

Contents of the Appendix

1. Broader Impact
2. Appendix A . . . Omitted proofs
3. Appendix B . . . Experimental and evaluation details
4. Appendix C . . . Additional SEA experiments, ablations and comparisons
5. Appendix D . . . Visualizations of adversarial images generated by SEA

Broader Impact

We propose new techniques to test the robustness of segmentation models to adversarial attacks. While we consider it important to estimate the vulnerability of existing systems, such methods might potentially be used by malicious actors. However, we also provide insights on how to obtain, at limited computational cost, models which are robust to such perturbations.

A Proof of the Properties of Cross-Entropy, and the Jensen-Shannon-Divergence loss

Cross-entropy loss:

The cross-entropy is given as: $\mathcal{L}_{\text{CE}}(\mathbf{p}, e_y) = -\log \mathbf{p}_y$, and has gradient

$$\frac{\partial \mathcal{L}_{\text{CE}}(\mathbf{u}, e_y)}{\partial u_t} = -\delta_{yt} + \mathbf{p}_t(u).$$

We note that

$$\|\nabla_u \mathcal{L}_{\text{CE}}(\mathbf{u}, e_y)\|_2^2 = \sum_{t \neq y} \mathbf{p}_t^2 + (1 - \mathbf{p}_y)^2.$$

As $0 \leq \mathbf{p}_t \leq 1$, it holds

$$\sum_{t \neq y} \mathbf{p}_t^2 \leq \sum_{t \neq y} \mathbf{p}_t = 1 - \mathbf{p}_y.$$

Moreover, the point of minimal ℓ_2 -distance on the surface of the ℓ_1 -ball with radius $1 - \mathbf{p}_y$ has equal components, and thus

$$\sum_{t \neq y} \mathbf{p}_t^2 \geq \frac{(1 - \mathbf{p}_y)^2}{K - 1},$$

which yields

$$\frac{K}{K-1}(1 - \mathbf{p}_y)^2 \leq \|\nabla_u \mathcal{L}_{\text{CE}}(\mathbf{u}, e_y)\|_2^2 \leq 1 - \mathbf{p}_y + (1 - \mathbf{p}_y)^2.$$

We note that both lower and upper bounds are monotonically increasing as $\mathbf{p}_y \rightarrow 0$.

Jensen-Shannon divergence:

The Jensen-Shannon-divergence between the predicted distribution p and the label distribution q is given by

$$D_{\text{JS}}(\mathbf{p} \parallel \mathbf{q}) = (D_{\text{KL}}(\mathbf{p} \parallel \mathbf{m}) + D_{\text{KL}}(\mathbf{q} \parallel \mathbf{m})) / 2, \quad \text{with } \mathbf{m} = (\mathbf{p} + \mathbf{q}) / 2,$$

Assuming that we have a one-hot label encoding $\mathbf{q} = e_y$ (where e_y is the y -th cartesian coordinate vector), one gets

$$D_{\text{JS}}(\mathbf{p} \parallel e_y) = \frac{1}{2} \log \left(\frac{2}{1 + \mathbf{p}_y} \right) + \frac{1}{2} \sum_{i=1}^K \mathbf{p}_i \log \left(\frac{2\mathbf{p}_i}{\delta_{yi} + \mathbf{p}_i} \right).$$

Then

$$\begin{aligned} \frac{\partial D_{\text{JS}}(\mathbf{p} \parallel e_y)}{\partial \mathbf{p}_r} &= \frac{1}{2} \left[-\frac{1}{1 + \mathbf{p}_y} \delta_{yr} + \log \left(\frac{2\mathbf{p}_r}{\delta_{yr} + \mathbf{p}_r} \right) + 1 - \frac{\mathbf{p}_r}{\delta_{yr} + \mathbf{p}_r} \right] \\ &= \frac{1}{2} \begin{cases} \log \left(\frac{2\mathbf{p}_y}{1 + \mathbf{p}_y} \right) & \text{if } r = y, \\ \log(2) & \text{else.} \end{cases} \end{aligned}$$

Given the logits u we use the softmax function

$$\mathbf{p}_r = \frac{e^{u_r}}{\sum_{t=1}^K e^{u_t}}, \quad r = 1, \dots, K,$$

to obtain the predicted probability distribution \mathbf{p} . One can compute:

$$\frac{\partial \mathbf{p}_r}{\partial u_t} = \delta_{rt} \mathbf{p}_t - \mathbf{p}_r \mathbf{p}_t \implies \sum_{r=1}^K \frac{\partial \mathbf{p}_r}{\partial u_t} = 0$$

Then

$$\begin{aligned} \frac{\partial D_{\text{JS}}(\mathbf{p} \parallel e_y)}{\partial u_t} &= \sum_{r=1}^K \frac{\partial D_{\text{JS}}(\mathbf{p} \parallel e_y)}{\partial \mathbf{p}_r} \frac{\partial \mathbf{p}_r}{\partial u_t} = \frac{1}{2} \left[\log \left(\frac{2\mathbf{p}_y}{1 + \mathbf{p}_y} \right) \frac{\partial \mathbf{p}_y}{\partial u_t} + \log(2) \sum_{r \neq y} \frac{\partial \mathbf{p}_r}{\partial u_t} \right] \\ &= \frac{1}{2} \left(\log \left(\frac{2\mathbf{p}_y}{1 + \mathbf{p}_y} \right) \frac{\partial \mathbf{p}_y}{\partial u_t} - \log(2) \frac{\partial \mathbf{p}_y}{\partial u_t} \right) \\ &= \frac{1}{2} \left(\log \left(\frac{\mathbf{p}_y}{1 + \mathbf{p}_y} \right) [\delta_{yt} \mathbf{p}_y - \mathbf{p}_y \mathbf{p}_t] \right) \\ &= \frac{1}{2} \left(\mathbf{p}_y \log \left(\frac{\mathbf{p}_y}{1 + \mathbf{p}_y} \right) [\delta_{yt} - \mathbf{p}_t] \right) \end{aligned}$$

Noting that $\lim_{x \rightarrow 0} x \log(x) = 0$ we get the result that: $\lim_{\mathbf{p}_y \rightarrow 0} \frac{\partial D_{\text{JS}}(\mathbf{p} \parallel e_y)}{\partial u_t} = 0$.

Thus the \mathcal{L}_{JS} loss automatically down-weights contributions from mis-classified pixels and thus pixels which are still correctly classified get a higher weight in the gradient.

Discussion. The (theoretical) discussion of benefits and weaknesses for each loss in Sec. 3.4 suggests that one main difference among losses is how they balance the weight of different pixels in the objective function. On one extreme, the plain cross-entropy maximizes the loss for all pixels independently of whether they are misclassified, and assigns them the same importance. Conversely, the masked losses exclude (via the mask) the misclassified pixels from the objective function, with the danger of reverting back the successful perturbations. As middle ground, losses like the JS divergence assign a weight to each pixel based on how “confidently” they are misclassified. We conjecture that for radii where robustness is low, masked losses help focusing on the remaining pixels, and already misclassified pixels are hardly reverted since they are far from the decision boundary. Conversely, at smaller radii achieving confident misclassification is harder (since the perturbations are smaller), and most pixels are still correctly classified or misclassified but close to the decision boundary: then it becomes more important to balance all of them in the loss, hence losses like JS divergence are more effective. This hypothesis is in line with the empirical results in Tab. 1 and Tab. 6.

B Experimental Details

We here provide additional details about both attacks and training scheme used in the experiments in the main part.

B.1 Attacks for semantic segmentation

Baselines. Since [1, 21] do not provide code for their methods, we re-implement both SegPGD and CosPGD following the indications in the respective papers and personal communication with the authors of CosPGD. In the comparison in Table 1, we use PGD with step size (8e-4, 9e-4, 1e-3, 2e-3, 3e-3, 5e-3, 6e-3) for radii (0.25/255, 0.5/255, 1/255, 2/255, 4/255, 8/255, 12/255) resp. for both CosPGD and SegPGD for 300 iterations each. The step size selection was done via a small grid-search in [2e-3, 3e-3, 5e-3, 6e-3, 1e-4] for $\epsilon = 4/255$ and $8/255$, the values for other radii were extrapolated from these. Moreover, at the end we select for each image the iterate with highest loss (strongest yet generated adversary).

APGD with masked losses. Since APGD relies on the progression of the objective function value to e.g. select the step size, using losses which mask the mis-classified pixels might be problematic, since the loss is not necessarily monotonic. Then, in practice we only apply the mask when computing the gradient at each iteration.

B.2 Training robust models

In the following, we detail the employed network architectures, as well as our training procedure for the utilized datasets. All experiments are conducted in

Table 4: Training and data configurations. For all the models trained in this work, we list according to the dataset, the training and dataset configurations. Warmup epochs are scaled depending on the total number of epochs. Poly dec. is the polynomially decaying schedule, from [49]. The setup stays the same across all setups of adversarial training (clean init./robust init. or 2 vs 5 step).

Configuration		PASCAL-VOC		ADE20K	
		PSPNet	UPerNet	UPerNet	Segmenter
DATA	Base size	512	512	520	520
	Crop size	473x473	473x473	512x512	512x512
	Random Horizontal Flip	✓	✓	✓	✓
	Random Gaussian Blur	✓	✓	✓	✓
TRAINING	Optimizer	SGD	AdamW	AdamW	SGD
	Base learning rate	5e-4	1e-3	1e-3	2e-3
	Weight decay	0.0	1e-2	1e-2	1e-2
	Batch size	16x8	16x8	16x8	16x8
	Epochs	50/300	50/300	32/128	32/128
	Warmup epochs	5/30	5/30	5/20	5/20
	Momentum	0.9	0.9, 0.999	0.9, 0.999	0.9
	LR schedule	poly dec.	poly dec.	poly dec.	poly dec.
	Warmup schedule	linear	linear	linear	linear
	Schedule power	0.9	1.0	1.0	0.9
	LR ratio (Enc:Dec)	1:10	✗	✗	✗
Auxiliary loss weight	0.4	0.4	0.4	–	

multi-GPU setting with PyTorch [34] library. For adversarial training we use PGD at $\epsilon = 4/255$ and step size 0.01. While training clean or adversarially, the backbones are initialized with publicly available IMAGENET pre-trained models, source of which are listed in Table 5.

Model architectures. Semantic segmentation model architectures have adapted to use image classifiers in their backbone. UPerNet coupled with ConvNeXt [29] and transformer models like ViT [17] with Segmenter [40] achieve SOTA segmentation results. We choose UPerNet and Segmenter architectures for our experiments with ConvNeXt and ViT as their respective backbones. For direct comparison to existing robust segmentation works [21, 48] which only train with a PSPNet [49], we also train a PSPNet with a ResNet-50 backbone (see Tables 2 and 6). Tab. 4 reports the training and data related information about the various architectures and the backbones used.

Table 5: Source of our pre-trained backbones. We employ the same backbone for both PASCAL-VOC and ADE20K. The robust column indicates if the backbone used is adversarially robust for IMAGENET and we also list the IMAGENET clean and robust accuracy at ℓ_∞ -radius of 4/255.

Architecture	Backbone	Robust	Source	IMAGENET acc.	
				clean	ℓ_∞
UPerNet	ConvNeXt-T + ConvStem	✗	[39]	80.9%	0.0%
UPerNet	ConvNeXt-T + ConvStem	✓	[39]	72.7%	49.5%
UPerNet	ConvNeXt-S + ConvStem	✓	[39]	74.1%	52.4%
Segmenter	ViT-S	✗	[43]	81.2%	0.0%
Segmenter	ViT-S	✓	[39]	69.2%	44.4%
PSPNet	ResNet-50	✓	[37]	64.0%	35.0%

UPerNet with ConvNeXt backbone. For both clean and robust initialization setups, we use the publically available IMAGENET-1k pre-trained weights⁴ from [39], which achieve SOTA robustness for ℓ_∞ -threat model at $\epsilon = 4/255$. They propose some architectural changes, notably replacing PatchStem with a ConvStem in their most robust ConvNeXt models, and we keep these changes intact in our UPerNet models, we always use a ConvNeXt with ConvStem in this work. We highlight that ConvNeXt-T, when adversarially trained for classification on IMAGENET, attains significantly higher robustness than ResNet-50 at a similar parameter and FLOPs count. For example, at $\epsilon_\infty = 4/255$, the ConvNeXt-T we use has 49.5% of robust accuracy, while ResNet-50 is reported to achieve around 35% [4, 37]. This supports choosing ConvNeXt as backbone for obtaining robust segmentation models with the UPerNet architecture. For UPerNet with the ConvNeXt backbone, we use the training setup from [29], listed in Tab. 4. We also use the same values of 0.4 or 0.3 for stochastic depth coefficient depending on the backbone, same as the original work.⁵ We do not use heavier augmentations and Layer-Decay [5] optimizer as done by [29].

Segmenter with ViT backbone. Testing with Segmenter also enables a further comparison across model size as Segmenter with a ViT-S backbone is less than half the size (26 million parameters) of UPerNet with a ConvNeXt-T backbone (60 million parameters). We define the training setup in Table 4, which is similar to the setup used by [40]. The decoder is a Mask transformer and is randomly initialized. Note that [40] predominantly use IMAGENET pre-trained classifiers at resolution of 384x384, whereas we use 224x224 resolution as no robust models at the higher resolution are available.

PSPNet with ResNet backbone. As prior works [21, 48] use a PSPNet with a ResNet [23] backbone to test their robustness evaluations, we also train the same model for the PASCAL-VOC dataset. Both DDCAT [48] and SegPGD-

⁴ <https://github.com/nmndeep/revisiting-at>

⁵ https://github.com/facebookresearch/ConvNeXt/blob/main/semantic_segmentation/configs/convnext

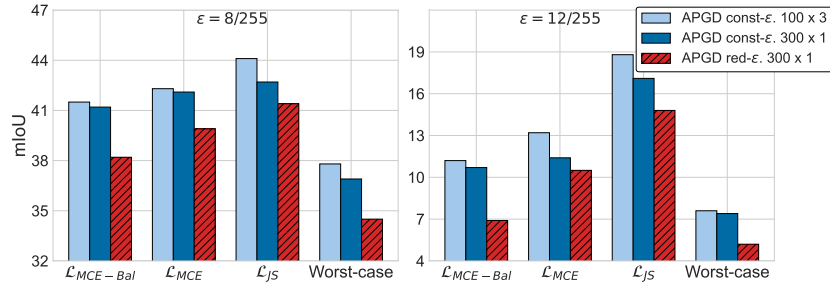


Fig. 4: Comparison of const- ϵ - and red- ϵ optimization schemes for mIoU. Balanced attack accuracy for the robust PIR-AT trained UPerNet + ConvNeXt-T model from Tab. 2 trained on PASCAL-VOC, across different losses for the same iteration budget. The radius reduction (red- ϵ) scheme performs best across all losses, and ϵ_∞ and even the worst-case over all losses improves.

AT [21] use a split of 50% clean and 50% adversarial inputs for training. Instead for PIR-AT with PSPNet, we just use adversarial inputs. Due to this change, and due to the fact that we initialize PIR-AT with IMAGENET pre-trained ResNet-50 (RN50), we slightly deviate from the standard training parameters (learning rate, weight decay, warmup epochs) as in the original PSPNet work [49]. The detailed training setup is listed in Tab. 4.

Training setup for PASCAL-VOC. We use the augmentation setup from [22]. Our training set comprises of 8498 images and we validate on the original PASCAL-VOC validation set of 1449 images. Data and training configurations are detailed in Tab. 4. Adversarial training is done with either 2 or 5 steps of PGD with the cross-entropy loss. Unlike some other works in literature, we train for 21 classes (including the background class).

Training setup for ADE20K. We use the full standard training and validation sets from [50]. Adversarial training is done with either 2 or 5 steps of PGD with the cross-entropy loss. Unlike the original work we train with 151 classes (including the background class).

B.3 Initialization with pre-trained backbones

PIR-AT uses pre-trained IMAGENET models as an initialization for the backbone. Note that in the semantic segmentation literature most modern works [29, 40] use clean IMAGENET pre-trained models as initialization for the backbone, making ours a natural choice. The robust models are sourced from [39] (see Tab. 5), and more are available e.g. in RobustBench [12], thus they do not cost us any additional pre-training. One can further reduce the cost of pre-training by using robust models trained for either 1-step [16] or 2-step [39] adversarial training, which is the common budget for robust IMAGENET training. For our UPerNet + ConvNeXt-S PIR-AT model for PASCAL-VOC, we use the 2-step 50 epoch IMAGENET trained model from [39] as initialization. Using such low-cost pre-

Table 6: Component analysis for SEA. We show the individual performance (Acc) of the runs of APGD (red- ϵ) with each loss in SEA for both PASCAL-VOC and ADE20K on 5-step robust models. The best results, among either individual runs, are in **bold**.

ϵ_∞	individual attacks						SEA	
	\mathcal{L}_{MCE}		$\mathcal{L}_{\text{MCE-Bal}}$		\mathcal{L}_{JS}			
model: PSPNet ResNet50, PIR-AT, 50 epochs, PASCAL-VOC								
4/255	83.3	48.6	84.7	49.9	81.8	47.8	81.5	47.7
8/255	53.4	13.5	56.4	12.2	53.7	14.1	50.6	11.2
12/255	14.9	2.3	17.6	1.7	20.7	4.1	12.9	1.4
model: UPerNet ConvNeXt-T, PIR-AT, 50 epochs, PASCAL-VOC								
4/255	89.2	65.9	90.4	67.4	88.7	64.9	88.6	64.9
8/255	74.0	40.6	77.5	38.4	73.9	41.3	71.7	34.6
12/255	31.5	10.3	36.9	6.7	38.6	15.1	28.1	5.5
model: UPerNet ConvNeXt-S, PIR-AT, 50 epochs, PASCAL-VOC								
4/255	89.7	67.5	90.9	68.9	89.3	66.7	89.1	66.0
8/255	73.6	41.0	77.5	36.9	74.3	42.7	71.0	36.4
12/255	31.2	10.7	36.9	7.5	39.0	15.6	27.6	6.2
model: UPerNet ConvNeXt-T, PIR-AT, 128 epochs, ADE20K								
4/255	56.8	20.0	58.2	17.9	55.9	18.9	55.5	17.2
8/255	28.5	6.6	31.1	5.3	28.5	7.2	26.4	4.9
12/255	3.7	0.9	4.5	0.9	5.2	1.1	3.1	0.4
model: UPerNet ConvNeXt-S, PIR-AT, 128 epochs, ADE20K								
4/255	58.6	20.4	59.8	18.6	57.6	19.4	56.8	17.9
8/255	31.3	8.1	33.3	5.8	30.9	7.7	28.7	5.4
12/255	4.6	1.1	5.4	0.8	6.2	1.3	3.1	0.6
model: Segmenter ViT-S, PIR-AT, 128 epochs, ADE20K								
4/255	56.9	17.8	57.6	15.6	55.6	16.6	55.3	14.9
8/255	36.2	8.5	37.8	5.6	34.2	7.7	33.3	5.4
12/255	10.5	2.2	11.7	1.3	11.2	2.2	8.9	1.1

trained backbones works well, as this model in Tab. 2 achieves better or similar robust accuracy as the 300 epoch 2-step IMAGENET pre-trained ConvNeXt-T in the same table.

C Additional Experiments and Discussion

We present additional studies of the properties of SEA and of the robust models.

C.1 Analysis of SEA

Effect of reducing the radius. We complement the comparison of const- ϵ and red- ϵ schemes provided in Sec. 3.6 by showing the different robust mIOU achieved by the various algorithms. In Fig. 4 one can observe that, consistently with what reported for average pixel accuracy in Fig. 2, reducing the value of ϵ (red- ϵ APGD) outperforms in all cases the other schemes.

Analysis of individual components in SEA. To assess how much each loss contributes to the final performance of SEA, we report the individual performance (both accuracy and mIOU) at different ϵ_∞ in Tab. 6, using robust models on PASCAL-VOC and ADE20K. We recall that each loss is optimized with 300 iterations of red- ϵ APGD. A common trend across all models is that either \mathcal{L}_{MCE} or \mathcal{L}_{JS} are best individual attacks for accuracy whereas $\mathcal{L}_{\text{MCE-BAL}}$ attacks the mIOU the best. Overall, SEA significantly reduces the worst case over individual attacks.

Analysing attack pairs in SEA. Further insights into SEA are given by looking at how different pairs of the components of SEA perform. Tab. 7 presents such evaluation for the robust UPerNet on PASCAL-VOC from Tab. 1: as expected, MCE + JS yields the best robust aAcc, while the pairs with MCE-Bal have the lowest mIOU. Moreover, the worst-case over all losses (SEA) gives further improvements.

More iterations. We also explore the effect of different number of iterations in SEA. In Fig. 5 we show the performance (measured by robust accuracy and mIOU) of SEA with 50, 100, 200, 300 and 500 iterations. There is a substantial improvement going from 50 to 300 iterations in all cases. On further increasing the number of attack iterations to 500, the drop in robust accuracy and mIOU is around 0.1% for both ℓ_∞ radii of 8/255 and 12/255. Since going beyond 300 iterations gives no or minimal improvement for significantly higher computational cost, we fix the number of iterations to 300 in SEA.

Effect of random seed. We study the impact of the randomness involved in our algorithm (via random starting points for each run) by repeating the evaluation on our robust model on PASCAL-VOC with 3 random seeds. Tab. 8 shows that the proposed SEA is very stable across all perturbation strengths. It

Table 7: Effectiveness of pairs of losses. We evaluate by pairing subset of components of SEA by measuring ACC and mIOU. Different pairs perform better or worse depending on perturbation strengths, while SEA always yields the strongest attack.

Loss pair	4/255		8/255		12/255	
$\mathcal{L}_{\text{MCE}} + \mathcal{L}_{\text{MCE-Bal}}$	88.8	65.1	73.2	35.1	31.6	5.6
$\mathcal{L}_{\text{MCE}} + \mathcal{L}_{\text{JS}}$	88.6	64.9	72.2	35.2	29.4	6.0
$\mathcal{L}_{\text{JS}} + \mathcal{L}_{\text{MCE-Bal}}$	88.8	64.9	73.0	34.7	32.6	5.6
SEA	88.6	64.9	71.7	34.6	28.1	5.5

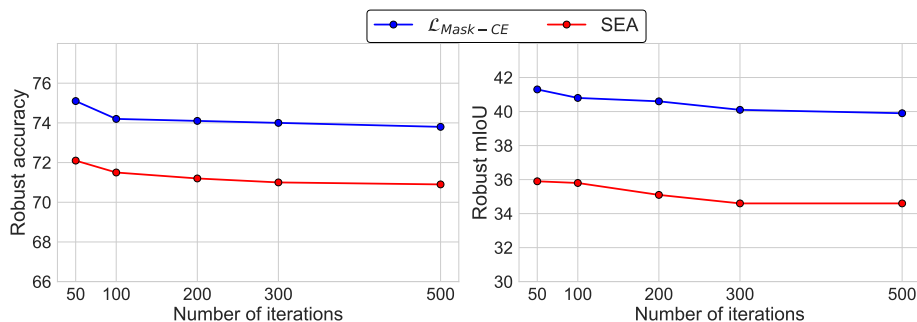


Fig. 5: Influence of number of iterations in SEA. We show robust average pixel accuracy (left) and mIoU (right) varying the number of iterations in our attack: 300 iterations give the best compute-effectiveness trade-off. We use the 5 step PIR-AT PASCAL-VOC trained ConvNeXt-T backbone UPerNet model and the attack is done for $\ell_\infty = 8/255$.

is also interesting to note that all individual losses have negligible variance across the different runs.

C.2 Excluding the background class from evaluation

For ADE20K, we train clean UPerNet + ConvNeXt-T models in two settings, i.e. either ignoring the background class (150 possible classes), which is the standard practice while training clean semantic segmentation models, or to predict it (151 classes). To measure the effect of the additional background class, we can evaluate the performance of both models with only 150 classes (for the one trained on 151 classes, we can exclude the score of the background class when computing the predictions). Training on 150 classes achieves (ACC, mIoU) of (80.4%, 43.8%), compared to (80.2%, 43.8%) for 151. This shows that we do not lose any performance when training with the background class, and the lower clean accuracy of clean trained ADE20K models, (ACC, mIoU) of (75.5%, 41.1%)

Table 8: Stability of SEA across different runs. We report ACC computed on PASCAL-VOC with the 5 step UPerNet model trained with PIR-AT. The mean across 3 runs is shown along with the standard deviation. Across components and perturbation strengths, SEA has a very low variance over random seeds.

ϵ_∞	\mathcal{L}_{MCE}	$\mathcal{L}_{MCE-BaI}$	\mathcal{L}_{JS}	SEA
model: UPerNet ConvNeXt-T, PIR-AT, 50 epochs				
4/255	89.2±0.2	65.8±0.3	90.4±0.1 69.0±0.2	88.7±0.1 64.9±0.4
8/255	73.8±0.4	40.8±0.4	77.5±0.2 38.1±0.2	73.9±0.1 41.3±0.0
12/255	31.5±0.3	10.2±0.2	36.9±0.2 6.6±0.1	38.6±0.4 15.0±0.1

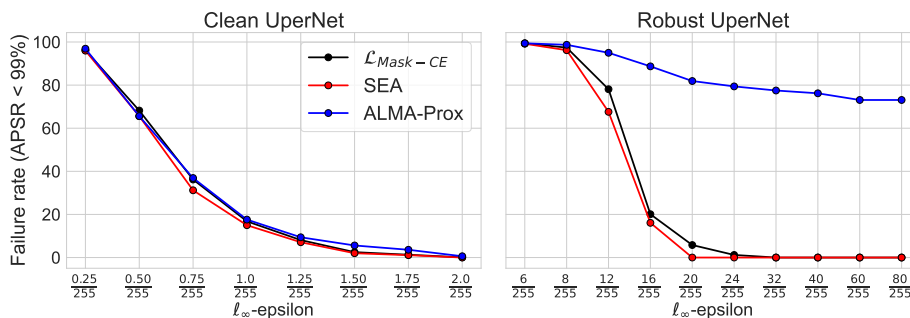


Fig. 6: Comparison to ALMA prox. We compare APGD with our novel loss ($\mathcal{L}_{Mask-CE}$) and the ensemble SEA according to the metric used by [36], which differs from those (ACC and mIoU) we use in the rest of our experiments. In the left plot, the attacks are tested on a clean trained model for the PASCAL-VOC dataset, and in the right plot we test against our robust PIR-AT model.

is due to including the background class when computing the statistics. This also translates to the robust models trained in the 2 step PIR-AT setting. For the robust model, the two settings have (76.6%, 37.8%) and (76.4%, 37.5%) (ACC, mIoU) respectively.

C.3 Additional comparisons to existing attacks

Rony et al. [36] have recently proposed ALMA prox as an adversarial attack against semantic segmentation models: its goal is to reach, for each image, a fixed success rate threshold (i.e. a certain percentage of mis-classified pixels, in practice 99% is used) with a perturbation of minimal ℓ_∞ norm. Thus, the threat model considered by [36] is not comparable to ours, which aims at reducing average pixel accuracy as much as possible with perturbations of a limited size.

In order to provide a comparison of our algorithms to ALMA prox, we measure the percentage of images for which the attack cannot make 99% of pixels be misclassified with perturbations of ℓ_∞ -norm smaller than a threshold ϵ (i.e. the model is considered robust on such images). In this case, lower values indicate stronger attacks. We show in Fig. 6 the results in such metric, at various ϵ , for ALMA prox (default values, 500 iterations), APGD on the Mask-CE loss (300 iterations) and SEA. We test for 160 random images from the PASCAL-VOC dataset using the clean trained UPerNet with a ConvNeXt-T backbone in the left plot and 5-step adversarially trained version of the same model in the right plot Fig. 6. For the clean model (left plot) the three attacks perform similarly, with a slight advantage of SEA at most radii. However, on the robust model (right plot), both APGD on the Mask-CE loss and SEA significantly outperform ALMA prox: for example, APGD, which uses even less iterations than ALMA prox, attains 0% robustness at 32/255, compared to 77% of ALMA prox. This

shows that, even considering a different threat model, our attacks are effective to estimate adversarial robustness.

C.4 Additional discussion of existing PGD-based attacks

Recently, [1, 21] revisited the loss used in the attack to improve the effectiveness of ℓ_∞ -bounded attacks, and are closest in spirit to our work. Since these methods represent the main baseline for our attacks, in the following we briefly summarize their approach to highlight the novelty of our proposed losses.

SegPGD: [21] proposes to balance the importance of the cross-entropy loss of correctly and wrongly classified pixels over iterations. In particular, at iteration $t = 1, \dots, T$, they use, with $\lambda(t) = (t - 1)/(2T)$,

$$\begin{aligned} \mathcal{L}_{\text{SegPGD}}(\mathbf{u}, y) = & ((1 - \lambda(t)) \cdot \mathbb{I}(\arg \max_{j=1, \dots, K} \mathbf{u}_j = y) \\ & + \lambda(t) \cdot \mathbb{I}(\arg \max_{j=1, \dots, K} \mathbf{u}_j \neq y)) \cdot \mathcal{L}_{\text{CE}}(\mathbf{u}, y). \end{aligned}$$

In this way the algorithm first focuses only on the correctly classified pixels and then progressively balances the attention given to the two subset of pixels: this has the goal of avoiding to make updates which find new misclassified pixels but leads to correct decisions for already misclassified pixels.

CosPGD: [1] proposes to weigh the importance of the pixels via cosine similarity between the prediction vector (after applying the sigmoid function $\sigma(t) = 1/(1 + e^{-t})$) and the one-hot encoding \mathbf{e}_y of the ground truth class. This can be written as

$$\mathcal{L}_{\text{CosPGD}}(\mathbf{u}, y) = \frac{\langle \sigma(\mathbf{u}), \mathbf{e}_y \rangle}{\|\sigma(\mathbf{u})\|_2 \|\mathbf{e}_y\|_2} \cdot \mathcal{L}_{\text{CE}}(\mathbf{u}, y) = \sigma(\mathbf{u}_y) / \|\sigma(\mathbf{u})\|_2 \cdot \mathcal{L}_{\text{CE}}(\mathbf{u}, y),$$

and again has the effect of reducing the importance of the pixels which are confidently misclassified.

C.5 Transfer attacks

To complement the evaluation of the robustness of our PIR-AT models, we further test them with transfer attacks from less robust models. In particular, we run APGD on the Masked-CE loss on Segmenter models obtained with either clean training or AT (5 steps) on ADE20K. We then transfer the found perturbations to our PIR-AT (5 steps, 128 epochs), and report robust accuracy and mIOU in Tab. 9, together with the results of the white-box SEA on the same model (from Tab. 2) as baseline. We observe that the transfer attacks are far from the performance of SEA, which further supports the robustness of the PIR-AT models.

Table 9: Transfer attacks. We show the robustness of PIR-AT to various transfer attacks (measured with ACC and mIOU at various radii). For each case we indicate the source and target models. Moreover, we report the evaluation given by white-box attacks as baseline.

Attack	Source	Target		0	4/255	8/255	12/255	
ADE20K, Segmenter with ViT-S backbone								
APGD w/ $\mathcal{L}_{\text{Mask-CE}}$	clean	PIR-AT		69.1	28.7	68.8 28.3	68.6 28.0	68.3 27.8
APGD w/ $\mathcal{L}_{\text{Mask-CE}}$	AT	PIR-AT		69.1	28.7	66.3 26.0	63.1 23.8	57.4 19.9
SEA (white-box)	PIR-AT	PIR-AT		69.1	28.7	55.3 14.9	33.3 5.4	8.9 1.1

D Additional Figures

Untargeted attacks. Fig. 7 shows examples of our untargeted attacks at different radii ϵ_∞ on the clean model for PASCAL-VOC dataset. In particular, we use 300 iterations of red- ϵ APGD on the $\mathcal{L}_{\text{Mask-CE}}$ loss. The first column presents the original image with the ground truth segmentation mask, The following columns contain the perturbed images and relative predicted segmentation masks for increasing radii ($\epsilon_\infty = 0$ is equivalent to the unperturbed image): one can observe that the model predictions progressively become farther away from the ground truth values. We additionally report the average pixel accuracy for each image. In Fig. 8, we repeat the same visualization for the most robust 5 step 300 epochs PIR-AT model. Note that we use different values of ϵ_∞ for the two models, i.e. significantly smaller ones for the clean model, following Tab. 1. Finally, the same setup is employed on the UPerNet + ConvNeXt-T model trained for ADE20K dataset for the illustrations in Fig. 9 (clean model) and Fig. 10 (5-step robust PIR-AT model), and we have similar observations as for the smaller dataset. Again we use smaller radii for the clean model, since it is significantly less robust than the PIR-AT one.

Targeted attacks. In Fig. 1 we show examples of the perturbed images and corresponding predictions resulting from targeted attacks. In this case, we run APGD (red- ϵ scheme with 300 iterations) on the negative JS divergence between the model predictions and the one-hot encoding of the target class. In this way the algorithm optimizes the adversarial perturbation to have all pixels classified in the target class (e.g. “grass” or “sky” in Fig. 1). We note that other losses like cross-entropy can be adapted to obtain a targeted version of SEA, and we leave the exploration of this aspect of our attacks to future work.

























original	0	0.25/255	0.5/255	1/255	2/255
	Acc: 95.9%	Acc: 94.8%	Acc: 75.9%	Acc: 48.3%	Acc: 0.0%
					
					
	Acc: 96.1%	Acc: 61.4%	Acc: 0.0%	Acc: 0.0%	Acc: 0.0%
					
					

Fig. 7: Visualizing the perturbed images, corresponding predicted masks and ACC for increasing radii. The attacks are generated on the clean model on PASCAL-VOC with APGD on $\mathcal{L}_{\text{Mask-CE}}$. Original image and ground truth mask in the first column.

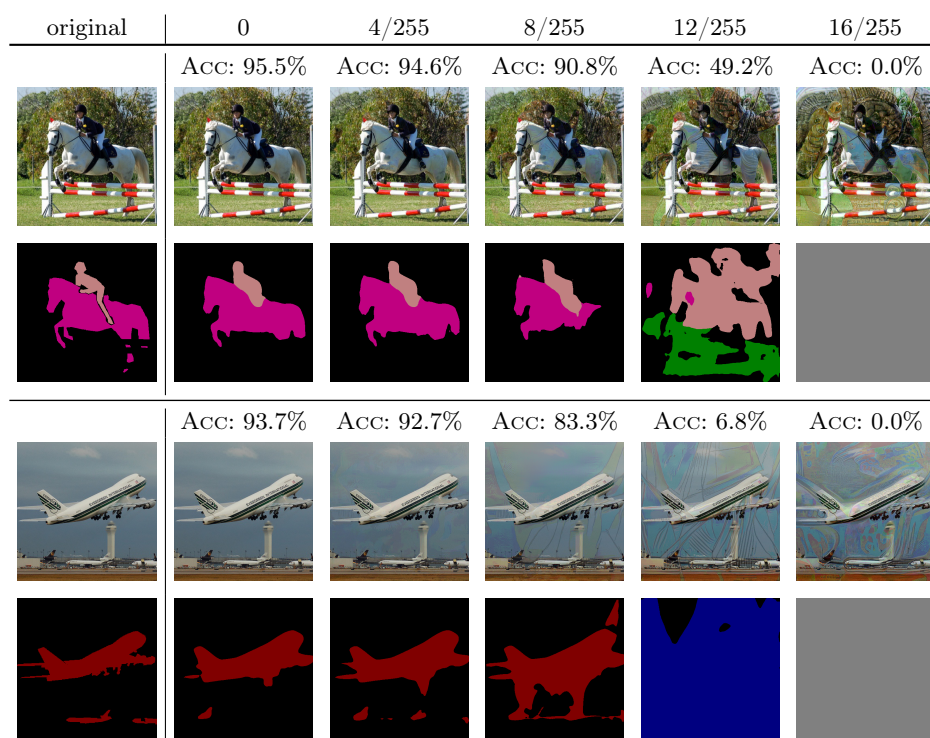


Fig. 8: Same setting as in Fig. 7 for the 5-step PIR-AT model







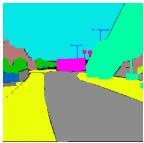

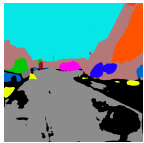
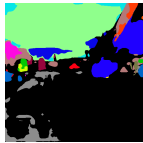


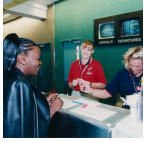
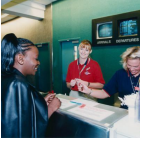
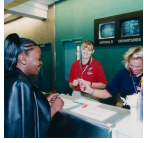
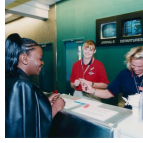
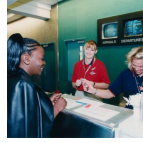
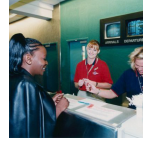






original	0	0.25/255	0.5/255	1/255	2/255
	Acc: 65.9%	Acc: 54.9%	Acc: 4.9%	Acc: 0.0%	Acc: 0.0%
					
					
	Acc: 81.2%	Acc: 47.9%	Acc: 21.9%	Acc: 2.6%	Acc: 0.0%
					
					

Fig. 9: Visualizing the perturbed images, corresponding predicted masks and Acc for increasing radii. The attacks are generated on the clean model on ADE20K with APGD on $\mathcal{L}_{\text{Mask-CE}}$. Original image and ground truth mask in the first column.

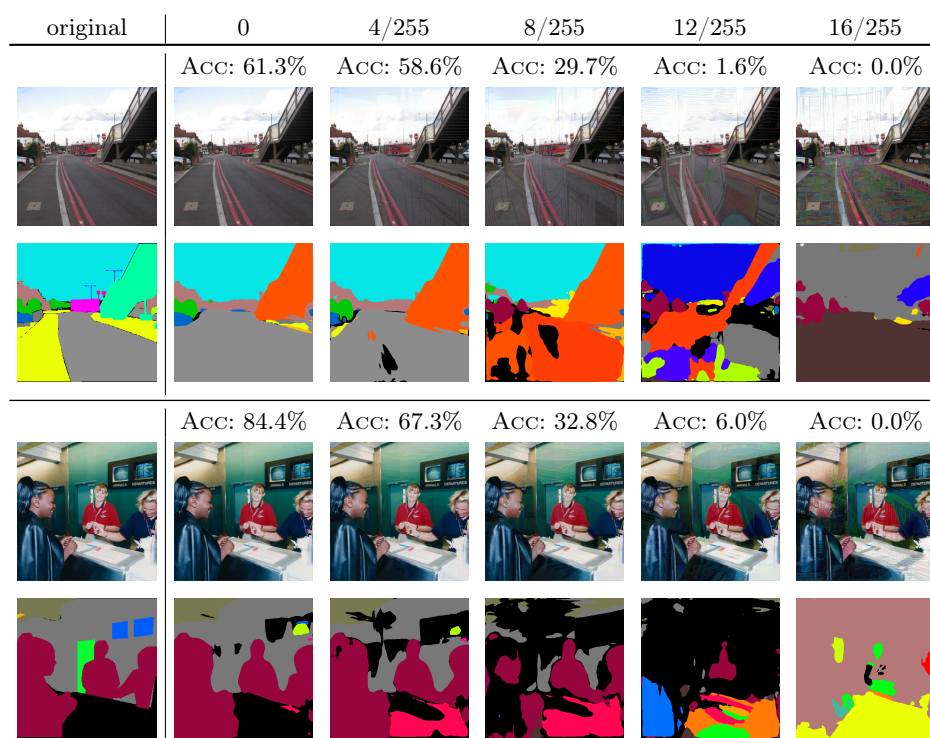


Fig. 10: Same setting as in Fig. 9 for the 5 step PIR-AT model.