SEGGUARD: DEFENDING SCENE SEGMENTATION AGAINST ADVERSARIAL PATCH ATTACK

Thomas Gittings* *Steve Schneider** *John Collomosse**†

* Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey. UK.
† Adobe Research, San Jose. USA.

ABSTRACT

Adversarial Patch Attacks (APAs) induce prediction errors by inserting carefully crafted regions into images. This paper presents the first defence against APAs for deep networks that perform semantic segmentation of scenes. We show that a conditional generator can be trained to produce patches on demand targeting specific classes and achieving superior performance versus conventional pixel-optimised patch attacks. We then leverage this generator along with the segmentation network as part of a generative adversarial network, which trains the model to ignore the adversarial patches produced by the generator, while simultaneously training the generator to produce updated patches to attack the fine-tuned network. We show that our process confers strong protection against adversarial patches, and that this protection generalises to traditional pixel-optimised adversarial patches.

***Index Terms*—** Semantic segmentation, Adversarial Patches, Adversarial Attack.

1. INTRODUCTION

Semantic segmentation is a fundamental computer vision task underpinning applications such as robotics, autonomous driving, and medical imaging. Contemporary segmentation methods are enabled via convolutional neural networks (CNNs) and so are prone to adversarial attacks; minor modifications to images that induce a significant change in the network prediction [1, 2]. Adversarial attacks may be covert, via imperceptible changes distributed across an image [2, 3], or overt via *adversarial patch attacks* (APAs) that introduce visible changes localised to a particular region or 'patch' [4, 5, 6]. The majority of adversarial attacks target CNNs trained for image classification. Recently, APAs have been developed [7] for models performing *semantic segmentation*, *e.g.*DDRNet [8], BiSeNet [9] and ICNet [10]. Given the importance of semantic segmentation networks it is vital that their robustness to these adversarial patches is studied and improved. Additionally, semantic segmentation provides a well defined context in which to study APAs, since it is clear that a patch should have no influence outside of the area it covers. Semantic segmentation also enables the study of different forms of targeted attack, such as insertion and deletion of objects in a scene.

This paper contributes the first training-time defence against APAs that target semantic segmentation networks. APAs targeting segmentation have been sparsely researched and, consequently, few defences exist. Our core technical contribution is to harness adversarial training to improve the resilience of segmentation models at training time. Such training need not be applied from scratch, enabling pre-trained models to be fine-tuned via our method in order to confer protection against APAs. Our training-time defence utilises a conditional Generative Adversarial Network (cGAN) that synthesizes patches to attack the segmentation network, whilst simultaneously improving the resilience of that network against those patches. The generator synthesizes adversarial patches which successfully attack the network, forming a second contribution of our work. We show that our framework confers protection against APAs to the segmentation network which generalizes beyond our own APAs. Therefore our two core technical contributions are as follows:

1. Segmentation Defence We show that adversarial training can be used to defend a semantic segmentation network against adversarial patch attacks, and that the protection provided by this extends to unseen state of the art attacks. We can achieve this protection against untargeted attacks (those which aim to decrease the mIoU of the model) and targeted attacks which seek to insert a specified target class in to a scene.

2. Conditional Generative Patches We show that a conditional patch generator can produce effective adversarial patches against semantic segmentation networks, and that these patches are more effective than the baseline in certain settings. Existing methods of producing adversarial patches train at the pixel level, with optimisation taking upwards of an hour to converge, making them unviable to use in a training loop to protect a model from attack. Hence we make use of a conditional generator, which can produce the adversarial patches via a forward inference pass, and be continuously updated during training in order to produce patches that attack the latest version of the segmentation model.

2. RELATED WORK

Adversarial Images are created by the strategic introduction of small perturbations to real images, causing them to be misclassified. Early methods [1] were slow, but later methods can produce adversarial examples in a single optimisation step [2, 11].

Adversarial Patches [4] restrict the perturbation to a small region of the image, but allow it to be arbitrary in magnitude.

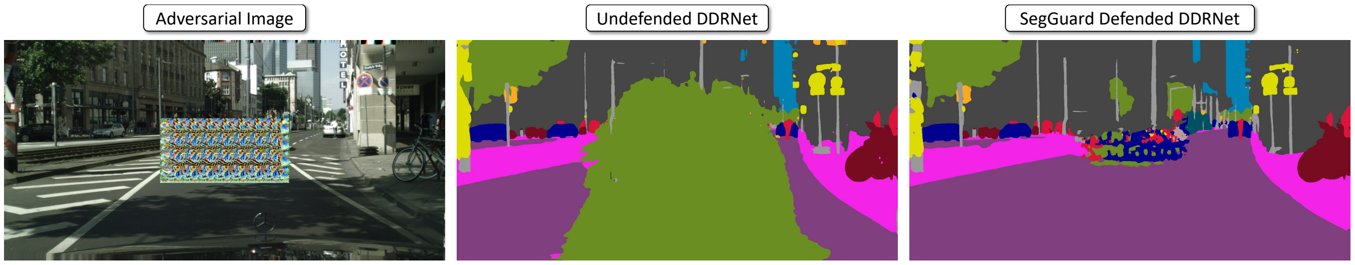


Fig. 1. SegGuard provides protection against adversarial patch attacks (APAs) for semantic segmentation. An undefended DDRNet (middle) is fooled when presented with an image containing an adversarial patch (left), detecting a large spurious region of class 'vegetation', whereas our SegGuard Defended DDRNet (right) is only affected within the boundaries of the patch itself, even correctly inferring the segmentation for some areas covered by the patch.

This allows for an attack that is printable and robust to affine transformations. Gittings *et al.*[6] added deep image prior regularisation, which creates patches that have a different appearance and could potentially elicit a different response from a defended network. Eykholt *et al.*[5] introduced robust physical perturbations specifically targeting road sign classification.

Adversarial Patches for Object Detection and Segmentation. Liu *et al.*[12] created an adversarial patch for detection causing all objects to be ignored, but the patch must be inserted digitally into the image, so it is not useful as a physical world attack. Chen *et al.*[13] and Eykholt *et al.*[14] created adversarial patches for object detection using stop signs.Nesti *et al.*[7] created a real-world adversarial patch for Semantic Segmentation, with versions designed to decrease the mIoU of the segmentation, as well as to transform one specific class into another..

Defences Against Adversarial Images. Szegedy *et al.*[1] proposed the *adversarial training*, where adversarial examples are created during the training of the model and then included as part of the training data. For this method to be practical a fast attack method must be available [2]. Kurakin *et al.*[15] applied adversarial training to a large dataset. Papernot *et al.*[16] introduced the distillation defence, which retrains the network so that an attacker cannot backprop through it. Meng and Chen [17] attempted to remove adversarial perturbations at inference time, by using an autoencoder to project the input onto the manifold of natural images. Others achieve similar using a GAN [18].

Defences Against Adversarial Patches. There are relatively few defences against adversarial patch attacks. Hayes [19] created the first such defence, by creating a saliency map using guided backpropagation, which is then filtered to find any localised regions of high saliency, which are then assumed to be patches and inpainted. Naseer *et al.*[20] observed that adversarial patches often form concentrated regions of high gradient in an image. They apply localised gradient smoothing (LGS), which blurs areas of high gradient when they are concentrated together in groups. Abdel-Hakim [21] introduced the Ally Patch defence, which splits the image into multiple small regions of interest, each of which is classified separately and a vote is taken amongst the regions to decide on the final classification. Since a patch is unlikely to appear in more than one region, this reduces its impact. All of these inference-time methods degrade the classification performance on clean images and slow the speed of inference. Gittings *et al.*[22] applied adversarial training to patches, using a conditional patch generator to allow the attack to be updated during training.

3. METHODOLOGY

In this work we study adversarial patch attacks on semantic segmentation networks. The primary goal of these attacks is to reduce the overall performance of the network, as measured by the mean Intersection Over Union (mIoU) over the evaluation classes. In some experiments we further focus our goal via *targeted attacks*. These can target the *insertion* or *deletion* of a given *target class*, but in the main paper we focus on insertion attacks. We restrict our attention to state of the art real-time semantic segmentation networks, in particular DDRNet [8], BiSeNet [9] and ICNet [10]. All the networks used in this paper are trained on the Cityscapes dataset [23], and designed for use in autonomous driving. Our defence is inspired by Vax-a-Net [22], which is a training-time defence against adversarial patches for image classification. The architecture (Figure 2) synthesises patches using a conditional generative network (G), and applies an adversarial training process to update the generator while simultaneously training the segmentation network (f) to build resistance to the patches.

3.1. Adversarial Patches for Semantic Segmentation

An adversarial patch for a semantic segmentation network f is a patch p which is designed to cause f to perform poorly on scenes containing p ($e.g.$Figure 1). Formally we define $A(p, x, l, t)$ to be the image x on which the patch p has been applied at location l with affine transformation t. We call this the *application* of p to x.

For our main experiments we sample l randomly from all possible patch locations during both training and testing. This choice is explored fully in the supmat. For t we randomly scale the patch up or down by 20% and we do not rotate the patch. During the training process the image x is sampled from the full Cityscapes training set, in order to produce patches which are as universal as possible.

Loss Function. To create our adversarial patches we use the loss function proposed by Nesti *et al.*[7] with an adaptive value

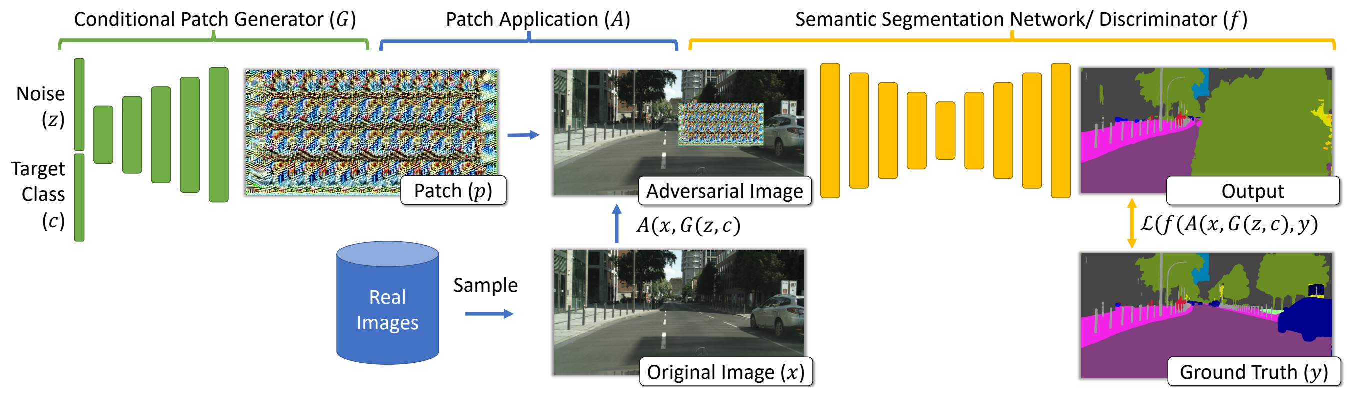


Fig. 2. Proposed architecture for defending semantic segmentation models against adversarial patches (APAs). The conditional patch generator G synthesises patches targeting all 19 evaluation classes of Cityscapes [23]. We alternate training of f and G to encourage the model resilience to unseen attacks.

of γ. Define $\mathcal{N} = \{1, \ldots, H \times W\}$ to be the set of output pixels, and $\tilde{\mathcal{N}} = \{1, \ldots, \tilde{H} \times \tilde{W}\} \subset \mathcal{N}$ the pixels which correspond to the adversarial patch. Then define the set of pixels which are not part of the patch and which are currently classified correctly according to target segmentation map y

$$\Upsilon = \{i \in \mathcal{N} \setminus \tilde{\mathcal{N}} \mid f_i(\tilde{x}) = y_i\}. \quad (1)$$

Now we can define two terms for the loss function:

$$\mathcal{L}_M(f(\tilde{x}), y) = \sum_{i \in \Upsilon} \mathcal{L}_{\text{CE}}(f_i(\tilde{x}), y_i), \quad (2)$$

$$\mathcal{L}_{\bar{M}}(f(\tilde{x}), y) = \sum_{i \in \bar{\Upsilon}} \mathcal{L}_{\text{CE}}(f_i(\tilde{x}), y_i), \quad (3)$$

where $\bar{\Upsilon} = \mathcal{N} \setminus \tilde{\mathcal{N}} \setminus \Upsilon$ and $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy loss, so $\mathcal{L}_M$ is the cross-entropy loss over the pixels which are misclassified according to y, and $\mathcal{L}_{\bar{M}}$ is over the remaining pixels (excluding those covered by the patch). This enables us to define an overall loss function $\mathcal{L}_p = \gamma \mathcal{L}_M + (1 - \gamma)\mathcal{L}_{\bar{M}}$. The value of γ is adaptively defined by $\gamma = \frac{|\Upsilon|}{|\mathcal{N} \setminus \tilde{\mathcal{N}}|}$, which increases the number of misclassified pixels when there are few, and improves their score when there are many.

To create an untargeted attack we set the target segmentation map y to be the ground truth segmentation map for x, and then aim to *maximise* $\mathcal{L}$. For an insertion attack targeting the class c we set y to be a segmentation map consisting entirely of class c, and minimise $\mathcal{L}$. For a deletion attack targeting c we redefine $\mathcal{N}$ to consist only of pixels which have ground truth class c. Then set y to be the ground truth, and aim to maximise $\mathcal{L}$. This means that the adversarial patch is aiming for pixels with ground truth class c to be misclassified, and all other pixels are left unconstrained.

3.2. Patch Generation and Optimisation

Since a generator is only able to produce a limited selection of output sizes, and requires significant architectural changes to

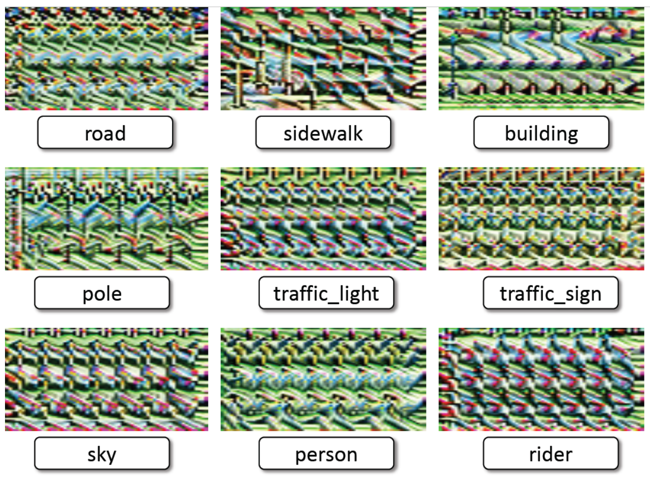


Fig. 3. Visualisations of adversarial patches sampled from our conditional generator G trained to attack an undefended DDRNet [8], targeting each evaluation class of the Cityscapes dataset [23] in an insertion attack.

do so, we fix the size of patch output from the generator and scale it to the appropriate size for application.

An unconditional patch generator G takes as its input an $N \times 100$ noise vector z, where N is the batch size. In this paper we explore networks of 5-7 up-convolutional layers, which produce patches of size 64×64, 128×128, or 256×256 respectively. We also consider rectangular variants producing patches of size 64×128, 128×256, and 256×512. Each layer uses 4×4 up-convolutional filters, except for rectangular variants which have 4×8 filters in the first layer. We use leaky-ReLu activation and batch normalisation after each layer. To make the generator conditional we add an $N \times 19$ class vector input c. We refer to the attack using the unconditional generator as A-UGen, and the conditional A-CGen. In the Supplemental Materials we explore the impact of generator architecture on attack and defence performance. A-UGen and A-CGen are both trained for 20 epochs using an ADAM optimizer with a learning rate of 10^{-4}.

Figure 3 shows the output of a conditional generator G after it is trained to perform an insertion attack an undefended DDRNet. Unlike adversarial patches for classification, the patches do not obviously contain visual features from the objects they are targeting, but the patches for different classes are visually distinct. The distribution of colours skews heavily towards bright and vibrant.

3.3. Discriminator and Training Methodology

The loss function for training f is defined as follows:

$$\mathcal{L}_f = \mathcal{L}_{\mathrm{B}}(x, y) + \mathcal{L}_{\mathrm{B}}(A(G(z, c), x, l, t), y), \tag{4}$$

where y is the ground truth. The first term tries to ensure that clean images are correctly classified and the second is aiming for images with adversarial patches to be correctly classified. $\mathcal{L}_{\mathrm{B}}$ is a pixel-wise bootstrap cross entropy loss, where only the 10% of pixels with the highest loss are included. We also freeze all batch normalisation parameters of the network during training.

Our network is trained in a similar manner to a Generative Adversarial Network (GAN). The segmentation network that we are defending takes the place of the discriminator in the GAN architecture. Instead of the discriminator acting to make the generator better, we are using the generator to produce a more effective discriminator. When training the networks we train alternately the discriminator and generator at each iteration, in the manner of a standard GAN.

The generator is pre-trained for 20 epochs prior to training the discriminator. After this the generator and discriminator are trained together for another 50 epochs to provide protection against adversarial patches. Both the generator and discriminator use Adam optimisers, the generator with a learning rate of 10^{-4} and the discriminator 10^{-5}.

4. RESULTS AND DISCUSSION

We evaluate the ability of our proposed SegGuard method to defend against both seen and unseen adversarial patch attacks (APAs), as well as its performance on clean images, compared with two baseline defences. Additionally we evaluate the attack performance of both conditional and unconditional patch generators, compared with the baseline of a traditional pixel-optimised patch attack.

Datasets. All models are trained and evaluated on the Cityscapes dataset [23]. We use the 2,975 finely labeled training images to train both our adverarial attacks and our defended segmentation networks. We evaluate on the 500 validation images, and we consider only the 19 classes which are labeled for evaluation.

Metrics. To measure segmentation performance we use metrics based on *intersection over union* (IoU). For a class c, a lower value of $\mathrm{IoU}(c)$ indicates better segmentaton performance on c. Hence when we measure the performance of *attacks* a *higher* IoU indicates better performance, but for *defences* a *lower* IoU is better. The usual way to capture the performance of a segmentation model is with the mean IoU, which is defined by $\mathrm{mIoU} = \sum_c \mathrm{IoU}(c)$. We use this metric to evaluate models without any adversarial attack, or with an untargeted attack. In the case of targeted attacks, we consider two metrics which encapsulate the performance of adversarial patches targeting each of the 19 evaluation classes into a single number. The first metric is the mean mIoU, which is defined by $\mathrm{mmIoU} = \sum_t \sum_c \mathrm{IoU}(c, t)$, where $\mathrm{IoU}(c, t)$ is the IoU of class c when each image has an adversarial patch targeting class t. The second metric is defined by $\mathrm{mcIoU} = \sum_t \mathrm{IoU}(t, t)$. This enables a focused look at whether the attacks are achieving their specific goals, since only the target class is considered for each patch. In all cases we compute the IoU only over those pixels which are not covered by the patch.

Baselines. We compare our attacks against the pixel-optimised patch method of Nesti *et al.*[7], which we refer to here as A-Patch. This also serves as an unseen attack for the evaluation of defended models. We use patches of size 300×600. We compare our defence against two baselines. The first (D-LGS) is the local gradients smoothing method of Hayes, with the only adjustment being to scale the image to compute the areas to smooth based on a 224×224 downsampled image, which is then upsampled to apply the smoothing. The second baseline (D-Patch) is inspired by the Ally Patch method of Abdel-Hakim *et al.*[21]. We split the image in to 2×2 non-overlapping patches, rescale each of them to 1024×2048, apply the segmentation network to each patch, then resize each output and combine to create the full segmentation patch.

Implementation Details. All experiments were completed using the open-source PyTorch library. The implementation of the segmentation networks is from `pytorch-semseg`, using weights pre-trained on cityscapes from the original authors.

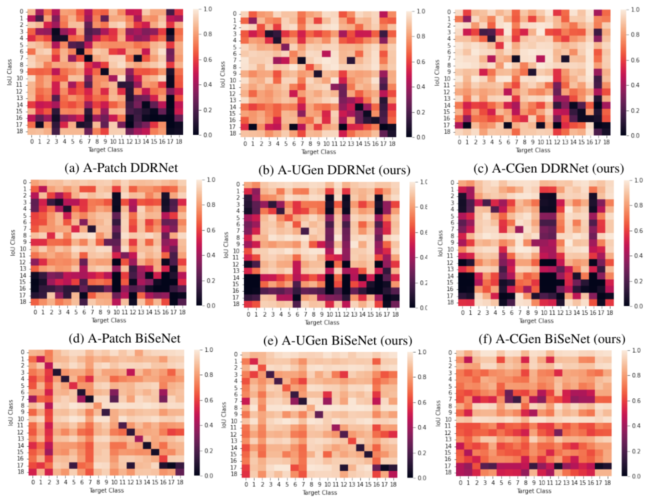
4.1. Pixel-Optimised Patch vs Generator Attack

Untargeted Attack. The left half of Table 1 compares the performance of untargeted patches produced by the baseline method A-Patch with those produced by our unconditional patch generator (A-UGen), as well as a patch consisting of random noise as a baseline (A-Noise). On DDRNet and BiSeNet the best results are achieved by A-UGen. These attacks decrease the performance of the model by over 40 percentage points when compared to A-Noise. For these networks A-Patch, on the other hand, reduces the performance by less than 30 percentage points. For ICNet the performance of A-Patch is better than A-UGen, reducing the performance by over 40 percentage points.

Targeted Attack. The right half of Table 1 compares the overall performance of targeted insertion attacks. The attacks we consider are A-Patch, A-UGen and A-CGen. For A-Patch and A-UGen we train a separate patch/generator targeting each of the 19 evaluation classes. For A-CGen we train a single generator conditioned on the target class. For all three networks the best performing attack on both metrics is A-Patch. A-UGen and A-CGen both reduce mmIoU and mcIoU

Arch.	Untargeted A-Noise	Untargeted A-Patch	Untargeted A-UGen	Metric	Targeted A-Patch	Targeted A-UGen	Targeted A-CGen
DDRNet	0.76	0.53	**0.26**	mcIoU	**0.16**	0.27	0.31
				mmIoU	**0.46**	0.58	0.59
BiSeNet	0.65	0.42	**0.16**	mcIoU	**0.24**	0.28	0.32
				mmIoU	**0.44**	0.45	0.45
ICNet	0.73	**0.30**	0.49	mcIoU	**0.28**	0.35	0.37
				mmIoU	**0.63**	0.66	0.69

Table 1. Average performance of Untargeted and Targeted Attacks with three kinds of optimisation: a pixel optimised-patch (A-Patch), unconditional generator (A-UGen) , conditional generator (A-CGen), and random noise (A-Noise)



(a) A-Patch DDRNet (b) A-UGen DDRNet (ours) (c) A-CGen DDRNet (ours)

(d) A-Patch BiSeNet (e) A-UGen BiSeNet (ours) (f) A-CGen BiSeNet (ours)

(g) A-Patch ICNet (h) A-UGen ICNet (ours) (i) A-CGen ICNet (ours)

Fig. 4. Comparison of targeted insertion attacks, broken down by target class (zoom for detail). Each column corresponds to a target class, and then each row is the IoU for a class when the network is attacked with that patch. A value of 1 (white) means that the patch did not change the performance on that class, and 0 (black) means the IoU was 0 for that class.

significantly for all three networks, meaning they are useful attacks in practice and likely suitable for use in SegGuard to finetune a segmentation model. For a breakdown by class, as well as a study of deletion attacks see the sup. mat.

In Figure 4 we break down the performance of targeted insertion attacks by class. Each column represents a target class and then each row contains the relative IoU for a particular target class when the model is attacked by a patch of that target class. In this context *relative IoU* means that for each class instead of reporting the IoU for that class directly, we report it as a proportion of the IoU of that class in the original model with no adversarial patch attacks applied. This makes the impact of the patch attack on different classes clear even when the classes had different IoUs initially. For DDRNet and BiSeNet the class performance of A-UGen and A-CGen patches is very similar. Targeting the same class results in very similar IoUs across every class from both of these attacks. This shows that the conditioning is respected and the generator is effective at producing patches. This is useful because it allows the creation of a patch targeting any of 19 classes on demand from a single generator, and it also enables the defence which is explored in a following section. On ICNet the diagonal is not as strong for A-CGen as for A-UGen, showing that the conditioning is not as effective, but away from the diagonal A-CGen is more effective. We can also see that for all three networks the A-Patch attack performs slightly better.

4.2. Defences

We examine the robustness of four defended networks, trained with either a conditional insertion attack generator (D-CGen) or an unconditional generator (D-UGen), and with the generator being either 7 layers and rectangular (L) or 5 layers and square (S). All attacks are fully white box, *i.e.*the attacker has access to the weights of the defended networks in the SegGuard case, and knows the additional preprocessing that is used for the baseline defences. Since D-Patch is fully differentiable it is straightforward to attack in this setting. D-LGS is not differentiable so we bypass the defence using BPDA [24].

Table 2 shows the performance of each method when faced with targeted attacks, untargeted attacks, as well as no attack, and a patch consisting of random noise (A-Noise). On the images with no patch, our methods always achieve within 2% of the Undefended performance. In several cases our networks actually meet or exceed the performance of the undefended network. The baseline methods both always perform worse on clean images that our defended networks and the undefended network. For A-Noise the story is similar, with our networks performing similarly to the undefended network (or better in some cases), and the baseline defences losing some performance. When faced with A-UGen, the undefended method drops significantly. The performance of the baseline defences is in most cases not much better. Our defence can recover the performance to be similar to images with no patch. For A-Patch, the drop in performance from the attack is sometimes not as large, but again the baseline defences fail to improve performance much. The best performing method are again ours. Against targeted A-Patch the best results for all three networks are achieved by D-CGen (L), which provides an almost complete defence. The baseline defences provide almost no benefit over the undefended network. Against A-CGen the best performance is achieved by D-CGen (S), which should not be surprising since this is the attack the network was trained on. However D-CGen (L) still performs well with this attack. Again the baseline defences provide minimal benefit.

4.3. Visualising Segmentation Results

Figure 5 shows representative segmentations. Our method can correctly in-fill behind the patch, unlike other defences. The two baseline defences produce lower accuracy segmentations with artifacts visible. The baselines are successful at defending against A-Patch, localising spurious class detections within the patch. However for A-UGen, neither of the baselines keep class ‘motorcycle’ within the confines of the patch, although they reduce its spread vs. the undefended network. Our defended network produces accurate segmentations outside of

Table 2. Mean intersection over union (mIoU) of defended networks tested with targeted and untargeted adversarial patches.

Arch.	Defence	Untargeted				Targeted			
		No Patch	A-Noise	A-Patch	A-UGen	A-CGen		A-Patch	
						mcIoU	mmIoU	mcIoU	mmIoU
DDRNet	Undefended	**0.78**	0.74	0.53	0.36	0.31	0.59	0.16	0.46
	D-UGen (S)	0.77	0.75	0.63	0.72	0.62	0.73	0.23	0.60
	D-UGen (L)	0.76	0.73	0.71	0.73	0.74	0.75	0.44	0.69
	D-CGen (S)	0.77	**0.76**	0.61	**0.75**	**0.76**	**0.77**	0.27	0.62
	D-CGen (L)	0.76	0.74	**0.74**	**0.75**	0.75	0.75	**0.74**	**0.74**
	D-Patch	0.73	0.71	0.50	0.55	0.30	0.61	0.18	0.55
	D-LGS	0.73	0.68	0.52	0.39	0.38	0.62	0.17	0.48
BiSeNet	Undefended	**0.69**	0.65	0.45	0.19	0.32	0.45	0.24	0.44
	D-UGen (S)	**0.69**	0.66	0.55	0.60	0.52	0.61	0.42	0.59
	D-UGen (L)	0.67	0.65	0.53	0.55	0.54	0.63	0.48	0.62
	D-CGen (S)	**0.69**	**0.67**	**0.63**	**0.66**	**0.67**	**0.68**	0.57	0.66
	D-CGen (L)	**0.69**	0.66	0.62	0.65	**0.67**	0.67	**0.66**	**0.67**
	D-Patch	0.62	0.58	0.42	0.40	0.30	0.46	0.20	0.48
	D-LGS	0.63	0.59	0.41	0.19	0.32	0.46	0.26	0.44
ICNet	Undefended	0.78	0.74	0.45	0.53	0.37	0.69	0.28	0.63
	D-UGen (S)	**0.79**	**0.77**	0.59	0.69	0.53	0.74	0.37	0.72
	D-UGen (L)	0.78	0.76	0.73	0.75	0.67	0.75	0.60	0.75
	D-CGen (S)	**0.79**	0.78	0.69	**0.77**	**0.78**	**0.78**	0.65	0.76
	D-CGen (L)	**0.79**	**0.77**	**0.77**	**0.77**	**0.78**	**0.78**	**0.77**	**0.77**
	D-Patch	0.76	0.70	0.57	0.64	0.44	0.69	0.21	0.63
	D-LGS	0.74	0.68	0.63	0.52	0.48	0.67	0.42	0.65

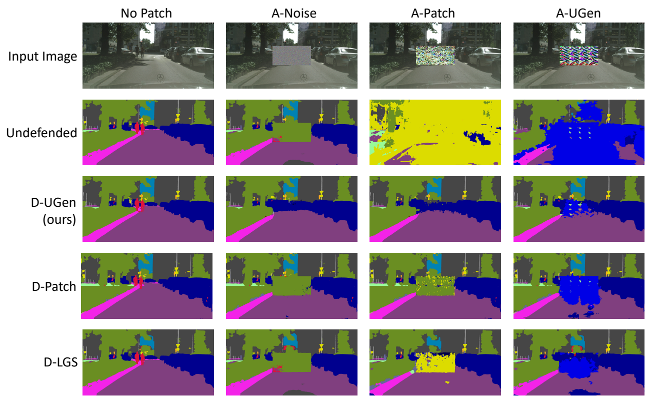


Fig. 5. Visualisation of the segmentation output from our SegGuard defended model vs an undefended model and baseline defences (zoom for detail). Our method learns to ignore the adversarial patches, and to inpaint the segmentation map for the area which was covered by the patch, without degrading the performance on clean images.

the patch area, and plausibly in-fills the patch.

Table 3 compares inference time vs. baselines. We process a single batch of images, averaged across the 500 Cityscapes validation images and 5 runs, with a batch size of 12. The SegGuard defended network has the same speed as the original network for all architectures, since it changes only the weights. A-Patch does not add a significant time to the inference for each batch, but requires batch size to be decreased by $4\times$, meaning that significantly more memory is required to achieve the same speed. A-LGS takes longer and is not feasible for real time segmentation.

Table 3. Inference time (ms) / frame rate (fps) for networks defended by SegGuard, vs. undefended and baseline defences

Arch.	Undefended	D-U/CGen	D-Patch	D-LGS
DDRNet	8.6/116	**8.5/118**	9.0/110	10^3/0.8
BiSeNet	**14.0/73**	**14.0/73**	14.0/70	1.2/0.8
ICNet	12.0/87	**11.0/88**	12.0/86	850.0/1.2

5. CONCLUSION

We proposed SegGuard[1], the first training-time defence against adversarial patch attacks for semantic segmentation networks. We showed that our method achieves better results than two baseline methods across multiple attacks, including those which are unseen during the adversarial training process. We achieve this while maintaining the performance of the network on the clean data and without any modification to the network architecture. Additionally our patch generator provides superior attack performance on undefended networks for some types of attack, as well as enabling the efficient production of adversarial patches of many classes via conditional generation. Future work will apply this method to different segmentation models and datasets, and explore different attack modalities such as the remote adversarial patch [25].

[1]**Supplementary material** for this paper is available at https://sigport.org/documents/segguard-defending-scene-segmentation-against-adversarial-patch-attack-supplementary

6. REFERENCES

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013. 1, 2

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[3] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582. 1

[4] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017. 1

[5] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. CVPR*, 2018. 1, 2

[6] Thomas Gittings, Steve Schneider, and John Collomosse, "Robust synthesis of adversarial visual examples using a deep image prior," in *Proc. BMVC*, 2019. 1, 2

[7] Federico Nesti, Giulio Rossolini, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo, "Evaluating the robustness of semantic segmentation for autonomous driving against real-world adversarial patch attacks," in *Proc. WACV*, 2022, pp. 2280–2289. 1, 2, 4

[8] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia, "Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes," *arXiv preprint arXiv:2101.06085*, 2021. 1, 2, 3

[9] 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)*, 2018, pp. 325–341. 1, 2

[10] 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)*, 2018, pp. 405–420. 1, 2

[11] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, "Adversarial examples in the physical world," in *Proc. ICLR Workshops*, 2017. 1

[12] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen, "Dpatch: An adversarial patch attack on object detectors," *arXiv preprint arXiv:1806.02299*, 2018. 2

[13] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," *Proc. Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018. 2

[14] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno, "Physical adversarial examples for object detectors," in *Proc. USENIX Workshop on Offensive Technologies*, 2018. 2

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, "Adversarial machine learning at scale," *Proc. ICLR*, 2017. 2

[16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symposium on Security and Privacy*, 2016. 2

[17] Dongyu Meng and Hao Chen, "MagNet," in *Proc. Comp. and Comm. Security*, 2017. 2

[18] Pouya Samangouei, Maya Kabkab, and Rama Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. ICLR*, 2018. 2

[19] Jamie Hayes, "On visible adversarial perturbations & digital watermarking," in *Proc. CVPR Workshops*, 2018. 2

[20] Muzammal Naseer, Salman Khan, and Fatih Porikli, "Local gradients smoothing: Defense against localized adversarial attacks," in *Proc. WACV*, 2019. 2

[21] Alaa E Abdel-Hakim, "Ally patches for spoliation of adversarial patches," *Journal of Big Data*, vol. 6, no. 1, pp. 1–14, 2019. 2, 4

[22] Thomas Gittings, Steve Schneider, and John Collomosse, "Vax-a-net: Training-time defence against adversarial patch attacks," in *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, 2016. 2, 3, 4

[24] Anish Athalye, Nicholas Carlini, and David Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 274–283. 5

[25] Yisroel Mirsky, "Ipatch: A remote adversarial patch," *arXiv preprint arXiv:2105.00113*, 2021. 6