Acknowledgements

The authors acknowledge support from the DFG Cluster of Excellence "Machine Learning – New Perspectives for Science", EXC 2064/1, project number 390727645. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting NDS. This research was also supported by the Center for AI Safety Compute Cluster. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

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 Property of the Jensen-Shannon-Divergence

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

$$D_{\mathrm{JS}}(\boldsymbol{p} \,\|\, \boldsymbol{q}) = \left(D_{\mathrm{KL}}(\boldsymbol{p} \,\|\, \boldsymbol{m}) + D_{\mathrm{KL}}(\boldsymbol{q} \,\|\, \boldsymbol{m})\right)/2, \quad \text{with} \quad \boldsymbol{m} = (\boldsymbol{p}+\boldsymbol{q})/2$$

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

$$D_{\mathrm{JS}}(\boldsymbol{p} \,\|\, e_y) = \frac{1}{2}\log\left(\frac{2}{1+\boldsymbol{p}_y}\right) + \frac{1}{2}\sum_{i=1}^{K} \boldsymbol{p}_i \log\left(\frac{2\boldsymbol{p}_i}{\delta_{yi}+\boldsymbol{p}_i}\right).$$

Then

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

Given the logits u we use the softmax function

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

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

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

Then

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

Noting that $\lim_{x\to 0} x\log(x) = 0$ we get the result that: $\lim_{\boldsymbol{p}_y\to 0} \frac{\partial D_{\mathrm{JS}}(\boldsymbol{p} \,\|\, e_y)}{\partial u_t} = 0$. We note that this is in contrast to the cross-entropy loss where, $\mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, e_y) = -\log \boldsymbol{p}_y$, and

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, e_y)}{\partial u_t} = -\delta_{yt} + \boldsymbol{p}_t$$

and thus $\lim_{\boldsymbol{p}_y\to 0} \frac{\partial \mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, e_y)}{\partial u_t} \neq 0$. In particular, this implies that when optimizing the sum of the cross-entropy loss over all pixels, even pixels which are already successfully attacked will still influence the gradient. In contrast, for the Jensen-Shannon-divergence the pixels which are already successfully attacked ($\boldsymbol{p}_y$ is small) do not contribute anymore significantly to the gradient and thus the attack can focus on the pixels which are not yet successfully attacked without the need of masking.

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 Gu et al. (2022); Agnihotri and Keuper (2023) 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 4, we use PGD with step size 0.01[1] for ℓ_∞-bound ϵ, and 100 iterations. Moreover, we select for each image the iterate with highest loss.

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 misclassified pixels might be problematic, since the loss is not necessarily

[1] For CosPGD, the authors suggested a step-size of 0.03, but we found 0.01 to yield a stronger attack.

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 multi-GPU setting with `PyTorch` library.

Model architectures. Semantic segmentation model architectures have adapted to use image classifiers in their backbone. Given UPerNet coupled with ConvNeXt (Liu et al., 2022) and/or transformer models like Swin (Liu et al., 2021) achieves art segmentation results, we use UPerNet as the base architecture for our experiments with ConvNeXt as the backbone. For both clean and robust initialization setups, we use the IMAGENET-1k pre-trained weights[2] from Singh et al. (2023). (Singh et al., 2023) achieve art robustness for ℓ_∞-threat model at $\epsilon = 4/255$ on the IMAGENET-1k dataset. 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 backbone models. 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% (Bai et al., 2021; Singh et al., 2023). This supports choosing ConvNeXt as backbone for obtaining robust segmentation models.

Training setup for PASCAL-VOC. For training on the PASCAL-VOC dataset, we use the augmentation setup from Hariharan et al. (2011). Our training set comprises of 8498 images and we validate on the original PASCAL-VOC validation set of 1449 images. For both training and testing the image is cropped from a base size of 512x512 to 473x473, we employ random horizontal flip and Gaussian Blur as the only two augmentations on top as done by (Zhao et al., 2017). We use the auxiliary head, with a loss coefficient of 0.4 in training the model. We train robust-initialized models only for 50 epochs whereas for clean initialized models we have other configurations as well. Adversarial training is done with either 2 or 5 steps of PGD on the cross-entropy loss. Throughout all experiments the base learning rate (LR) of 1e-3 with AdamW (Loshchilov and Hutter, 2019) optimizer and a weight decay factor of 1e-2 is used. We employ linear LR decay with a warm-up of 10 epochs for 50 epoch runs. Wherever we train for more than 50 epochs we linearly scale the warm-up epochs as well. For testing, we keep the same resolution resizing as for training without the augmentations and use the single-scale approach. Unlike most 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 Zhou et al. (2019). For both training and testing the image is resized to 520x520, then cropped to 512x512, keeping the same augmentations as for PASCAL-VOC. We train for 128 epochs (for clean and adversarial training) with a base LR of 1e-4 with AdamW optimizer and a weight decay factor of 5e-2, as used for the original ConvNeXt backbone UPerNet by Liu et al. (2022). We employ linear LR decay with a warm-up of 20 epochs and use stochastic depth coefficient of 0.4 or 0.3 depending on the backbone, same as the original work.[3] We do not use heavier augmentations and LayerDecay (Bao et al., 2021) optimizer as done by Liu et al. (2022). For the 32 epoch run in Table 3, we do a warm-up of 5 epochs, while the other parameters are the same as for 128 epochs. Unlike the original work we train with 151 classes (including the background class).

C. Additional Experiments

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

C.1. APGD for semantic segmentation attacks

Projected gradient descent (PGD) (Madry et al., 2018) uses the following iterative scheme for *minimizing* an objective function g in the set $S = \{z \in \mathbb{R}^d : \|z - x\|_\infty \leq \epsilon, z \in [0,1]^d\}$, i.e. an ℓ_∞-ball around a given point x intersected with the image box, denoting by P_S the projection onto S:

$$x^{(t+1)} = P_S\left(x^{(t)} - \alpha^{(t)}\,\mathrm{sign}(\nabla g(x^{(t)}))\right).$$

APGD has been introduced in Croce and Hein (2020) as an improved version of standard PGD which does not require parameter tuning e.g. selection of the step size $\alpha^{(t)}$. While it has been used in attacks for image classification, it can be more generally applied as optimizer for constrained problems. We show in Table 4 that simply replacing PGD with APGD for two previously proposed losses consistently improves their performance on a robust model from Sec. 4 (both PGD and APGD use 100 iterations, see App. B for details). The two attacks we consider are: *i)* SegPGD (Gu et al., 2022), an attack which optimizes $\mathcal{L}_{\text{Bal-CE}}$ with PGD, and *ii)* CosPGD (Agnihotri and Keuper, 2023), an attack which optimizes $\mathcal{L}_{\text{CosSim-CE}}$ with PGD. We observe that for both attacks and all values of the radius ϵ, APGD yields lower average pixel accuracy and mIOU (in grey), without imposing extra costs during optimization. Note that for high values of ϵ the improvements are quite large. Thus we use

[2] https://github.com/nmndeep/revisiting-at

[3] https://github.com/facebookresearch/ConvNeXt/blob/main/semantic_segmentation/configs/convnext

Table 4: **APGD consistently outperforms PGD** for two existing attacks, SegPGD (Gu et al., 2022) and CosPGD (Agnihotri and Keuper, 2023), on semantic segmentation. We report average pixel accuracy and mIoU after the adversarial attack, at various strengths of ϵ_∞.

ϵ_∞	$\mathcal{L}_{\text{Bal-CE}}$				$\mathcal{L}_{\text{CosSim-CE}}$			
	SegPGD		SegAPGD		CosPGD		CosAPGD	
	Acc	mIoU	Acc	mIoU	Acc	mIoU	Acc	mIoU
4/255	**88.6**	65.0	88.7	**64.8**	89.0	65.5	**88.9**	**65.4**
8/255	74.6	**39.4**	**74.2**	41.3	78.2	47.5	**77.8**	**47.3**
12/255	45.8	15.0	**43.3**	**14.9**	58.2	28.3	**56.2**	**26.4**
16/255	26.2	7.1	**20.7**	**5.7**	38.0	17.2	**34.0**	**15.3**

APGD for the optimization of all losses in this paper.

C.2. Analysis of SEA

Selection of losses. In Table 1 we have shown the performance of the attacks (APGD with 100 iterations) with six loss functions, and their worst-case. We then selected the best four of those ($\mathcal{L}_{\text{Mask-CE}}, \mathcal{L}_{\text{Bal-CE}}, \mathcal{L}_{\text{JS}}, \mathcal{L}_{\text{Mask-Sph.}}$) to be included in SEA. In Table 5 we additionally compare the robustness computed as worst-case of either all attacks (six runs, one per loss) or the four attacks with objective functions included in SEA. For both clean and robust models, using the two additional losses does not significantly improve performance of the ensemble attack, while increasing runtime. Moreover, we show below that each of the remaining four losses positively contributes to the results of SEA, which justifies the choice of including them.

Effect of reducing the radius. We complement the comparison of const-ϵ and red-ϵ schemes provided in Sec. 3.3 by showing the different robust mIoU achieved by the various algorithms. In Fig. 3 one can observe that, consistently with what reported for average accuracy in Fig. 2, reducing the value of ϵ (red-ϵ APGD) outperforms in most cases the other schemes.

Contribution of individual components in SEA. To assess how much each loss contributes to the final performance of SEA, we report the individual performance (as average pixel accuracy after attack) at different ϵ_∞ in Table 6, using robust models on PASCAL-VOC and ADE20K. We recall that each loss is optimized with 300 iterations of red-ϵ APGD. Additionally, we report the worst-case average pixel accuracy over subgroups of 3 out of 4 losses, and the worst-case of all 4, i.e. SEA. Although subset A works well for both datasets at $\epsilon_\infty \in \{4/255, 8/255\}$, it is not as effective as subsets B and C for higher ϵ_∞, and vice-versa. Interestingly, $\mathcal{L}_{\text{Mask-Sph.}}$ does not yield the best individual results in any case, but the excluding it from the worst-case computation (subset A) significantly degrades the performance. Hence, the four losses have complementary properties, as observed in Sec. 3.2, which allows us to have effective attacks across the entire range of ϵ_∞.

More iterations. We explore the effect of different number of iterations in SEA. In Fig. 4 we show the performance (measured by robust accuracy and mIoU) of SEA with 50, 100, 300 and 500 iterations. The performance at 0 iterations is the clean performance of this model. There is a substantial improvement on going from 50 to 300 iterations in all cases. On further increasing the number of attack iterations to 500, the drop in robust accuracy is around 0.1% for ℓ_∞ radius of 4/255 and none for 8/255. Also, there is no drop in mIoU for 4/255 whereas 8/255 sees a 0.1% decrease. Since going beyond 300 iterations gives no or minimal improvement for significantly higher computational cost, we fix the number of iterations to 300.

Effect of random seed. Table 7 shows that the proposed SEA is very stable across all perturbation strengths. It is also interesting to note that all individual loss have negligible variance across 5 different runs.

C.3. Larger backbone for robust models

We test in this section the effect of using ConvNeXt-S, which is nearly 1.7x larger in terms of number of parameters than ConvNeXt-T considered until now, as backbone in UPerNet. We again take the $\ell_\infty = 4/255$ robust ConvNeXt-S pre-trained on IMAGENET from (Singh et al., 2023) as initialization for our models. We note that this increases the size of the networks by only 1.4 times, since the backbone constitutes around 45% of it, while the FLOPs (in Giga) increase negligibly from 939 GFLOPs to 1027 GFLOPs. We keep the training setting consistent to ConvNeXt-T backbone models from the main text, see App. B.

In Table 8 we see that for PASCAL-VOC the clean performance ($\epsilon_\infty = 0$) of ConvNeXt-S backbone is better than ConvNeXt-T backbone models with the same training setup. The robustness for the model with AT^2 in mIoU is better by 2% on average, with smaller radii seeing larger improvements. For AT^5, the increase of robust mIoU at

Table 5: **Loss selection for SEA.** We compare taking the worst-case over all six losses from Table 1 APGD with 100 iterations versus using only the subset of the four losses part of SEA (average accuracy and mIoU are shown). The models are clean or adversarially trained for PASCAL-VOC. Leaving out two losses does not substantially impact the performance while considerably reduces the computational cost.

Subset	$\epsilon_\infty \rightarrow$	4/255		8/255		12/255		16/255	
model: clean, 50 epochs									
all losses		71.1	39.9	32.5	12.1	5.3	1.2	0.1	0.0
four losses		71.2	40.0	32.6	12.1	5.3	1.2	0.1	0.0
model: AT^5, robust init., 50 epochs									
all losses		88.3	64.4	72.3	38.4	31.9	8.4	6.4	1.1
four losses		88.3	64.6	72.3	38.4	31.9	9.0	6.4	1.2

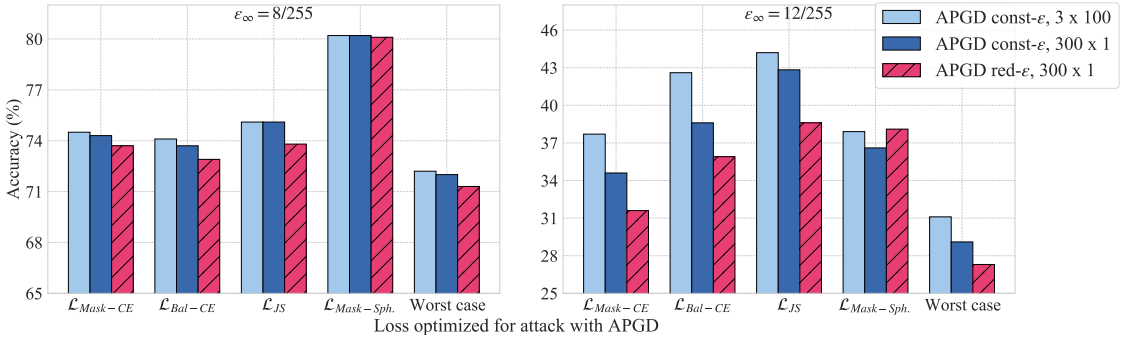

Figure 2: **Testing difference between const-ϵ and red-ϵ schemes.** Under the same iteration budget for the robust init. AT^5 model, across different losses we see radius reduction (red-ϵ) scheme leads to most drop in robust ACC. Specifically for worst-case over all losses, red-ϵ leads to the best attack across different perturbation strengths. The same trend holds for mIOU, see Fig. 3 in App. C.2

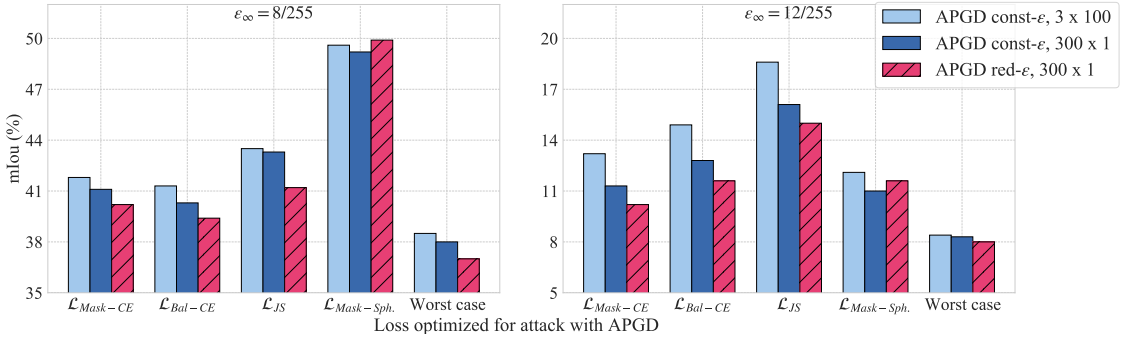

Figure 3: **Testing difference between const-ϵ and red-ϵ schemes.** Under the same iteration budget for the robust init. AT^5 model, across different losses we see red-ϵ scheme leads to most drop in robust mIOU. Specifically for worst-case over all losses, red-ϵ leads to the best attack across different perturbation strengths.

Table 6: **Component analysis for SEA.** We show the individual performance of the runs of APGD with each loss in SEA. Additionally we include the worst-case over subsets of 3 out of 4 losses. We report robust average accuracy for robust models on both the datasets. The best results, among either individual runs or subgroups, are **highlighted**.

A: $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{Bal-CE}} + \mathcal{L}_{\text{JS}}$ **B:** $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{Bal-CE}} + \mathcal{L}_{\text{Mask-Sph.}}$ **C:** $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{JS}} + \mathcal{L}_{\text{Mask-Sph.}}$

ϵ_∞	individual losses				subsets of three losses			all
	$\mathcal{L}_{\text{Mask-CE}}$	$\mathcal{L}_{\text{Bal-CE}}$	$\mathcal{L}_{\text{JS}}$	$\mathcal{L}_{\text{Mask-Sph.}}$	A	B	C	(SEA)
model: AT[5], robust init., 50 epochs, PASCAL-VOC								
4/255	89.0	88.5	**88.4**	90.5	**88.3**	88.5	88.4	88.3
8/255	73.7	**72.9**	73.8	80.7	**71.6**	71.7	71.8	71.3
12/255	**31.6**	35.9	38.6	38.1	29.4	**27.4**	27.8	27.3
16/255	**6.7**	11.9	12.5	6.8	5.8	4.3	**4.3**	4.2
model: AT[5], robust init., 50 epochs, ADE20K								
4/255	57.1	56.0	**55.9**	63.0	**55.6**	55.9	55.7	55.6
8/255	**28.6**	**28.6**	28.7	39.6	**26.5**	27.1	26.7	26.4
12/255	**4.2**	4.4	4.5	4.3	3.8	**3.5**	**3.5**	3.3

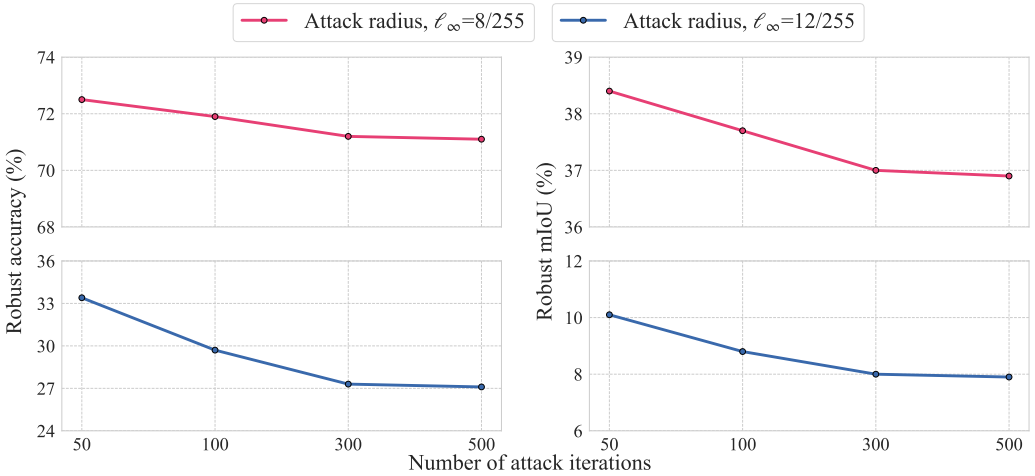

Figure 4: **Influence of number of iterations in SEA.** We show robust average accuracy (left) and mIoU (right) varying the number of iterations in our attack: 300 iterations give the best compute-effectiveness trade-off. We use the AT[5] ConvNeXt-T backbone model with robust initialization for PASCAL-VOC.

Table 7: **Stability of SEA across different runs.** We report ACC computed on PASCAL-VOC with the AT^5 model. The mean across 5 runs is shown along with the standard deviation. Across all its components and perturbation strengths, SEA has a very low variance over random seeds.

A: $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{Bal-CE}} + \mathcal{L}_{\text{JS}}$ **B:** $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{Bal-CE}} + \mathcal{L}_{\text{Mask-Sph.}}$ **C:** $\mathcal{L}_{\text{Mask-CE}} + \mathcal{L}_{\text{JS}} + \mathcal{L}_{\text{Mask-Sph.}}$

ϵ_∞	individual losses				subsets of three losses			all
	$\mathcal{L}_{\text{Mask-CE}}$	$\mathcal{L}_{\text{Bal-CE}}$	$\mathcal{L}_{\text{JS}}$	$\mathcal{L}_{\text{Mask-Sph.}}$	A	B	C	(SEA)
model: AT^5, robust init., 50 epochs, PASCAL-VOC								
4/255	89.0 ± 0.1	88.5 ± 0.1	**88.4** ± 0.1	90.5 ± 0.0	**88.4** ± 0.1	88.5 ± 0.0	88.4 ± 0.0	88.3 ± 0.0
8/255	73.6 ± 0.2	**73.0** ± 0.3	73.7 ± 0.1	80.6 ± 0.2	**71.5** ± 0.1	71.7 ± 0.0	71.7 ± 0.1	71.2 ± 0.1
12/255	**31.4** ± 0.4	35.9 ± 0.2	38.2 ± 0.4	38.1 ± 0.1	29.3 ± 0.2	**27.6** ± 0.2	27.9 ± 0.1	27.3 ± 0.1
16/255	**6.6** ± 0.1	11.9 ± 0.3	12.3 ± 0.2	6.9 ± 0.1	5.7 ± 0.1	4.3 ± 0.1	**4.3** ± 0.1	4.2 ± 0.1

Table 8: **Comparison of training schemes for PASCAL-VOC and ADE20K with larger models.** In continuation of Table 2 and Table 3, we show the effect of using a larger (robust) backbone, i.e. ConvNeXt-S, as initialization. The improvement in robustness (accuracy and mIOU) over ConvNeXt-T backbone for PASCAL-VOC is marginal for small perturbations, whereas significant benefit can be seen for ADE20K across all perturbation strengths.

Training scheme			0		4/255		8/255		12/255		16/255	
PASCAL-VOC												
ConvNeXt-T backbone												
AT^2	robust init.	50 ep.	92.9	75.9	86.7	60.8	50.2	21.0	9.3	2.4	0.8	0.3
AT^5	robust init.	50 ep.	92.7	75.2	88.3	63.8	71.2	37.0	27.4	8.1	4.2	0.9
ConvNeXt-S backbone												
AT^2	robust init.	50 ep.	93.4	77.2	87.8	63.2	53.5	23.0	10.3	2.7	0.9	0.4
AT^5	robust init.	50 ep.	93.1	76.6	89.2	66.2	70.8	38.0	27.0	8.6	3.9	1.0
ADE20K												
ConvNeXt-T backbone												
AT^2	clean init.	128 ep.	73.4	36.4	0.2	0.0	0.0	0.0	0.0	0.0	–	–
AT^2	robust init.	128 ep.	72.0	34.7	46.0	15.4	6.0	1.8	0.0	0.0	–	–
AT^5	robust init.	128 ep.	70.5	31.7	55.6	18.6	26.4	6.7	3.3	0.8	–	–
ConvNeXt-S backbone												
AT^5	robust init.	128 ep.	71.3	32.1	57.2	19.2	28.8	7.2	3.9	0.9	–	–

$\epsilon_\infty = 4/255$ is 3%, but for higher radii the improvement is only marginal if at all any. We hypothesize that the improvement could be much more if one increases the number of training epochs for these bigger models, as higher capacity needs more time to approach a better (robust) solution.

For ADE20K, clean performance gets slightly higher with the larger architectures. The improvement in robustness is around 2% (mIOU) for both $\epsilon_\infty = 4/255$ and $\epsilon_\infty = 8/255$, and around 0.6% for the largest radius. These results are consistent to what the original work introducing ConvNeXt showed when transitioning from ConvNeXt-T to ConvNeXt-S backbone. In fact, Liu et al. (2022) had a gain of 2.7% in clean mIOU which here translates to a slightly smaller improvement given we do adversarial training on top. In a similar vein, even large backbones could be tried and the increase in robustness would be tantamount to the increase from ConvNeXt-T to ConvNeXt-S, and we leave this to future work.

C.4. Excluding the background class from evaluation

For ADE20K, we train clean 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 apparent lower results reported e.g. in Table 3, (ACC, mIOU) of (75.5%, 41.1%) are due to including the background class when computing the statistics. This also translates to the robust models trained in the AT^2 setting. For the robust model, the two settings have (76.6%, 37.8%) and (76.4%, 37.5%) (ACC, mIOU) respectively.

D. Related Work

Adversarial attacks for semantic segmentation. ℓ_∞-bounded attacks on segmentation models have been first proposed by Hendrik Metzen et al. (2017), which focus on targeted (universal) attacks, and Arnab et al. (2018), using FGSM (Goodfellow et al., 2015) or PGD on the cross-entropy loss. Recently, Gu et al. (2022); Agnihotri and Keuper (2023) revisited the objective function used by PGD to improve the effectiveness of ℓ_∞-bounded attacks, and are closest in spirit to our work. Additionally, there exist a few works presenting algorithms for other threat models, including unconstrained, universal and patch attacks (Xie et al., 2017; Cisse et al., 2017; Mopuri et al., 2018; Shen et al., 2019; Kang et al., 2020; Nesti et al., 2022).

Robust segmentation models. As mentioned above, limited attention has been paid to developing defenses for segmentation models. First, Xiao et al. (2018a) propose a method to detect attacks, while stating adversarial training is hard to adapt for semantic segmentation. Later, DDC-AT (Xu et al., 2021) attempts to integrate adversarial points during training exploiting additional branches to the networks, but such models are shown to be not robust by Gu et al. (2022), who also propose to use their version of PGD, named SegPGD, for adversarial training. Finally, Cho et al. (2020); Kapoor et al. (2021) present defenses based on denoising, with either Autoencoders or Wiener filters, the input to remove the adversarial perturbations before feeding it to clean models. However these methods do not provide standalone robust models. Moreover, they are tested only via attacks with limited budget, while similar techniques for protecting image classifiers have been shown ineffective once evaluated with adaptive attacks (Athalye et al., 2018; Tramèr et al., 2020).

E. Threat model, metrics

Threat model. In this work we focus on the ℓ_∞-threat model, which means that every pixel of an input image can be modified independently. It is common practice for semantic segmentation tasks to exclude the pixels belonging to the background class, which means that those are not accounted for when minimizing the training loss or computing the test performance of a model. However, it is unrealistic that an attacker only modifies non-background pixels and thus we want models which are robust for all pixels independently of how they are classified or what their ground-truth label is. Thus, we train all our models with an additional background class. Note that this does not significantly influence the clean segmentation performance, but allows us to use a realistic definition of adversarial robustness for semantic segmentation, see App. C.4.

Metrics. For image classification the 0-1 loss is the established metric for measuring the performance of a model. Then an attacker has the clear goal of maximizing it, equivalent to reducing the classification accuracy of the model, and many techniques have been developed for this purpose (Carlini and Wagner, 2017; Madry et al., 2018). In the case of semantic segmentation, while it is possible to measure the classification accuracy averaged over pixels (ACC), it is common to use other performance metrics like Intersection over Union (IoU), averaged over classes (mIOU).

Differentiable losses which approximate mIOU exist, e.g. the Dice loss (Sudre et al., 2017): however, mIOU is typically computed across all the images of the test set. This means that an attacker should optimize the perturbations

over the entire set, which is not practically possible (mini-batches are often used). Conversely, the classification of pixels of an image is independent of other images, and so are the perturbations. Moreover, if an attacker is able to get all pixels to be misclassified, then the mIoU is trivially 0%. For this reason we use average accuracy as main metric for developing our attacks, but we always report mIoU as well. When reporting the worst-case over multiple runs, we select for each image the perturbation which yields the lowest average accuracy, and use it to compute the mIoU.

F. Additional Figures

Untargeted attacks. Fig. 5 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. 6, we repeat the same visualization for AT^5 model with robust initialization. Note that we use different values of ϵ_∞ for the two models, i.e. significantly smaller ones for the clean model, following Table 1. Finally, the same setup is employed on the ADE20K dataset for the illustrations in Fig. 7 (clean model) and Fig. 8 (robust 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 AT^5 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) to 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 for future work.

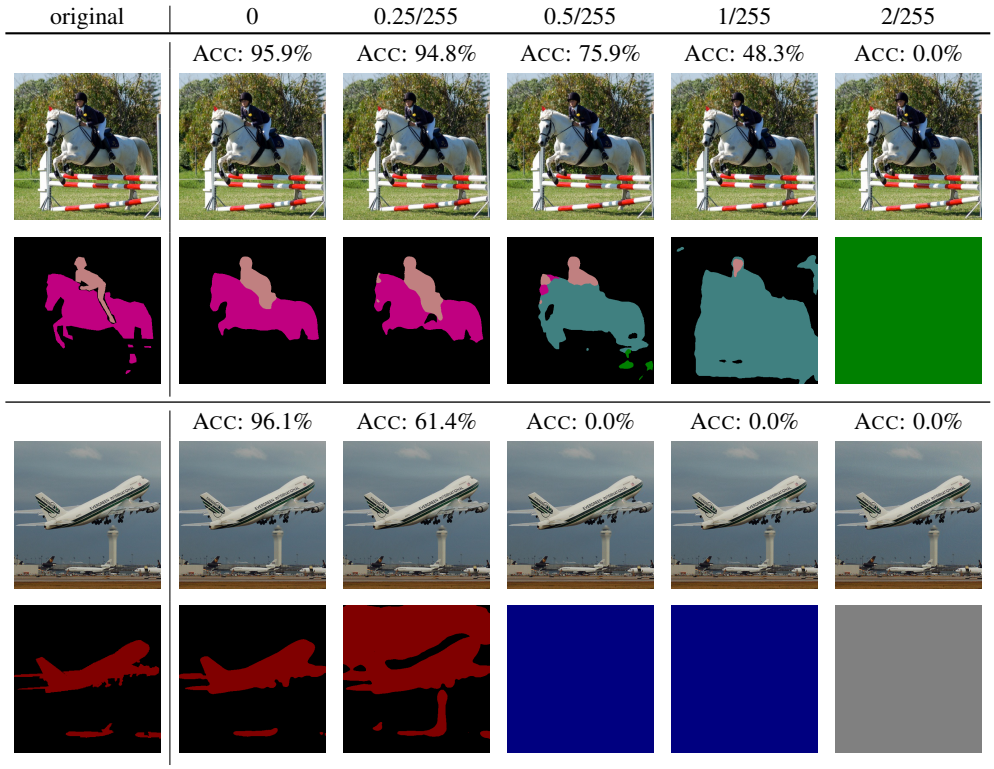

Figure 5: We show the perturbed images, corresponding predicted segmentation masks and average accuracy for increasing radii. The attacks are generated on the clean model on PASCAL-VOC with APGD on $\mathcal{L}_{\text{Mask-CE}}$. We additionally present (first column) the original image and ground truth mask.

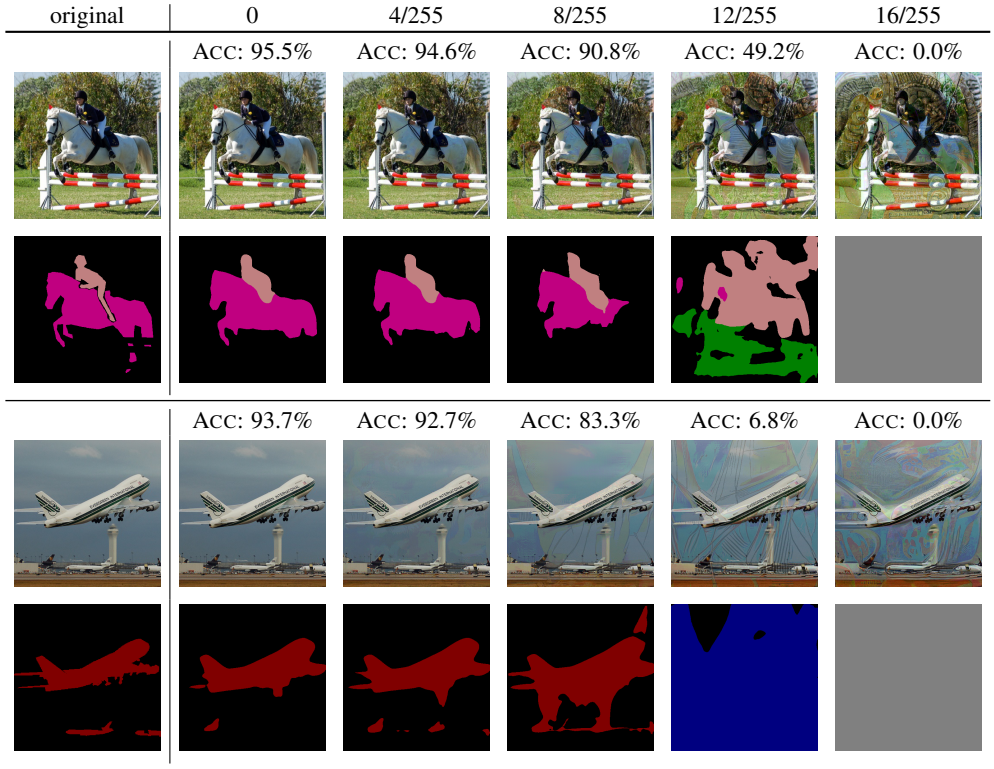

Figure 6: Same setting as in Fig. 5 for the robust AT^5 model. Note the larger radii ϵ for the attack.

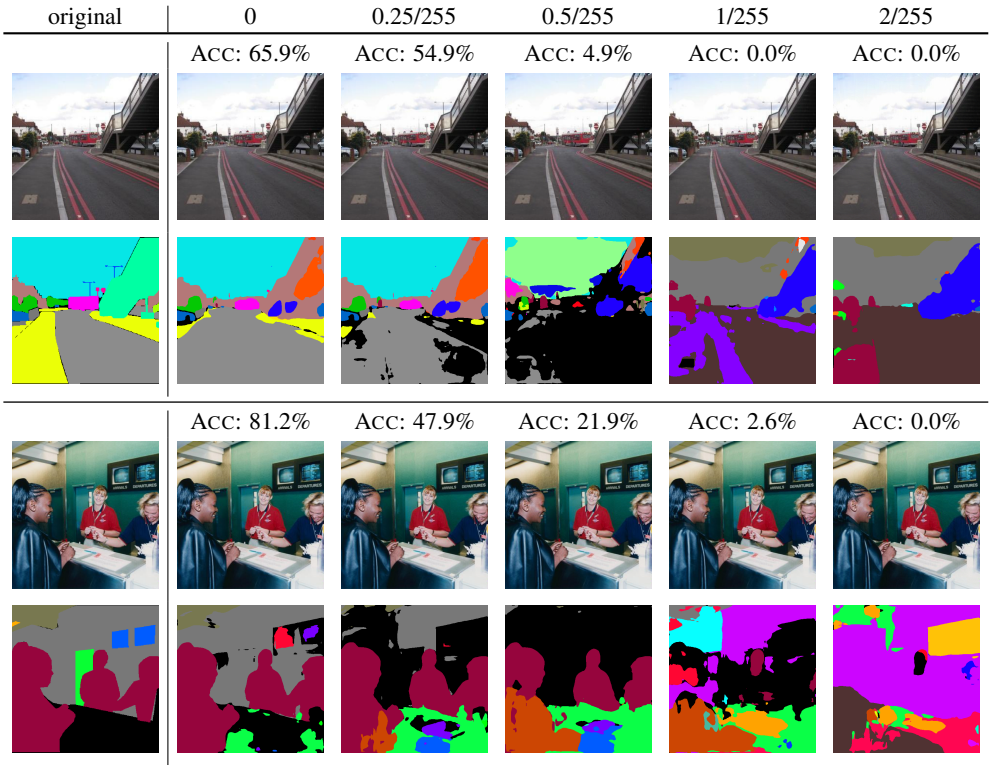

Figure 7: We show the perturbed images, corresponding predicted segmentation masks and average accuracy for increasing radii. The attacks are generated on the clean model on ADE20K with APGD on $\mathcal{L}_{\text{Mask-CE}}$. We additionally present (first column) the original image and ground truth mask.

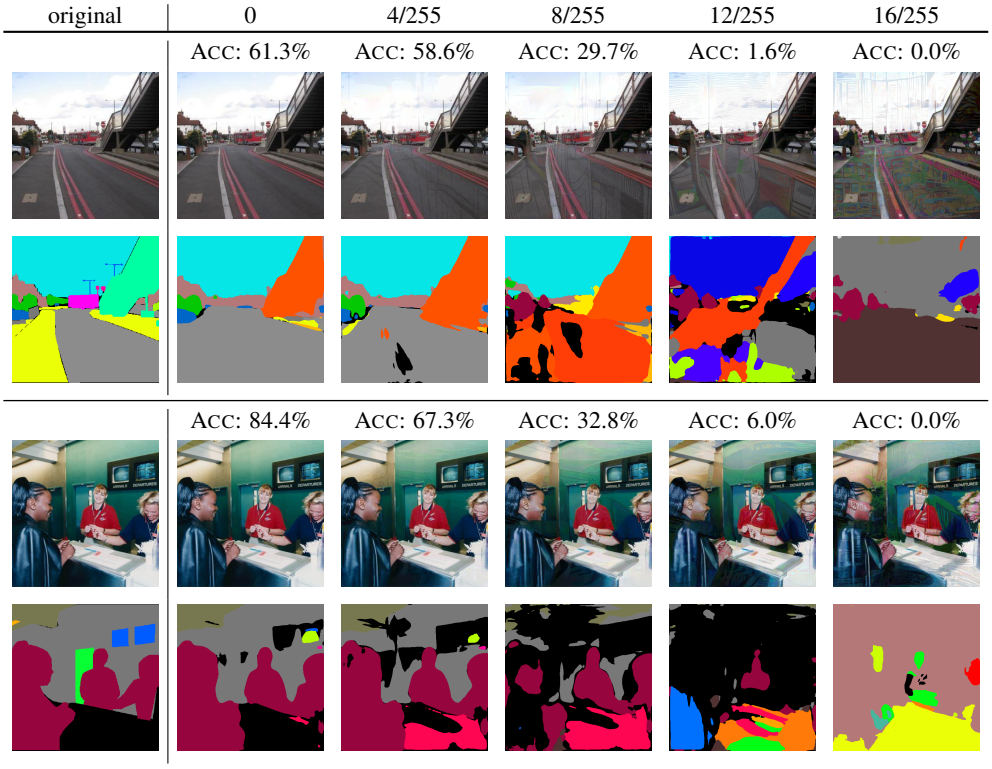

Figure 8: Same setting as in Fig. 7 for the robust AT^5 model. Note the larger radii ϵ for the attack.