patch reduces the overall accuracy, since it may occlude possible objects to be recognized.

Models. Three state-of-the-art models were selected for the semantic segmentation task, as done in [28]: ICNet [48], DDRNet [11], and BISeNet [46], using pretrained weights provided by their authors. For the object detection task, SSD [21], RetinaNet [18], and Faster R-CNN [32] were selected from the PyTorch model zoo[1]. More details are reported in the supplementary material.

Datasets. Several datasets were used for the experiments. The Cityscapes dataset [10] is a canonical dataset of driving images for semantic segmentation. It contains 2975 and 500 1024×2048 images for training and validation, respectively. For object detection, we considered the COCO 2017 dataset [19], containing 112k and 5k images for training and validation, respectively. Being COCO a dataset of common images, pictures have different sizes, hence a network-specific resizing is required. To assess the proposed approach on real-world scenarios, we considered three additional sets of images containing physical attacks. APRICOT [2] 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. We also used a private dataset of images with physical adversarial patches for object detection and another one containing road images with a large adversarial poster for semantic segmentation [35].

[1] https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

Attack strategies. Different attack methodologies were used to craft adversarial patches. For semantic segmentation models, we leveraged the untargeted attack pipeline used in [28], while, for object detection models, we performed an untargeted attack on the classes, similarly to [5]. The patches contained in the APRICOT dataset were instead crafted following a false-detection attack as shown in [2], where the network is forced to spawn a spurious detection on the area of the patch. More details about the applied loss function and attack optimization strategies are provided in the supplementary material.

***Z-Mask* settings and training.** For semantic segmentation models, the heatmaps in $\mathcal{S}$ were generated with a Spatial Pooling Refinement 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 object detection models, the Spatial Pooling Refinement 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 extensive preliminary tests performed to analyse the internal activations. The description of the layers selected for extracting $\mathcal{D}$ and $\mathcal{S}$ is reported in the supplementary material. Providing an automatic procedure for selecting the most useful layers is left as a future work.

The parameters of the *soft-thresholding* operations inside the Fusion and Detection block (to compute heatmap $\tilde{M}$) were trained in a supervised fashion by considering a set of patches (whose ground-truth binary mask $\bar{M}$ is known) and minimizing the pixel-wise binary cross-entropy loss function $\mathcal{L}_{\text{BCE}}(\tilde{M}, \bar{M})$. Not all the adversarial patches lead to the same magnitude of over-activation: for instance, as stated in [35], images including printed patches induce slightly lower over-activation values. For this reason, training was done by considering an augmented set of adversarial patches causing different levels of over-activations in the shallow layers. In this way, the Fusion and Detection Block resulted to be robust to a wider spectrum of over-activations.

To generate adversarial patches while controlling the magnitude of over-activation, inspired by the approach used in [35], we used the following problem formulation:

$$\hat{\boldsymbol{\delta}}_\beta = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \; \{\mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\boldsymbol{\Gamma}} \left[(1-\beta)\cdot \; \mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta})) + \beta \cdot \; \mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x}, \boldsymbol{\delta})), \mathbf{y})\right]\}, \tag{5}$$

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 shallow layers in the patch area (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 network to generate less over-activation in the shallow layers, hence simulating both real-world patches and patches that are maliciously crafted to result in low over-activation scores to bypass the detection mechanism. Leveraging this formulation, 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. This set of patches was 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 [15] was used for this purpose, with a learning-rate of 0.01 and training for 15 epochs. The channel-wise standard deviation 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 for the detection ROC curve. The ROC was generated by computing the over-activation 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.

Other works. 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) [27] and MaskNet [6], both with the original settings described by the authors. While training MaskNet for SSD and RetinaNet on COCO, we encountered learning limits of the masking model (as reported in Table 4), compatible with the practical issues described by the authors. For the adversarial detection, the proposed method was compared with state-of-the-art strategies tested on semantic segmentation and object detection models, which are FPDA [35] and HN [9]. Details are provided in the supplementary material.

Net	Patch	Defense Method (mAP Val)			
		Z-Mask	MaskNet	LGS	None
FRCNN	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)

Figure 4: Robustness performance evaluated for different patch sizes for object detection (a) and semantic segmentation models (b).

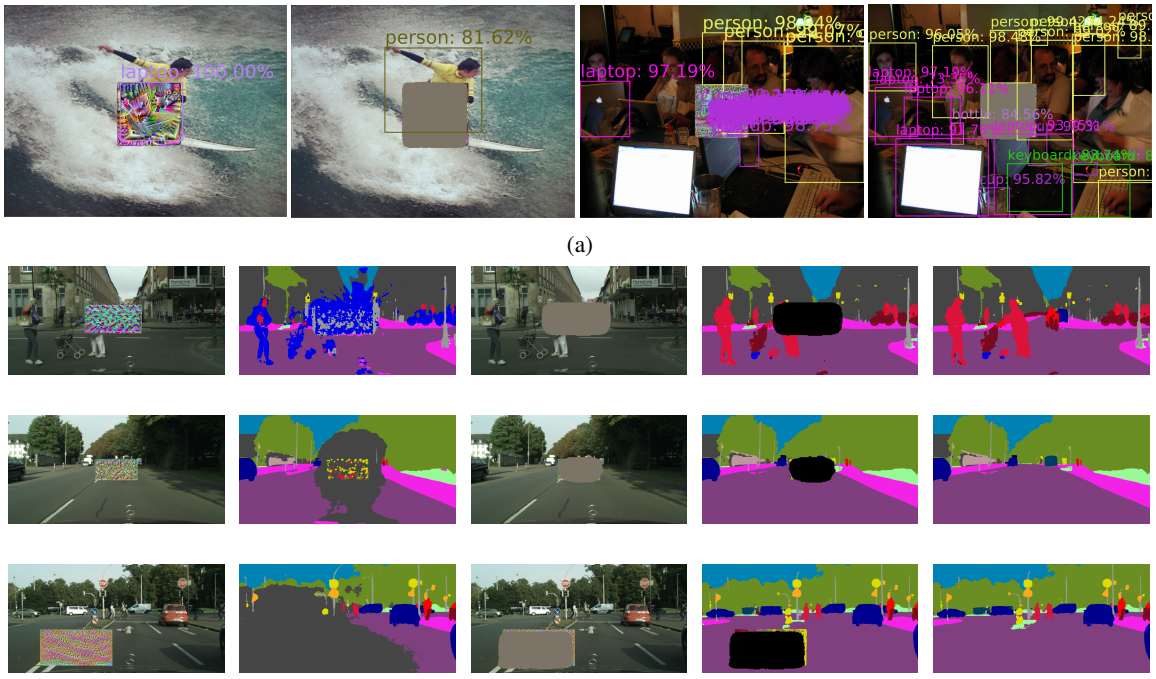
4.2 Evaluation for digital attacks

Masking performance. The benefits of the proposed defense mechanism were evaluated by attacking the validation sets with different adversarial patch sizes. For semantic segmentation models on 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 for both datasets 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 4, *Z-Mask* outperformed other strategies in the defense against patch attacks, achieving scores similar to the random case when tested against adversarial attacks. Furthermore, *Z-Mask* kept results close to the nominal ones of the models (i.e., without adversarial patches), meaning that it did not affect the original model performance. Figure 5 provides some examples of Cityscapes and COCO images attacked digitally with adversarial patches. *Z-Mask* identified and covered attacked regions of the input without affecting other portions of the image.

Detection performance. All the adversarial patches evaluated in Table 4 were perfectly detected by both *Z-Mask*, HN, and FPDA (i.e., each method obtained 1.00 AUC of ROC). To better assess the performance of these adversarial detection methods, we used the optimization described in Equation (5) 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 adversarial effect to reduce the magnitude of over-activation. An L-sized patch was generated for each β. Figure 6 shows the detection and masking accuracy against this set of patches as a function of β for ICNet and 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 (as negative and positive samples, respectively). Note that *Z-Mask* achieved better results than the other adversarial detectors against effective patch attacks, providing good detection performance also to patches that do not retain much adversarial effect. The bottom part of the image reports the corresponding masking performance of *Z-Mask*, MaskNet, LGS, and the original model. Again, our method achieved higher mIoU among all the values of β. The same analysis was performed for the object detection models with similar results (see the supplementary material).

4.3 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 (trained with COCO and Cityscapes), 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 7(a) reports the corresponding ROC curves, 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



(a)

(b)

Figure 5: (a) Masking performance for object detection on the COCO Dataset, using RetinaNet (left) and Faster R-CNN (right).(b) Masking performance for semantic segmentation on Cityscapes, using BiseNet (fist row), DDRNet (second row) and ICNet (third row). Each row shows the patched image, its predicted output, the cleaned image and its corresponding output, finally also the no patched predicted image is shown in the last column for a visual comparison.

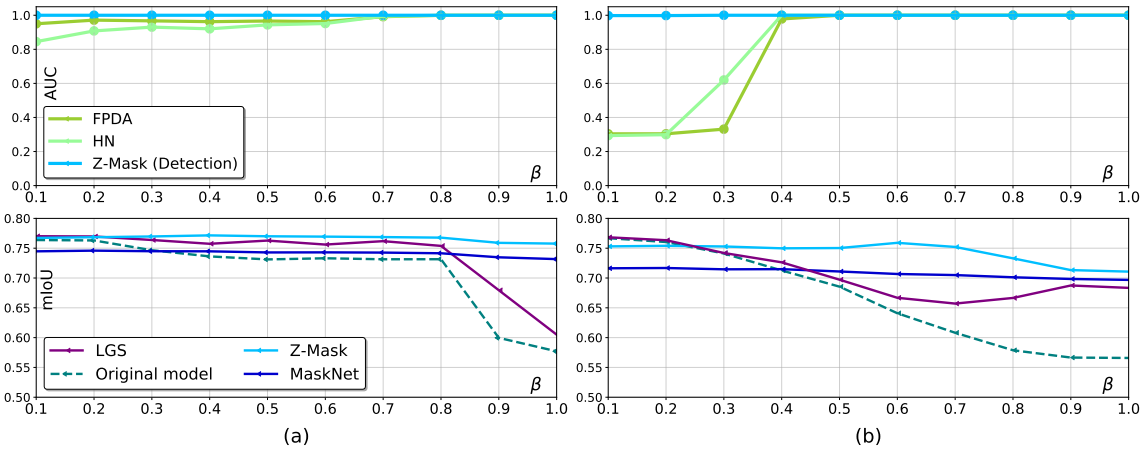


Figure 6: Detection accuracy and comparison using ICNet (a) and DDRNet (b) on the Cityscapes dataset. First row shows the AUC of ROC computed for each β value, while second row plots the corresponding mIoU computed on the attacked Cityscapes validation set.

network restrained APRICOT patches to just a few pixels in the input image, thus neutralizing their adversarial effect. Figure 7(b) illustrates the effect of *Z-Mask* on some inputs of APRICOT. Additional examples are provided in the supplementary material. To further validate the defense performance of *Z-Mask*, Figure 8 reports some illustrations of the masking effect. Notably, the effect of the patches was largely reduced, for both real-world semantic segmentation images (provided by [35]) and object detection pictures (from a personal dataset).

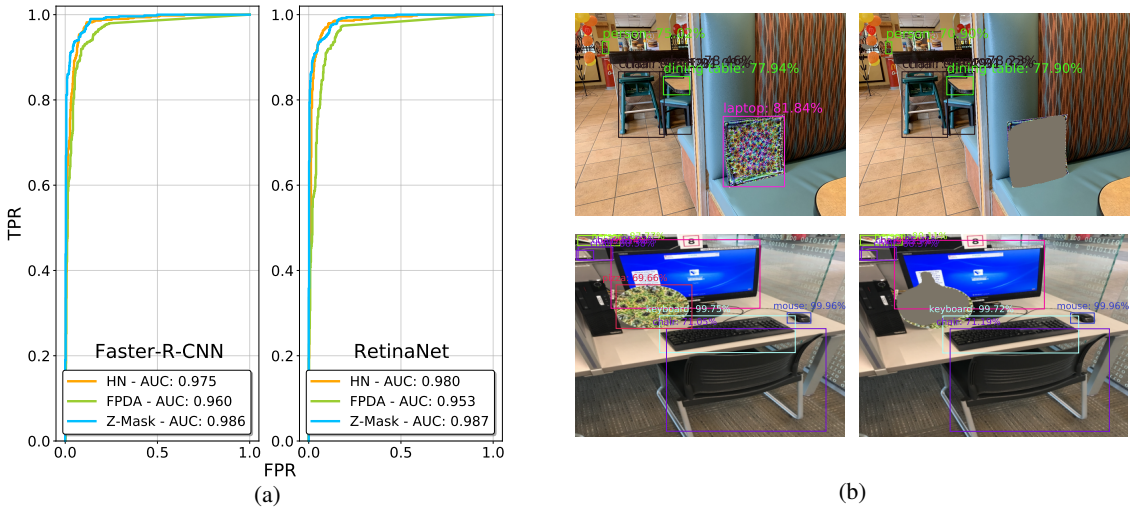


Figure 7: (a) ROC analysis performed on the dataset including APRICOT images and 1000 COCO images with the Faster R-CNN and RetinaNet. (b) APRICOT sample images with physical patches covered by *Z-Mask*.

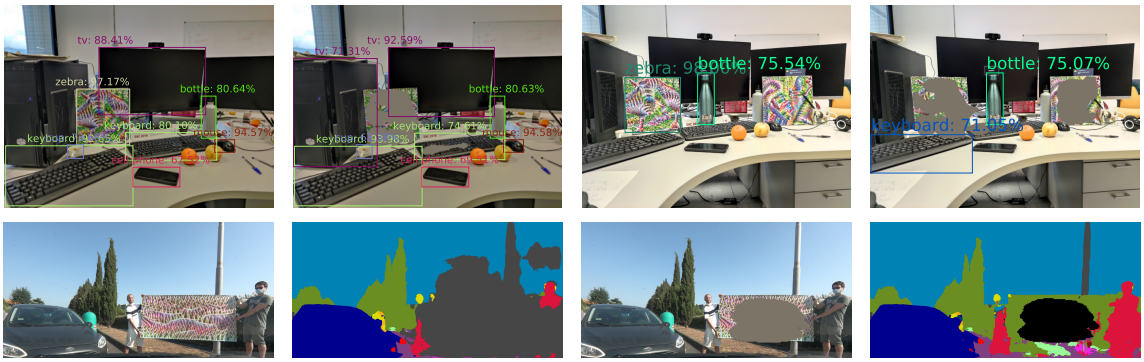


Figure 8: Examples of the masking effects on physical attacks for object detection (on RetinaNet) and semantic segmentation (on ICNet). A defense mask not only protects the model by removing wrong local adversarial predictions associated with the patch, but also mitigates its global effects in corrupting other objects in the image.

4.4 Defense-aware attacks

Since the pipeline defined by *Z-Mask* is fully differentiable up to $\tilde{M}$ (the last operation is a thresholding), an attacker might exploit that knowledge to craft defense-aware attacks, i.e., optimize patches that are adversarial for the model and the defense together. To this end, we propose two different defense-aware attacks.

The first attack, denoted as *Adv-Mask*, 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 $\alpha \in \{0, 0.1, 0.2, \ldots, 1.0\}$:

$$\hat{\boldsymbol{\delta}}_\alpha = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \left\{ \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\Gamma} \left[(1-\alpha) \cdot (-\mathcal{L}_{BCE}(\tilde{M}, \bar{M})) + \alpha \cdot \mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x}, \boldsymbol{\delta})), \mathbf{y}) \right] \right\}. \tag{6}$$

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 *Adv-Flag*, 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 $\mathcal{L}_{\text{BCE}}(\textit{Sigmoid}(d - \lambda_0), 1)$. This is done to force $d < \lambda_0$ in the optimization process, hence resulting in a mask $M(\mathbf{x}) = 1$.

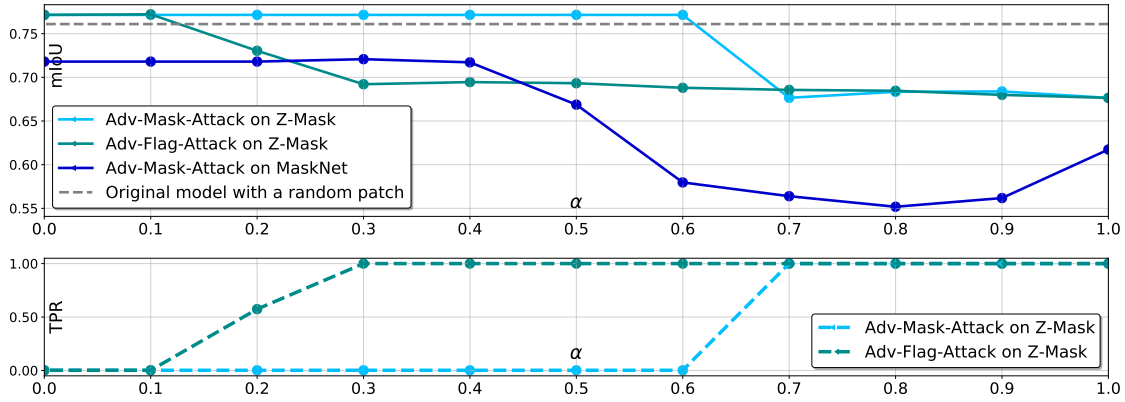


Figure 9: Evaluation of masking (mIoU) and detection (TPR) performance against defense-aware attacks as a function of α. The Adv-Mask attack is also evaluated for MaskNet. The results refer to DDRNet evaluated on the validation set of Cityscapes.

Figure 9 shows the results of *Z-Mask* against the two attacks introduced above for DDRNet (results on other networks in the supplementary material). The *Adv-Mask* was also tested on *MaskNet* to provide a comparison of the robustness of these methods. The results of LGS are not reported, since methods that relies on input transformations shows to be extremely prone to defense-aware attacks [6, 29]. 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 4. Indeed, observe from Figure 9 that, when Z-Mask does not detect the attack (TPR=0), the attack is not effective (maximum mIoU). Conversely, for MaskNet, certain values of α induce larger performance degradation.

5 Conclusions and final remarks

This paper presented *Z-Mask*, a method able to detect physically-realizable adversarial examples and generate a defense mask against them. The method is based on identifying over-activations in the hidden layers of the defended model, which are typically induced by physical adversarial attacks. This is accomplished by leveraging specific processing modules, such as the Spatial Pooling Refinement and the Fusion and Detection Block. Differently from most of the defense strategies in the literature, the proposed defense consists of a light-weight model that is not expensive to train, is robust to defense-aware attacks, and is able to transfer well on large dataset and in real-world scenarios. Also, it does not require retraining the model parameters. Furthermore, *Z-Mask* is task-agnostic and was tested with object detection and semantic segmentation models, obtaining state-of-the-art results for both adversarial pixel masking and detection.

As a future work, we plan to address (i) a more formal analysis of the behaviour of the network aimed at understanding the evident correlation between over-activations and adversarial effects, and (ii) the design of a simpler learning strategy for the definition of the thresholds involved in the computation of the binary mask and the selection of the shallow and deep layers to be selected for the over-activation analysis.

References

[1] Athalye, A., Engstrom, L., Ilyas, A., Kwok, K.: Synthesizing robust adversarial examples. In: Proceedings of the 35th International Conference on Machine Learning. pp. 284–293. Proceedings of Machine Learning Research (2018)

[2] Braunegg, A., Chakraborty, A., Krumdick, M., Lape, N., Leary, S., Manville, K., Merkhofer, E., Strickhart, L., Walmer, M.: Apricot: A dataset of physical adversarial attacks on object detection. In: European Conference on Computer Vision. pp. 35–50. Springer (2020)

[3] Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial Patch. arXiv:1712.09665 [cs] (May 2018)

[4] Carrara, F., Becarelli, R., Caldelli, R., Falchi, F., Amato, G.: Adversarial examples detection in features distance spaces. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. Springer (September 2018)

[5] Chen, S.T., Cornelius, C., Martin, J., Chau, D.H.P.: Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In: Machine Learning and Knowledge Discovery in Databases. pp. 52–68. Springer (2019)

[6] Chiang, P.H., Chan, C.S., Wu, S.H.: Adversarial pixel masking: A defense against physical attacks for pre-trained object detectors. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 1856–1865. MM '21, Association for Computing Machinery (2021)

[7] Chollet, F.: Xception: Deep Learning with Depthwise Separable Convolutions. arXiv e-prints arXiv:1610.02357 (Oct 2016)

[8] Chou, E., Tramer, F., Pellegrino, G.: Sentinet: Detecting localized universal attacks against deep learning systems. In: 2020 IEEE Security and Privacy Workshops (SPW). pp. 48–54. IEEE (2020)

[9] Co, K.T., Muñoz-González, L., Kanthan, L., Lupu, E.C.: Real-time detection of practical universal adversarial perturbations. arXiv:2105.07334 (2021)

[10] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Conference on Computer Vision and Pattern Recognition CVPR. pp. 3213–3223. IEEE Computer Society (2016)

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

[12] Hu, Y.C.T., Kung, B.H., Tan, D.S., Chen, J.C., Hua, K.L., Cheng, W.H.: Naturalistic physical adversarial patch for object detectors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). IEEE (2021)

[13] Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., Yi, X.: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review **37**, 100270 (2020)

[14] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22Nd ACM International Conference on Multimedia. pp. 675–678. MM '14, ACM, New York, NY, USA (2014)

[15] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA (2015)

[16] Kong, Z., Guo, J., Li, A., Liu, C.: 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. pp. 14242–14251. IEEE (2020)

[17] Lee, M., Kolter, Z.: On Physical Adversarial Patches for Object Detection. arXiv:1906.11897 (Jun 2019)

[18] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV). pp. 2980–2988 (2017)

[19] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)

[20] Liu, J., Levine, A., Lau, C.P., Chellappa, R., Feizi, S.: Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. arXiv:2112.04532 (2021)

[21] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision ECCV. pp. 21–37. Springer (2016)

[22] Metzen, J.H., Finnie, N., Hutmacher, R.: Meta adversarial training against universal patches. In: ICML 2021 Workshop on Adversarial Machine Learning (2021)

[23] Metzen, J.H., Kumar, M.C., Brox, T., Fischer, V.: Universal adversarial perturbations against semantic image segmentation. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy. pp. 2774–2783. IEEE Computer Society (2017)

[24] Minaee, S., Boykov, Y.Y., Porikli, F., Plaza, A.J., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021)

[25] Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 86–94. IEEE Computer Society (2017)

[26] Nakka, K.K., Salzmann, M.: Indirect local attacks for context-aware semantic segmentation networks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) 16th European Conference Computer Vision ECCV, Glasgow, UK. vol. 12350, pp. 611–628. Springer (2020)

[27] Naseer, M., Khan, S., Porikli, F.: Local gradients smoothing: Defense against localized adversarial attacks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (2019)

[28] Nesti, F., Rossolini, G., Nair, S., Biondi, A., Buttazzo, G.: Evaluating the robustness of semantic segmentation for autonomous driving against real-world adversarial patch attacks. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 2826–2835. IEEE Computer Society (2022)

[29] Nesti, F., Biondi, A., Buttazzo, G.: Detecting adversarial examples by input transformations, defense perturbations, and voting. IEEE Transactions on Neural Networks and Learning Systems pp. 1–13 (2021)

[30] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)

[31] Rao, S., Stutz, D., Schiele, B.: Adversarial training against location-optimized adversarial patches. In: European Conference on Computer Vision ECCV. pp. 429–448. Springer (2020)

[32] Ren, S., He, K., Girshick, R., Sun, J.: 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)

[33] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)

[34] Rossolini, G., Biondi, A., Carlo Buttazzo, G.: Increasing the Confidence of Deep Neural Networks by Coverage Analysis. arXiv:2101.12100 (2021)

[35] Rossolini, G., Nesti, F., D'Amico, G., Nair, S., Biondi, A., Buttazzo, G.: On the Real-World Adversarial Robustness of Real-Time Semantic Segmentation Models for Autonomous Driving. arXiv:2201.01850 (2022)

[36] Saha, A., Subramanya, A., Patil, K., Pirsiavash, H.: Role of spatial context in adversarial robustness for object detection. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 3403–3412. IEEE (2020)

[37] Satti, P., Sharma, N., Garg, B.: Min-max average pooling based filter for impulse noise removal. IEEE Signal Processing Letters **27**, 1475–1479 (2020)

[38] Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: 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. pp. 1528–1540. ACM, Vienna Austria (Oct 2016)

[39] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada (2014)

[40] Wu, T., Tong, L., Vorobeychik, Y.: Defending against physically realizable attacks on image classification. In: 8th International Conference on Learning Representations ICLR (2020)

[41] Wu, Z., Lim, S., Davis, L.S., Goldstein, T.: Making an invisibility cloak: Real world adversarial attacks on object detectors. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) 16th European Conference on Computer Vision ECCV, Glasgow, UK. vol. 12349, pp. 1–17. Springer (2020)

[42] Xiang, C., Bhagoji, A.N., Sehwag, V., Mittal, P.: {PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 2237–2254 (2021)

[43] Xiang, C., Mittal, P.: Detectorguard: Provably securing object detectors against localized patch hiding attacks. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. p. 3177–3196. Association for Computing Machinery, New York, NY, USA (2021)

[44] Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.L.: Adversarial examples for semantic segmentation and object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy. pp. 1378–1387. IEEE Computer Society (2017)

[45] Xu, Z., Yu, F., Chen, X.: 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). pp. 470–475. IEEE (2020)

[46] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 325–341. Springer (2018)

[47] Yu, C., Chen, J., Xue, Y., Liu, Y., Wan, W., Bao, J., Ma, H.: Defending against universal adversarial patches by clipping feature norms. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16434–16442 (2021)

[48] Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: Icnet for real-time semantic segmentation on high-resolution images. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 405–420. Springer (2018)

[49] Zhou, G., Gao, H., Chen, P., Liu, J., Dai, J., Han, J., Li, R.: Information distribution based defense against physical attacks on object detection. In: 2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW). pp. 1–6 (2020)

Supplementary Material for "Defending From Physically-Realizable Adversarial Attacks Through Internal Over-Activation Analysis"

A CNN models

This section reports additional details of the models and settings used in our experiments. To obtain fair experimental results we used pretrained models made available by the corresponding authors. Each network structure was imported into our repository keeping the original image normalization parameters required by each pretrained model.

A.1 Semantic segmentation

DDRNet We used the DDRNet23Slim version provided by the author [11][2] that has shown to be one of fastest networks in the state-of-the-art for real-time semantic segmentation.

BiSeNet We used the original Pytorch implementation [46][3] with its pretrained weights. The applied version uses an Xception39 model [7] as backbone.

ICNet The original pretrained model[4] provided by the authors [48] (trained using the Caffe framework [14]) was imported in PyTorch.

A.2 Object detection

The object detection models used in the paper were downloaded from the PyTorch model zoo[5]: Faster R-CNN with ResNet-50 FPN backbone, RetinaNet with ResNet-50 FPN backbone, and SSD300 with VGG16 backbone. The pre-trained versions on COCO were used, with input size of 800×1333 for Faster R-CNN and RetinaNet, and 300×300 for SSD.

A.3 Selected layers for the over-activation analysis

As illustrated in the main manuscript, we selected specific hidden layers from each model to generate the two set of heatmaps $\mathcal{S}$ and $\mathcal{D}$. Table 1 reports the selected layers denoted through the layer name available in the pretrained models mentioned above.

The decision of using such layers came after an extensive set of preliminary experiments performed to evaluate the layer-wise over-activation behavior against adversarial patches. In fact, the obtained results showed that, in the presence of an adversarial input, there are layers that are more prone to be corrupted by over-activations.

The development of an automatic procedure for selecting the most appropriate layers for detection and masking is left as a future work.

B Details on the adversarial loss functions

As stated in the main paper, an adversarial patch is crafted with the following optimization

$$\hat{\boldsymbol{\delta}} = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\zeta\sim\boldsymbol{\Gamma}} \; \mathcal{L}_{Adv}(f(g_{\zeta}(\mathbf{x},\boldsymbol{\delta}),\mathbf{y}_{Adv}). \tag{7}$$

This formula includes a few simplifications regarding the adversarial loss $\mathcal{L}_{Adv}$. First, $\mathcal{L}_{adv}$ is actually a linear combination of different terms, one for achieving the adversarial effect, and two additional terms for physical realizability (that are described in Section B.2). The term responsible for the adversarial effect is referred to as $\mathcal{L}_{adv-eff}$, and has different formulations for semantic segmentation and object detection.

[2] https://github.com/ydhongHIT/DDRNet

[3] https://github.com/ycszen/TorchSeg

[4] https://github.com/hszhao/ICNet

[5] https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

Network	Layers for $\mathcal{S}$	Layers for $\mathcal{D}$
DDRNet	rbb2, rbb1, rb1, rb2	rbb2, rbb1
ICNet	res-block5, convbnrelu1-2, res-block2	res-block5, conv5-4-k1, convbnrelu1-2
BiseNet	spatial-path-conv-1x1, spatial-path-conv-7x7, ct2	ct1,ct2,spatial-path-conv-1x1
RetinaNet	backbone-body-relu, backbone-body-conv1, backbone-body-layer1-0-conv1	backbone-body-conv1, backbone-layer4, backbone-body-layer1-0-conv1
Faster R-CNN	backbone-body-relu, backbone-body-conv1, backbone-body-layer1-0-conv1	backbone-body-conv1, backbone-layer4, backbone-body-layer1-0-conv1
SSD	backbone-extra-0-3, backbone-extra-0-1	backbone-extr-0-3, backbone-extra-0-5

Table 1: List of the selected layer for $\mathcal{S}$ and $\mathcal{D}$. The notation used refers the Pytorch dictionary format available for all the pretrained models used.

Semantic Segmentation Since the output of a semantic segmentation model is a tensor encoding the predicted class for each pixel of the image, $\mathcal{L}_{adv-eff}$ must be a classification loss. In particular, we used the loss formulation proposed in [28], which is a modification of the standard cross-entropy that improves the adversarial effect for attacks against semantic segmentation, indicated with $\mathcal{L}_{SS}$ and weighted with w_{SS}.

Object Detection Object detection architectures are mainly categorized in two classes: single-stage and two-stage detectors. Single-stage detectors, as RetinaNet or SSD, output directly the detected objects, whereas two-stage detectors, as Faster R-CNN, require two separate modules: a Region Proposal Network (RPN) that outputs detection proposals in the feature space, and a detection head that refines these predictions to output actual bounding boxes for the input image.

For both types of architectures, the final output is a set of detections, each of which defined by a bounding box and by the classification logits. The bounding box is expressed as a tuple (x, y, h, w), where (x, y) denotes its left-upper corner position in the image and (h, w) represent its height and width, in pixels, respectively. Then, the total loss is composed by a linear combination of a classification loss (cross-entropy loss $\mathcal{L}_{CE}$), and a regression loss (L1 loss on the bounding box output error) of the matched detections with respect to the ground truth. The matching process is done by applying a heuristic threshold on the IoU between the detection and ground truth bounding boxes. Also, bounding boxes are filtered considering the output confidence and the size of the detection.

If the detector is two-stage, an additional loss tuple is used, with the same classification-regression separation, for the RPN output.

Hence, the total loss used for the adversarial effect under object detection is

$$\mathcal{L}_{adv-eff,S}(f(\mathbf{x}), \mathbf{y}_{Adv}) = w_{class}\mathcal{L}_{CE}(f_{class}(\mathbf{x}), \mathbf{y}) + w_{regr}\mathcal{L}_1(f_{bbox}(\mathbf{x}), \mathbf{y}_{Adv}) \tag{8}$$

for the single-stage detectors[6] and

$$\begin{aligned}\mathcal{L}_{adv-eff,T}(f(\mathbf{x}), \mathbf{y}_{Adv}) = \mathcal{L}_{adv-eff,S}(f(\mathbf{x}), \mathbf{y}_{Adv}) \quad + \\ w_{RPN-class}\mathcal{L}_{CE}(f_{RPN-class}(\mathbf{x}), \mathbf{y}_{Adv}) \quad + \\ w_{RPN-regr}\mathcal{L}_1(f_{RPN-bbox}(\mathbf{x}), \mathbf{y}_{Adv})\end{aligned} \tag{9}$$

for two-stage detectors.

Please note that $w_i \in \mathbb{R}$ and that the optimization objective changes drastically for $w_i < 0$ and $w_i > 0$, depending also on the adversarial target $\mathbf{y}_{Adv}$. This is discussed in the following section.

B.1 Optimization objective

Different values of w_i and $\mathbf{y}_{Adv}$ lead to different optimization objectives. For the semantic segmentation models, we used an untargeted attack because it has been shown [35] that this type of attacks are much more effective than targeted attacks in real-world scenarios. Hence, we set $w_{SS} < 0$ to force the adversarial effect to produce a maximization of $\mathcal{L}_{SS}$ with respect to $\mathbf{y}_{Adv} = \mathbf{y}$.

[6] f_{class} and f_{bbox} are the matched classification and bounding box output. Please note that, in this formulation, both the regression and classification losses include the heuristic matching and filtering procedures not discussed here.

For object detection models, different objectives can be defined, considering all the possible combinations of the sign of w_{class} and w_{regr}. A fully untargeted attack can be achieved by setting $w_{class} < 0$ and $w_{regr} < 0$. However, a more interesting setting for a real-world attack would be to only change the classes, keeping the same bounding boxes, hence considering $w_{class} < 0, w_{regr} > 0$. We adopted this setting throughout all the experiments, because, while a fully-untargeted attack is more effective overall, we found that a class-untargeted attack is more effective in lowering the detection confidence in the scene, often leading to missed detections. This is more challenging for safety-critical systems.

False detection attacks, as the ones included in the APRICOT dataset [2], can be created by defining $\mathbf{y}_{Adv}$ as a single target detection, defined by a certain class and a certain bounding box (APRICOT patches define this target bounding box as coincident with the area of the patch itself). Then, it is necessary to craft a targeted attack, by setting $w_{class} > 0, w_{regr} > 0$. This is a really easy attack to craft and transfer to the real world.

All these considerations hold for both one- and two-stage detectors. The weights on the additional RPN losses for two-stage detectors can easily be set with the same sign of the corresponding output losses.

B.2 Loss functions for physical realizability

The adversarial effect loss function is sufficient to optimize digital adversarial examples, which rely on the assumption that the attacker has full control on the digital representation of the image. Physically-realizable adversarial examples require a more specialized adversarial loss, based on universal perturbations [25] (to produce image-agnostic perturbations) and the EOT paradigm [1] (to render the perturbation robust to transformations typical of the real world), as explained in the main paper.

However, physical realizability is another important factor that has to be accounted for. In particular, it is likely that the printer used to print the adversarial perturbation is not able to print the entire continuous spectrum of colors. Hence, the non-printability score [38] is introduced to take this effect into account. Assuming that a pixel p of patch δ is composed of an RGB triplet, the non-printability score $\mathcal{L}_N$ is defined as

$$\mathcal{L}_N(p) = \prod_{c \in C} \|p - c\| \tag{10}$$

where C is the set of RGB triplets that compose the printable color palette of the printer. The non-printability score is then averaged over the totality of pixels in the perturbation, and should be low when the totality of the patch pixel colors is close to the printable ones.

Also, typical digital perturbations are very noisy, due to the little correlation between adjacent pixels. In the real world, these patterns 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 another additional loss, the "smoothness" loss, defined as

$$\mathcal{L}_S = \sum_i \sum_j [(\delta_{i+1,j} - \delta_{i,j})^2 + (\delta_{i,j+1} - \delta_{i,j})^2] \tag{11}$$

where (i, j) are the 2D coordinates of the patch. This additional loss function introduces a correlation among adjacent pixels and should help crafting a "smoother" perturbation, without noisy patterns.

Summing these additional loss terms, the total loss function for physically-realizable real-world adversarial examples becomes

$$\mathcal{L}_{Adv} = w_{adv-eff}\mathcal{L}_{adv-eff} + w_N\mathcal{L}_N + w_S\mathcal{L}_S \tag{12}$$

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

In practice, the adversarial loss (for the untargeted attack formulation) tends to increase in norm during the optimization. This increase 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, normalized, and then averaged according to the weights. This total gradient is applied to advance the optimization.

The experiments in Section IV of the main paper are performed with $w_{adv} = 1, w_N = 0, w_S = 0.1$. 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.

B.3 Other attack optimization settings

As stated in Equation (7), all the adversarial patches are obtained by performing universal attacks (i.e., the patch is optimized to be adversarial among several inputs). This process helps the adversarial features to

generalize their effect on different images, and hence, to transfer to the physical world. In particular, to craft the adversarial patches tested in the digital evaluation of the main paper, we used a dataset containing 100 images randomly extracted from the Cityscapes and COCO datasets for semantic segmentation and object detection models, respectively. The optimizations were performed for 150 epochs.

To limit the training time, we reduced the number of epochs to 50 and the dataset size to 100 to craft all the other patches, i.e., patches with different values of β and α (for the defense-aware experiments).

C Over-activation loss

To train the *soft-thresholding* parameters in the Fusion and Detection Block we defined a set of patches $\Delta = \{\hat{\boldsymbol{\delta}}_\beta \,:\, \beta \in [\beta_0, 1]\}$ obtained by solving the following optimization problem:

$$\hat{\boldsymbol{\delta}}_\beta = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \; \{\mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\boldsymbol{\Gamma}} \left[(1-\beta)\cdot \; \mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta})) + \beta \cdot \; \mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x}, \boldsymbol{\delta})), \mathbf{y})\right]\}, \tag{13}$$

where $\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta}))$ helps control the over-activation values yielded by adversarial patches, according to the β value. In particular, a low value of β reduces the importance assigned to the adversarial effect, while forcing the network to generate lower over-activation values in the shallow layers.

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

$$\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta})) = \frac{1}{|L_\mathcal{S}| \cdot |\bar{M}| \cdot C^{(l)}} \sum_{l \in L_\mathcal{S}} \sum_{c} \left| \sum_{i,j} (\boldsymbol{z}^{(l)}_{c,i,j} \odot \bar{M}) \right|, \tag{14}$$

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 shallow layers $L_\mathcal{S}$ 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 $l \in L_\mathcal{S}$ corresponding only to those neurons that are spatially associated to the adversarial patch δ.

Please note that, although we set $\beta_0 = 0.5$ to train the soft-thresholding blocks, the proposed defense is able to mask or partially mask also patches with lower values of β, such that their adversarial effect is notably reduced (see Figure 15b and experiments carried out changing the values of β).

D Visualization of the over-activations

Z-mask is founded on certain assumptions on the inner layer over-activation values induced by physical-realizable adversarial attacks. To validate these assumptions, Figure 10 shows the over-activations computed by *Z-Mask* in a shallow layer and a deep layer of RetinaNet and DDRNet. For a fair visual comparison, we also considered the not patched image and a random patch.

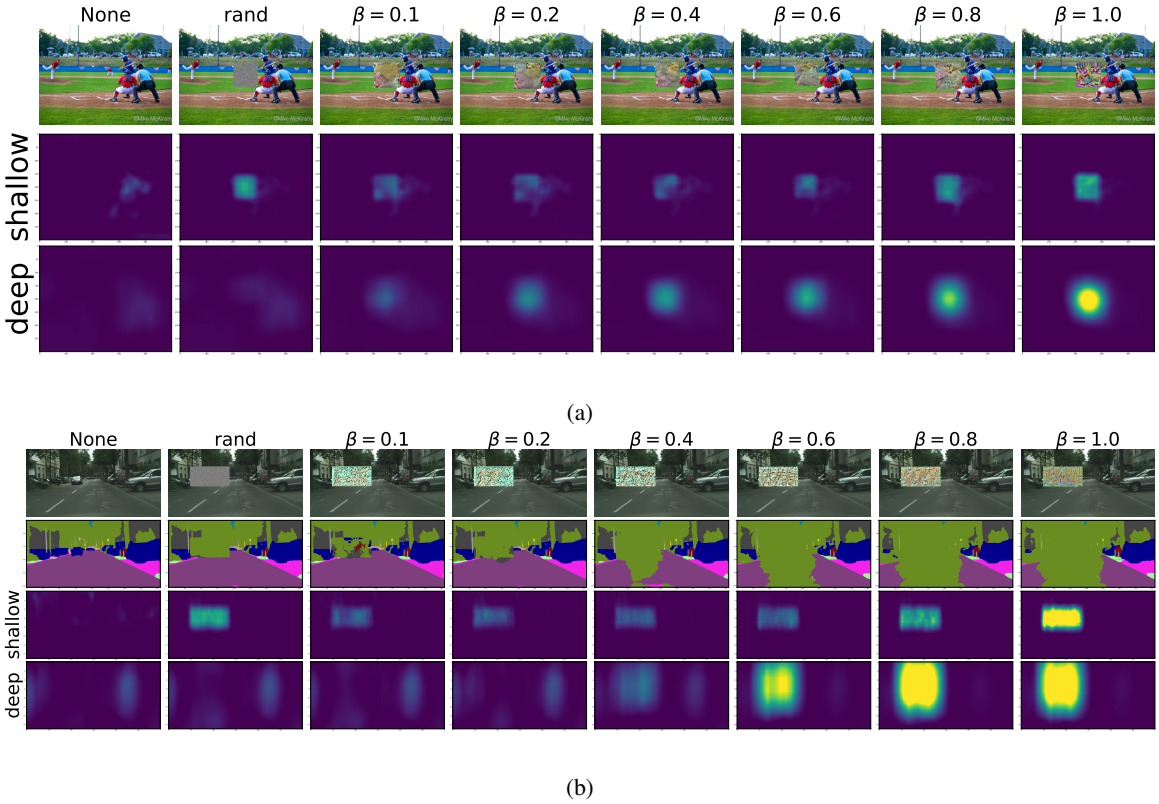
The figure shows that higher β yields larger over-activations in the shallow layers. Note that even a random patch (generated with a uniform noise function) induces strong over-activations in the shallow layers. However, they are not propagated in deep layers, which is consistent with the main assumption.

Note that the larger the over-activations in the deep layer (see Figure 10b), the larger the adversarial effect in the output. This illustration provides an additional insight on the correlation between dangerous adversarial patches and the corresponding over-activation. Going deeper with a formal analysis of the model behaviour for understanding such an evident correlation is left as a future work.

Please, also note that, as reported in the main manuscript and in Table 1, to extract $\mathcal{H}^\mathcal{D}$ shallow layers were also added to deep layers. This allows identifying also patches that are not adversarial, but still produce high over-activations in the shallow layers (e.g., random patches). We believe that this approach could help extending our defense mechanism to identify possible out-of-distribution objects, which are not necessarily adversarial. However, extending and testing this mechanism to recognize out-of-distribution objects is left as a future work.

E Details on the Spatial Pooling Refinement

This section provides a visualization of the benefits obtained with the Spatial Pooling Refinement that performs a cascade filtering of multiple Average Pooling Operators with different kernel sizes.



(a)

(b)

Figure 10: Illustration of the over-activation values for different values of β induced by adversarial patches in shallow and deep layers for RetinaNet (a) and DDRNet (b).

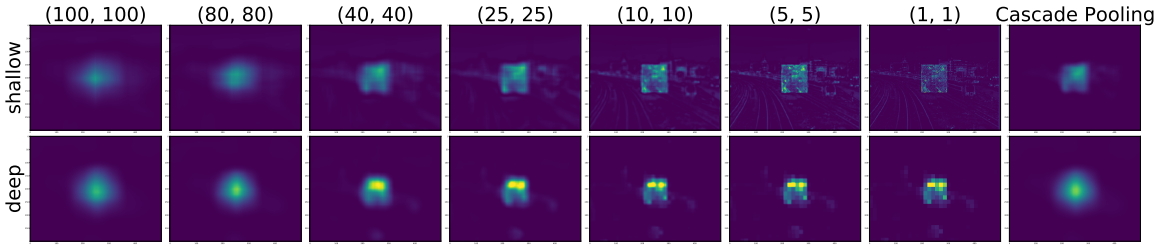
Figure 11 reports the over-activated areas computed through several strategies. This analysis considers the same images analyzed in Figure 10, using $\beta = 0.6$ for both RetinaNet on COCO and DDRNet on Cityscapes. In particular, Figure 11 compares the over-activation analysis obtained by a single Average Pooling operator having different kernel sizes and the proposed Spatial Pooling Refinement (denoted as Cascade Pooling and implemented with the same settings mentioned in the main paper). We recall that the goal of the analysis in shallow layers is to identify the areas (even not adversarial) that cause over-activations with high spatial accuracy, whereas the goal in the deep layers is to identify such areas that contain only adversarial objects, although with lower spatial accuracy.

As shown in Figure 11, the Spatial Pooling Refinement achieves the desired goal both in the shallow and deep layers. Conversely, single Average Pooling operations obtain lower performance. In fact, small kernels prevent from capturing spatial contiguous over-activation in shallow layers. Furthermore, also other non-adversarial objects are identified, without posing attention only to the adversarial patch. Conversely, large kernels allow extracting better regions in deep layers, but their application in shallow layer does not help identify the adversarial objects accurately.

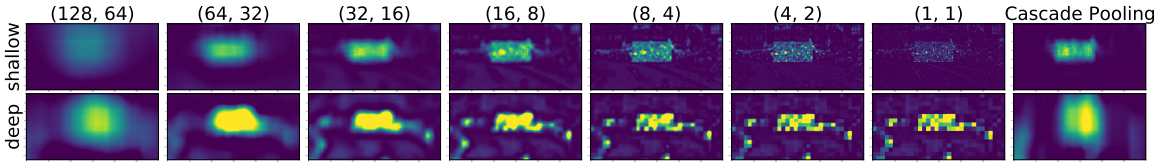
The cascade filtering operation helps filter out non-adversarial objects. Also, comparing the Spatial Refinement Pooling with medium kernel sizes, the former obtains a better identification of the areas outside the patch, both in shallow and deep layers. This helps avoid masking other objects that do not cause adversarial effects.

F Details of related works

This section provides further details about the related works implemented in the main manuscript to assess the benefits of our proposal with respect to the state of the art.



(a)



(b)

Figure 11: Illustration of the advantages of using the Spatial Pooling Refinement over single Pooling operations.

Related works for adversarial detection The proposed method (*Z-Mask*) has been compared with HyperNeuron (HN) [9] and the Fast Patch Detection Algorithm (FPDA) [35]. HN is a lightweight algorithm that detects the presence of adversarial patches by computing a score from the *Z-score* extracted from all the features given from a certain layer. Such a score is then compared with a pre-computed threshold to discriminate safe and unsafe inputs. FPDA performs similar operations. The only difference is that all the features are first compressed through the channel dimension to reduce the computation time for real-time applications, while keeping a good performance. Since both the algorithms evaluate the over-activations in a specific network layer, we compared their performance in the last deep layer, i.e., the deeper layer used for extracting $\mathcal{D}$ in *Z-Mask*.

Related works for adversarial masking For adversarial masking, we implemented two run-time approaches: MaskNet [6] and Local Gradient Smoothing [27]. The first approach uses an additional DNN (denoted as MaskNet) to generate a defense mask, which is then applied to the original image to filter out adversarial patches. Keeping the same approach presented in [6], we used a U-Net model [33] as MaskNet architecture, which was trained using the same settings and adversarial strategy:

$$\underset{\epsilon}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\mathbf{\Gamma}}\, \{\max_{\delta} \mathcal{L}(f(\tilde{x} \odot M_{\epsilon}(\tilde{x})), y))\} \tag{15}$$

where $\tilde{x} = g_{\gamma}(\mathbf{x}, \boldsymbol{\delta})$ and M_{ϵ} is the MaskNet model with its trainable parameters ϵ.

Local Gradient Smoothing (LGS) filters out high-frequency areas in the image domain. It is based on the assumption that localized adversarial attacks consists of high-frequency pixel variations, so they could be recognized by studying the image gradient. Also in this case, we re-implement the latter approach using the same settings provided in [27].

Figure 12 illustrates the defense mask produced by *Z-Mask*, *MaskNet*, and *LGS*. Notice how *Z-Mask* is able to better identify the mask, without corrupting other portions of the image. Also note that only *Z-Mask* returns a binary Mask.

LGS detects patch's edge precisely, however the mask values are not enough to fully mitigate the patch adversarial effect. Furthermore, LGS can mask other high-frequency objects, so reducing the nominal performance of the model.

MaskNet achieves a better performance with respect to LGS, but the learning strategy adopted is more expensive and not always capable of generalizing among multiple models [6]. This is because the optimization function takes into account the task-specific model loss, which also aims at improving the model performance without focusing only on mitigating the adversarial effect of the patch. This problem causes a masking of other portions of the image, which could jeopardize the nominal performance of the model.

(a) Z-Mask

(b) MaskNet

(c) Local Gradient Smoothing

Figure 12: Comparison of the masks obtained with different methods: (a) *Z-mask*, (b) MaskNet, and (c) LGS.

G Additional results

This section reports additional results on the adversarial detection/masking performance and the defense-aware analysis.

Figure 13 confirms the benefits of *Z-Mask* also for object detectors (RetinaNet and Faster R-CNN) against patches with different values of β (i.e., different over-activation values in the shallow layers). In fact, the proposed mechanism is able to perform well both on adversarial detection and masking against adversarial patches crafted with different values of β. For adversarial detection (top plots), *Z-Mask* always outperforms HN and FPDA, both for low and high values of β.

Concerning the defense masking task (bottom plots), again *Z-Mask* allows keeping nominal model performance also when adversarial patches are used, for both low and high values of β. In particular, focusing on the case of Faster R-CNN (Figure 13(b)), also MaskNet achieves a good defense performance. We omitted the results of MaskNet on RetinaNet (Figure 13(a)) because of the training issues already mentioned above and in the main paper. On the other hand, although LGS achieves good results against patches with low adversarial effects (i.e., low values of β), its performance drops with high adversarial patches, having a trend similar to the original model without defense.

Figure 14 shows the performance of *Z-Mask* and MaskNet against defense-aware attacks described in details in the main paper. Figure 14a) confirms for ICNet the same results discussed in the main paper about DDRNet. For Faster R-CNN (see Figure 14b), both MaskNet and *Z-Mask* achieved robust behaviours against defense-aware attacks.

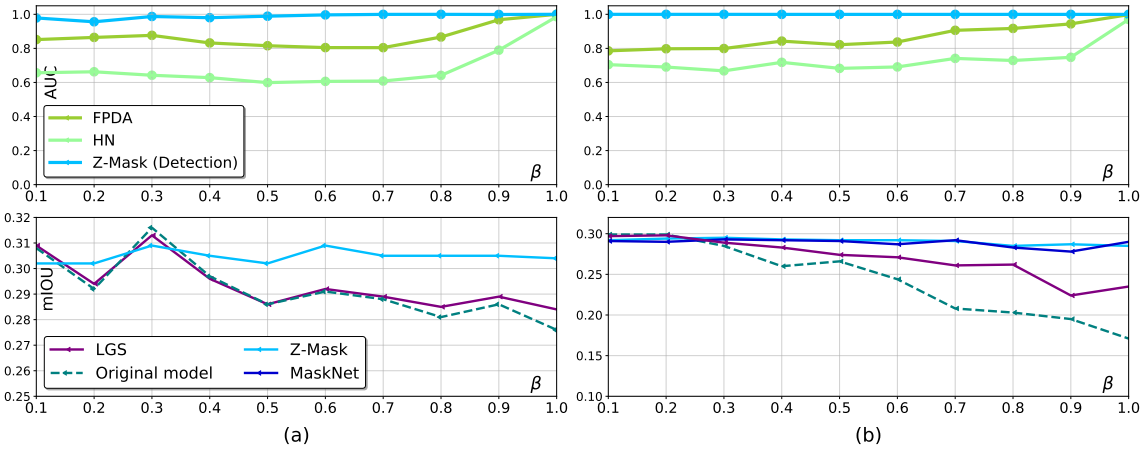


Figure 13: Comparison of detection and masking performance of *Z-mask*, MaskNet, and LGS for (a) RetinaNet and (b) Faster R-CNN against patches with different values of β.

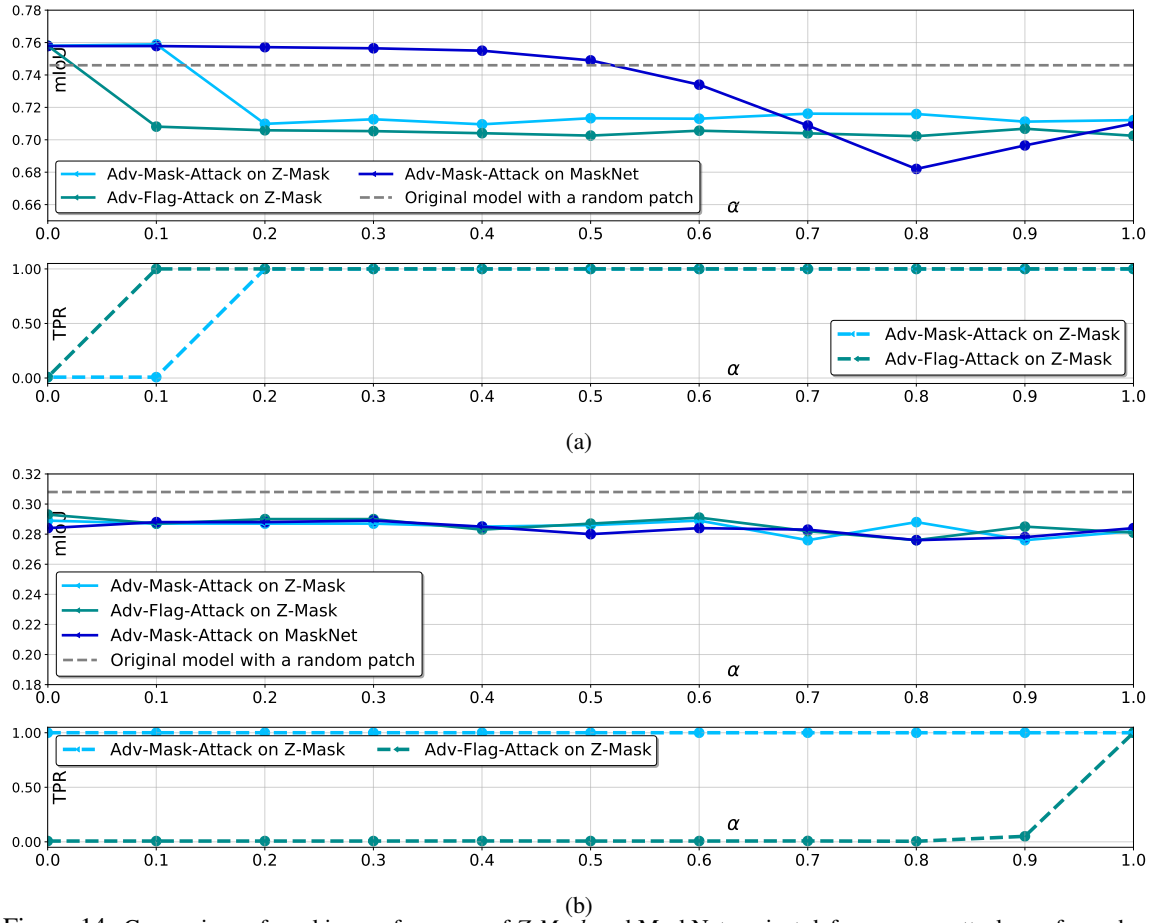
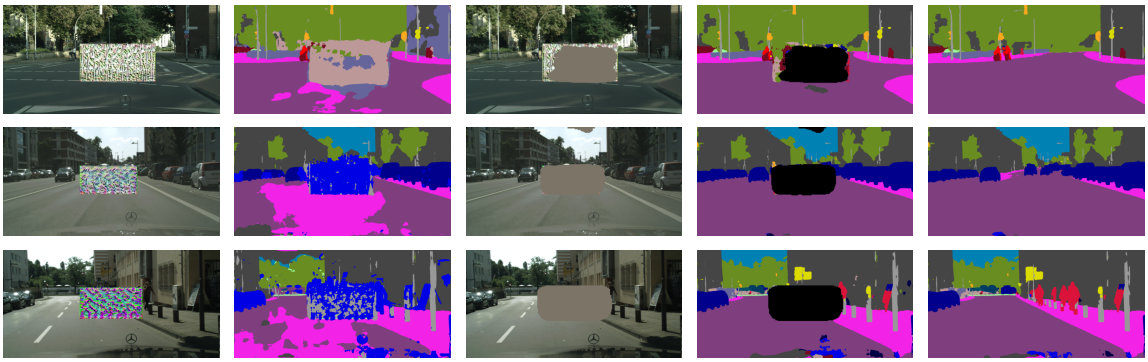


Figure 14: Comparison of masking performance of *Z-Mask* and MaskNet against defense-aware attacks performed on ICNet (a) and Faster R-CNN (b).

(a)



(b)

Figure 15: Masking Performance on digital images extracted from the COCO dataset (a) and Cityscapes (b).

G.1 Additional illustrations

Here we report other illustrations of the results achieved by the proposed algorithm for both the digital domain (Figure 15) and physical domain (Figure 16). In detail, images in Figure 15 were collected from the validation sets of COCO (using RetinaNet) and Cityscapes (using BiseNet). Figure 15b presents three different patches applied to BiseNet, in particular first row shows the effects with $\beta = 0.4$, second row with $\beta = 0.8$, and third row with $\beta = 1.0$ (full adversarial patch).

Figure 16 reports images collected from Apricot (a) and our private dataset (b). Also in these cases, *Z-Mask* was able to cover high portions of the tested adversarial patches, both in the digital and physical domains.

(a)

(b)

Figure 16: Masking performance on real-world images. (a) contains images of the Apricot [2] dataset, while (b) are private images.