Supplementary material for the paper "Defending From Physically-Realizable Adversarial Attacks Through Internal Over-Activation Analysis"

Anonymous Authors

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 [6][1] 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 [14][2] with its pretrained weights. The applied version uses an Xception39 model [4] as backbone.

ICNet The original pretrained model[3] provided by the authors [15] (trained using the Caffe framework [7]) was imported in PyTorch.

A.2. Object detection

The object detection models used in the paper were downloaded from the PyTorch model zoo[4]: 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

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

[1] https://github.com/ydhongHIT/DDRNet
[2] https://github.com/ycszen/TorchSeg
[3] https://github.com/hszhao/ICNet
[4] 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.

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.

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 [10], 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

$$\mathcal{L}_{adv-eff,S}(f(\mathbf{x}), \mathbf{y}_{Adv}) = w_{class}\mathcal{L}_{CE}(f_{class}(\mathbf{x}), \mathbf{y}) + w_{regr}\mathcal{L}_1(f_{bbox}(\mathbf{x}), \mathbf{y}_{Adv}) \tag{2}$$

for the single-stage detectors[5] and

$$\begin{aligned}\mathcal{L}_{adv-eff,T}(f(\mathbf{x}), \mathbf{y}_{Adv}) = \mathcal{L}_{adv-eff,S}(f(\mathbf{x}), \mathbf{y}_{Adv}) \quad + \\ w_{RPN-class}\mathcal{L}_{CE}(f_{RPN-class}(\mathbf{x}), \mathbf{y}_{Adv}) \quad + \\ w_{RPN-regr}\mathcal{L}_1(f_{RPN-bbox}(\mathbf{x}), \mathbf{y}_{Adv})\end{aligned} \tag{3}$$

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.

[5] 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.

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 [12] 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 $\mathbf{y}_{Adv} = \mathbf{y}$.

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 [8] (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 [13] 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

$$\mathcal{L}_N(p) = \prod_{c \in C} \|p - c\| \tag{4}$$

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

$$\mathcal{L}_S = \sum_i \sum_j [(\delta_{i+1,j} - \delta_{i,j})^2 + (\delta_{i,j+1} - \delta_{i,j})^2] \tag{5}$$

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

$$\mathcal{L}_{Adv} = w_{adv-eff}\mathcal{L}_{adv-eff} + w_N\mathcal{L}_N + w_S\mathcal{L}_S \tag{6}$$

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 (1), 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:

$$\hat{\boldsymbol{\delta}}_\beta = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \{\mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\boldsymbol{\Gamma}} \left[(1-\beta)\cdot \ \mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta})) + \beta \cdot \ \mathcal{L}_{Adv}(f(g_\gamma(\mathbf{x}, \boldsymbol{\delta})), \mathbf{y})\right]\}, \tag{7}$$

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:

$$\mathcal{L}_{OZ}(f, g_\gamma(\mathbf{x}, \boldsymbol{\delta})) = \frac{1}{|L_\mathcal{S}| \cdot |\bar{M}| \cdot C^{(l)}} \sum_{l \in L_\mathcal{S}} \ \sum_{c} \left| \sum_{i,j} (\boldsymbol{z}^{(l)}_{c,i,j} \odot \bar{M}) \right|, \tag{8}$$

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 7b and experiments carried out changing the values of β).

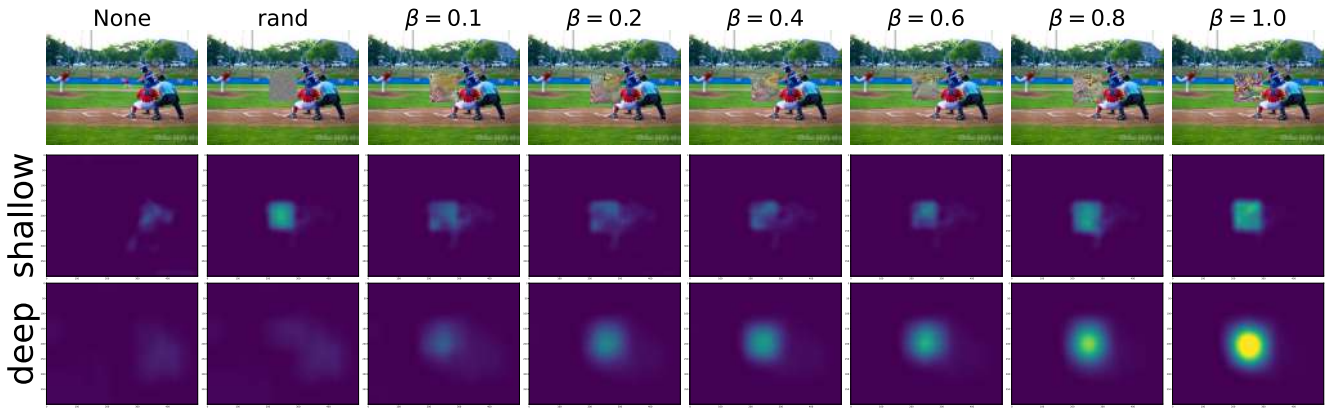
D. Visualization of the over-activations

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 1 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.

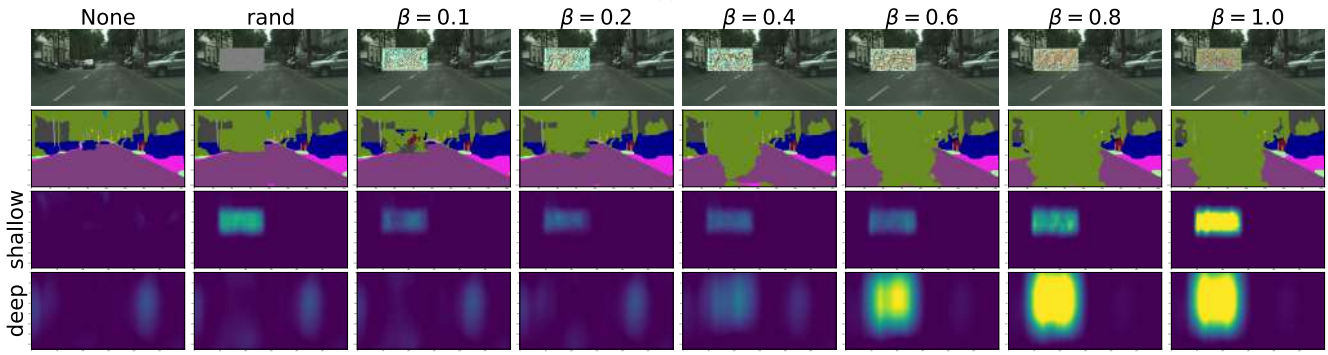
The figure shows that higher β yields larger over-activations in the shallow layers. Note that even a random patch (generated with a uniform noise function) induces strong over-activations in the shallow layers. However, they are not propagated in deep layers, which is consistent with the main assumption.

Note that the larger the over-activations in the deep layer (see Figure 1b), 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.



(a)



(b)

Figure 1: 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).

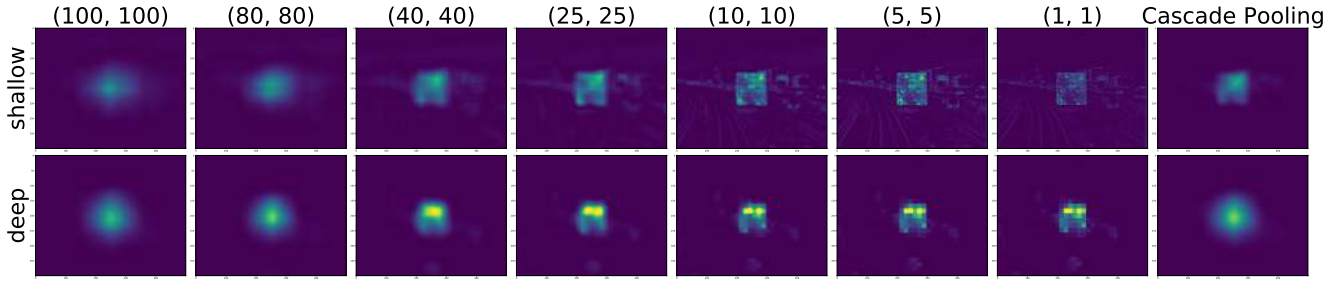
E. Details on the Spatial Pooling Refinement

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.

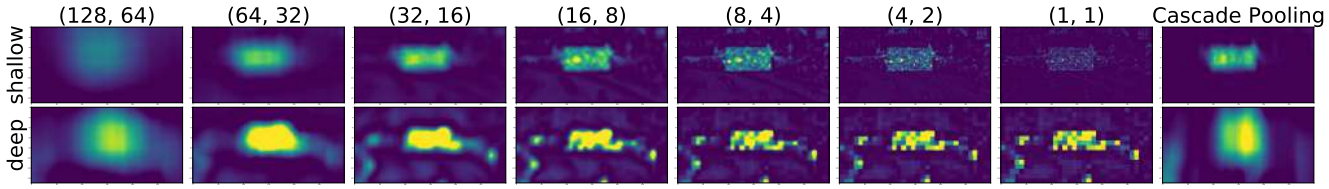
Figure 2 reports the over-activated areas computed through several strategies. This analysis considers the same images analyzed in Figure 1, using $\beta = 0.6$ for both RetinaNet on COCO and DDRNet on Cityscapes. In particular, Figure 2 compares the over-activation analysis obtained by a single Average 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 2, 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.



(a)



(b)

Figure 2: Illustration of the advantages of using the Spatial Pooling Refinement over single Pooling operations.

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.

Related works for adversarial detection The proposed method (*Z-Mask*) has been compared with Hyper-Neuron (HN) [5] and the Fast Patch Detection Algorithm (FPDA) [12]. 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 [3] and Local Gradient Smoothing [9]. 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 [3], we used a U-Net model [11] as MaskNet architecture, which was trained using the same settings and adversarial strategy:

$$\underset{\epsilon}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\gamma\sim\mathbf{\Gamma}}\, \{\max_{\delta} \mathcal{L}(f(\tilde{x} \odot M_\epsilon(\tilde{x})), y))\} \tag{9}$$

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 [9].

Figure 3 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 [3]. 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 3: 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 4 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 4(b)), also MaskNet achieves a good defense performance. We omitted the results of MaskNet on RetinaNet (Figure 4(a)) because of the training issues already mentioned above and in the main paper. On the other hand, although LGS achieves

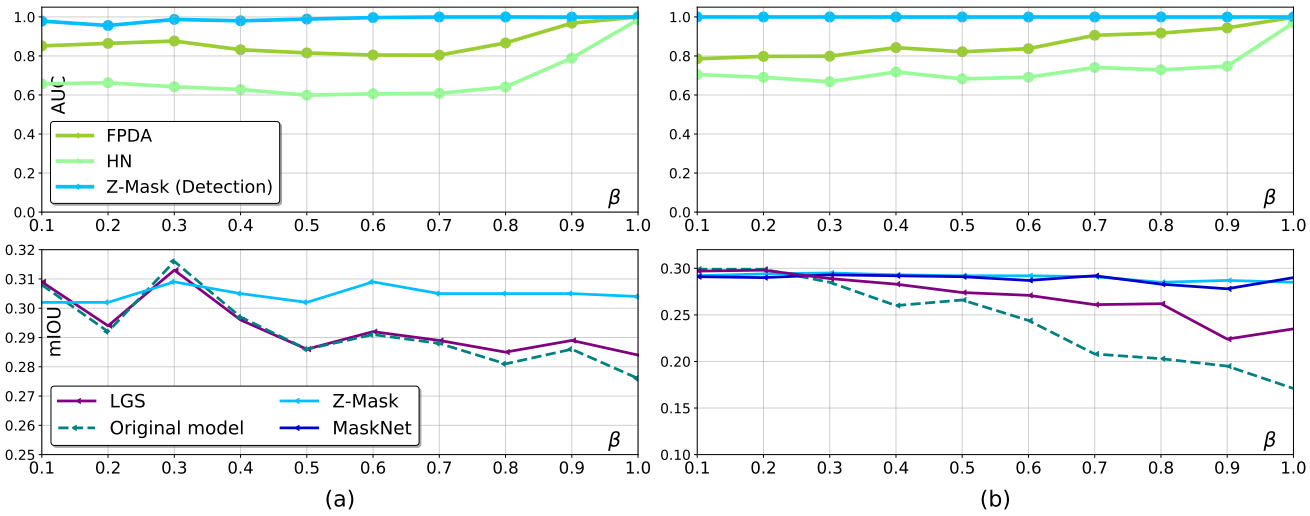


Figure 4: Comparison of detection and masking performance of *Z-mask*, MaskNet, and LGS for (a) RetinaNet and (b) Faster R-CNN against patches with different values of β.

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 5 shows the performance of *Z-Mask* and MaskNet against defense-aware attacks described in details in the main paper. Figure 5a) confirms for ICNet the same results discussed in the main paper about DDRNet. For Faster R-CNN (see Figure 5b), both MaskNet and *Z-Mask* achieved robust behaviours against defense-aware attacks.

G.1. Illustrations of the over-activation analysis

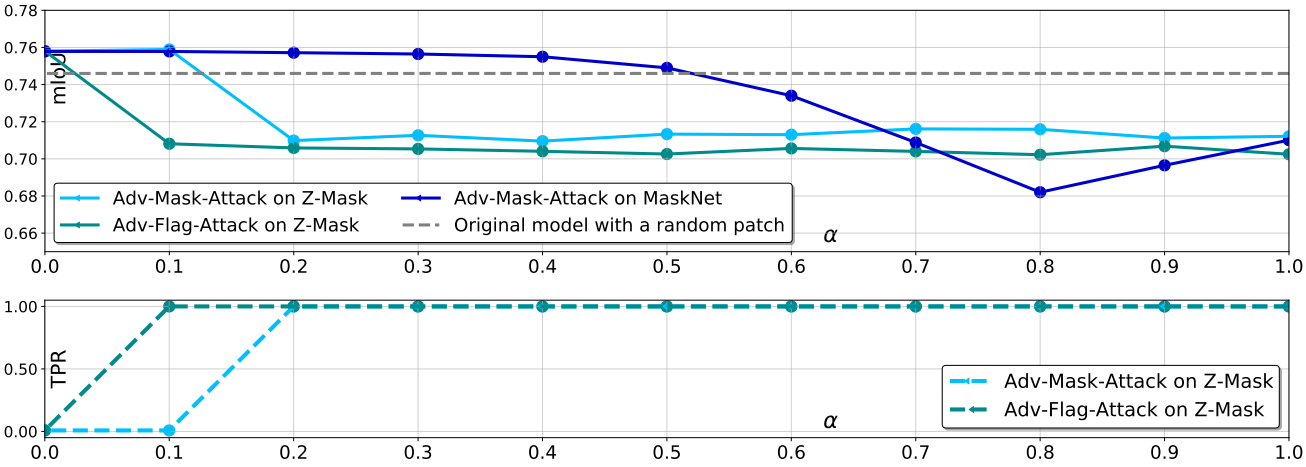
Figure 6 illustrates the empirical analysis carried out to identify those layers that are more susceptible to have over-activations against adversarial objects, In particular, it shows the heatmaps corresponding to the first five layers of the DDR-Net, which are obtained with different β-patches ($0.4, 0.6, 1.0$, respectively) by the Z-Score analysis of the SPR block.

As shown in the figure, the higher is the β value the higher is the over-activation caused from the adversarial patch. In this regard, we noticed that pure adversarial patches ($\beta = 1.0$) usually over-activate almost all the networks layers. However, patches with lower values of β only over-activate specific layers. That said, we remark that providing an automatic procedure to understand in advance those layers, as well supported with theoretical contributions, is not a straightforward task and requires further investigations.

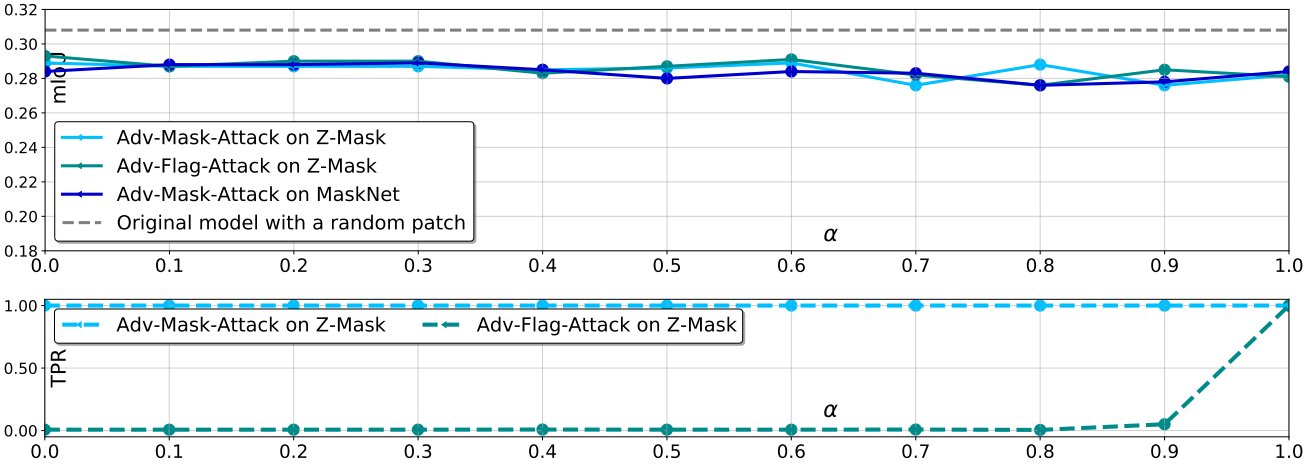
G.2. Additional illustrations

Here we report other illustrations of the results achieved by the proposed algorithm for both the digital domain (Figure 7) and physical domain (Figure 8). In detail, images in Figure 7 were collected from the validation sets of COCO (using RetinaNet) and Cityscapes (using BiseNet). Figure 7b 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 8 reports images collected from Apricot (a) and our private dataset (b) and an image provided by [12]. Also in these cases, *Z-Mask* was able to cover large portions of the tested adversarial patches, both in the digital and physical domains.



(a)



(b)

Figure 5: Comparison of masking performance of *Z-Mask* and MaskNet against defense-aware attacks performed on ICNet (a) and Faster R-CNN (b).

References

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 284–293, 2018.

[2] A Braunegg, Amartya Chakraborty, Michael Krumdick, Nicole Lape, Sara Leary, Keith Manville, Elizabeth Merkhofer, Laura Strickhart, and Matthew Walmer. Apricot: A dataset of physical adversarial attacks on object detection. In *European Conference on Computer Vision*, pages 35–50. Springer, 2020.

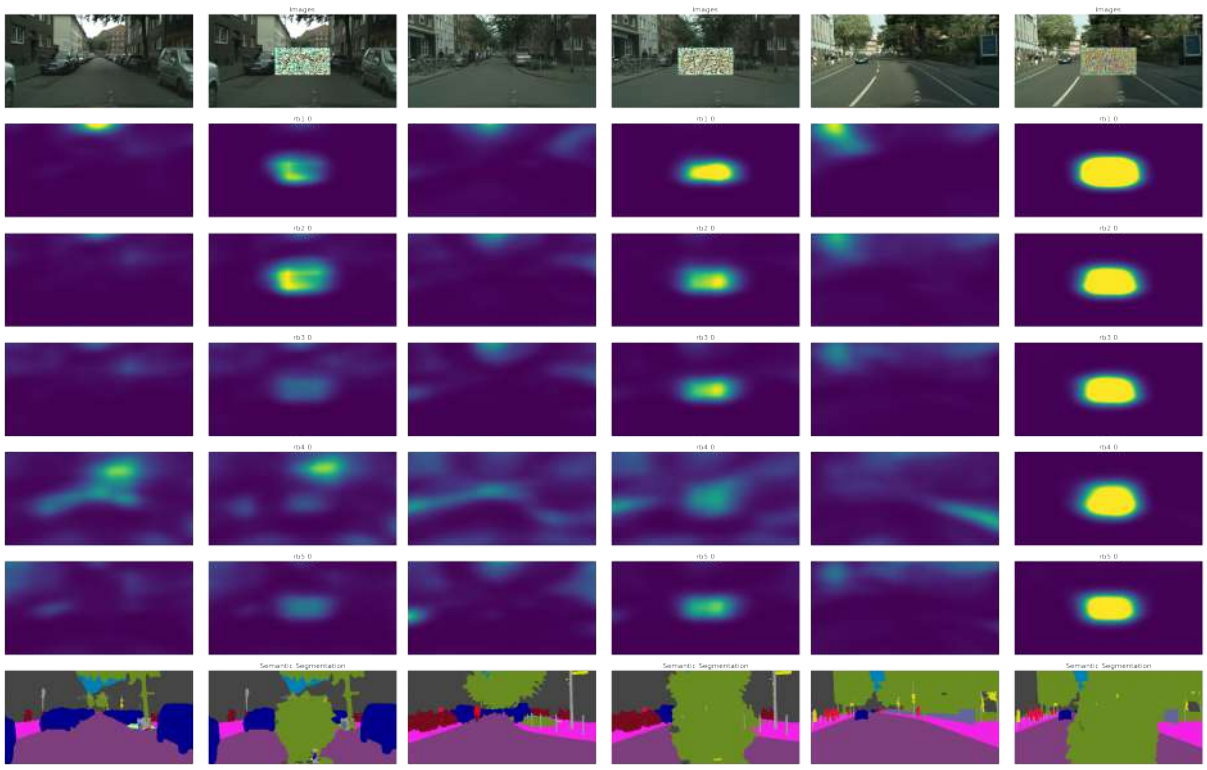
[3] Ping-Han Chiang, Chi-Shen Chan, and Shan-Hung Wu. Adversarial pixel masking: A defense against physical attacks for pre-trained object detectors. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, pages 1856–1865. Association for Computing Machinery, 2021.

[4] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv e-prints*, page arXiv:1610.02357, Oct. 2016.

[5] Kenneth T Co, Luis Muñoz-González, Leslie Kanthan, and Emil C Lupu. Real-time detection of practical universal adversarial perturbations. *arXiv:2105.07334*, 2021.

[6] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv:2101.06085*, 2021.

[7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.



(a) $\beta = 0.4$ (b) $\beta = 0.6$ (c) $\beta = 1.0$

Figure 6: Over-activation analysis in the first five layers of DDRNet. Each heatmap shows the spatial Z-Score analysis obtained from the SPR blocks. Several patches having different β were tested.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 86–94. IEEE Computer Society, 2017.

[9] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[10] 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 *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2826–2835. IEEE Computer Society, 2022.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

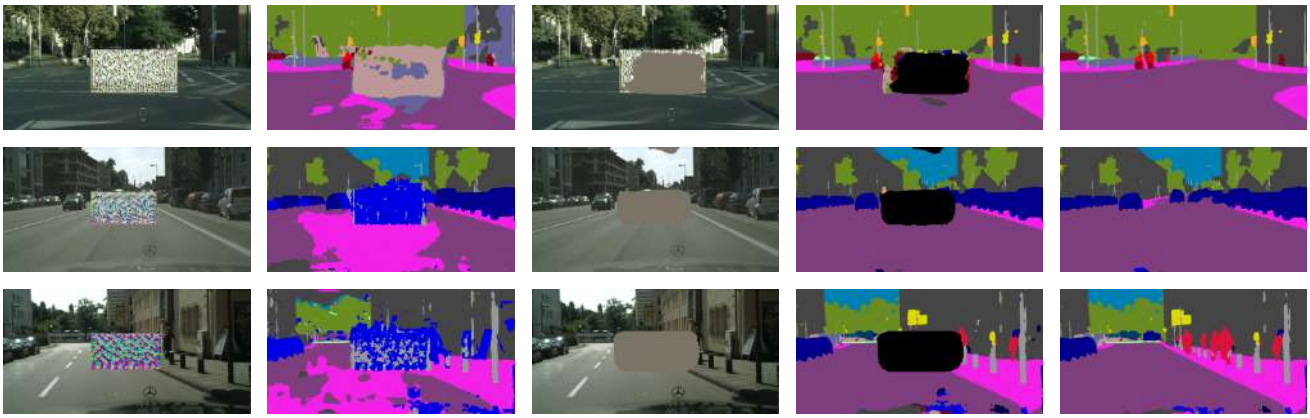
[12] Giulio Rossolini, Federico Nesti, Gianluca D'Amico, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. On the Real-World Adversarial Robustness of Real-Time Semantic Segmentation Models for Autonomous Driving. *arXiv:2201.01850*, 2022.

[13] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540, Vienna Austria, Oct. 2016. ACM.

[14] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341. Springer, 2018.

[15] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420. Springer, 2018.

(a)

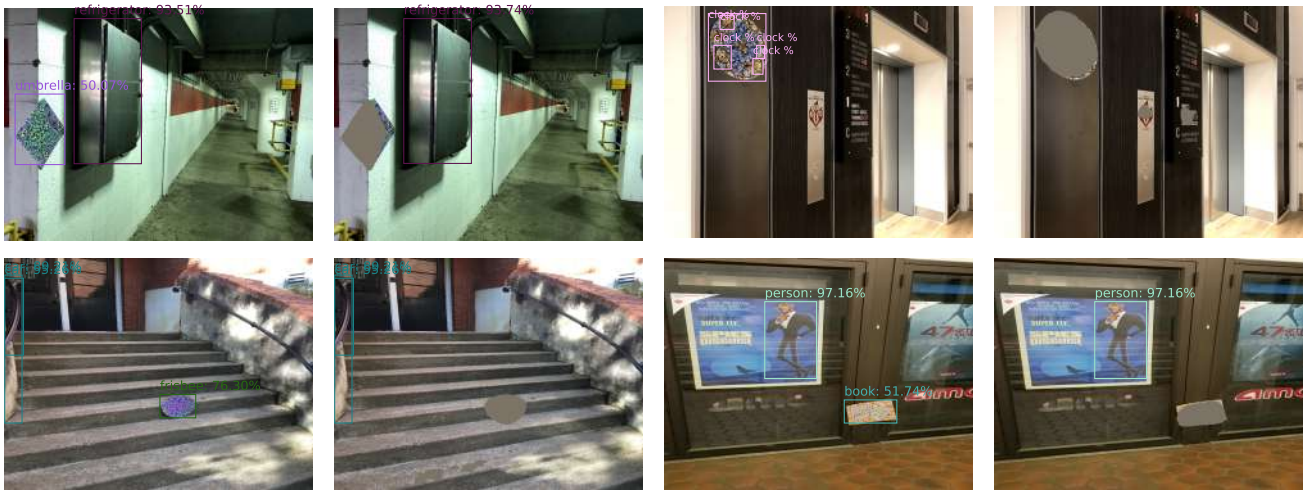


(b)

(c)

(d)

Figure 7: Masking Performance on digital images extracted from the COCO dataset (a) and Cityscapes with Bisenet (b), DDRNet (c) and ICNet (d).



(a)

(b)

(c)

Figure 8: Masking performance on real-world images. (a) contains images of the Apricot [2] dataset, while (b) are private images and (c) is an input image provided by [12]