# Supplemental Material for Scalable Certified Segmentation via Randomized Smoothing

## A. Experimental Details

### A.1. Experimental Details for §6.2

We use a HrNetV2 (Sun et al., 2019; Wang et al., 2019) with the HRNetV2-W48 backbone from their official PyTorch 1.1 (Paszke et al., 2019) implementation[1]. For Cityscapes we follow the outlined training procedure, only adding the $\sigma = 0.25$ Gaussian noise. For Pascal Context we doubled the number of training epochs (and learning rate schedule) and added the $\sigma = 0.25$ Gaussian noise. During inference we use different batch sizes for different scales. These are summarized in Table 3. All timing timing results are given for a single Nvidia GeForce RTX 2080 Ti and using 12 cores of a Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz. When training on an machine with 8 Nvidia GeForce RTX 2080 Ti one training epoch takes around 4 minutes for both of the data sets.

We evaluate on 100 images each, that is for Cityscapes we use every 5th image in the test set and for Pascal every 51st.

We consider two metrics:

- (certified) per-pixel accuracy: the rate of pixels correctly classified (over all images)
- (certified) mean intersection over union (mIoU): For each image $i$ and each class $c$ (ignoring the $\oslash$ "class") we calculate the ratio $IoU_c^i = \frac{|P_c^i \cap G_c^i|}{|P_c^i \cup G_c^i|}$ where $P_c^i$ denotes the pixel locations predicted as class $c$ for input $i$ and $G_c^i$ denotes the pixel locations for class $c$ in the ground truth of input $i$. We then average $IoU_c^i$ over all inputs and classes.

### A.2. Experimental Details for §6.3

Using the PointNetV2 architecture (Qi et al., 2017a;b; Yan et al., 2020) implemented in PyTorch [2] (Paszke et al., 2019). Again we keep the training parameters unchanged other than the addition of noise during training. One training epoch on a single Nvidia GeForce RTX 2080 Ti takes 6 minutes.

[1] https://github.com/HRNet/HRNet-Semantic-Segmentation

[2] https://github.com/yanx27/Pointnet_Pointnet2_pytorch

*Table 3.* Batch sizes used in segmentation inference.

| scale | Cityscapes | Pascal Context |
|---|---|---|
| 0.25 | 24 | 80 |
| 0.50 | 12 | 64 |
| 0.75 | 4 | 32 |
| 1.00 | 4 | 20 |
| multi | 4 | 10 |

In inference we use a batch size of 50. All timing results are given for a single Nvidia GeForce RTX 2080 Ti and using 12 cores of a Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz.

Again, we evaluate on 100 inputs. This corresponds to every 28th input in the test set. As a metric we consider the (certified) per-component accuracy: the rate of parts/components correctly classified (over all inputs).

## B. Additional Results

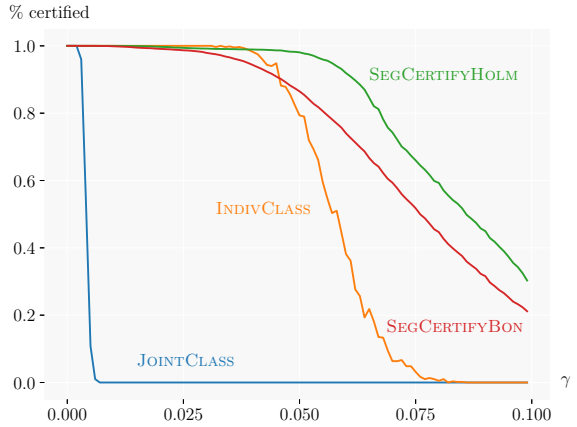
### B.1. Additional Results for §6.1

In Figs. 2b and 2c we show smoothed plots as JOINTCLASS and INDIVCLASS are either correct on all components or non at all. Here, we provide the unsmoothed results in Fig. 5. In order to obtain the plots in Fig. 2 we apply a Savgol filter (Savitzky & Golay, 1964) of degree 1 over the 11 closest neighbours (using the SciPy implementation) and use a step size of 0.001 for $\gamma$.

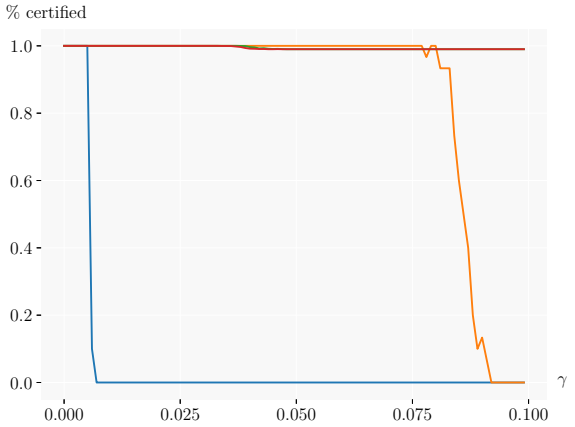
### B.2. $k$-FWER and error budget

Here we discuss the gains from allowing a small budget of errors and applying $k$-FWER control as outlined in §5.2. Control for $k$-FWER at level $\alpha$ means that $P(\geq k \text{ type I errors}) \leq \alpha$. Which for $k = 1$ recovers standard FWER control. Thus, if we allow a budget of $b$ type I errors at level $\alpha$ we need to perform $k$-FWER control with $k = b + 1$. In the following we will refer only to the budget $b$, to avoid confusion between the $k$ in $k$-FWER and the $k$ noisy components in the setting of §6.1.

Fig. 6 shows an empirical evaluation of this approach for different $b$ for $\gamma = 0.05$, one noisy components and different

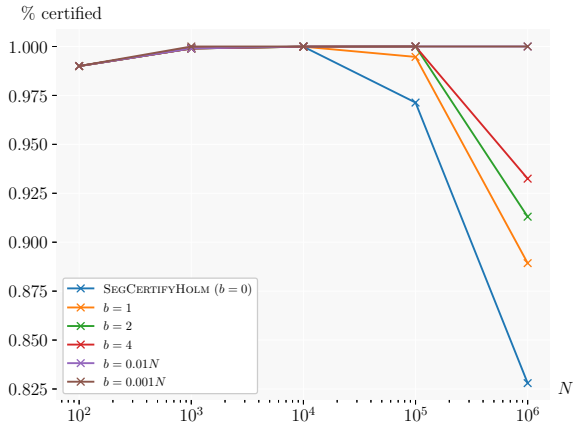



(a) Unsmoothed version of Fig. 2b. On $N = 100$ components, with a classifier that has error rate $5\gamma$ on one component and $\gamma$ on all others.

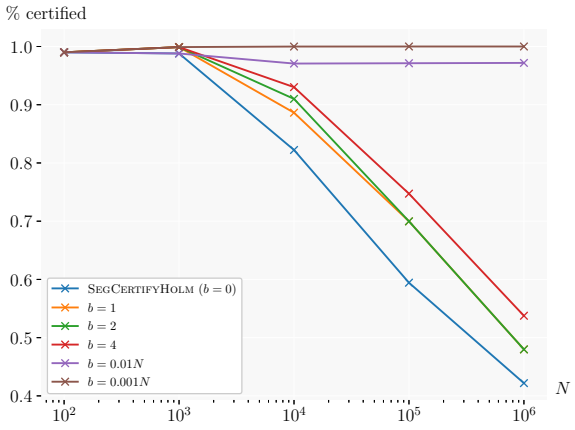



(b) Unsmoothed version of Fig. 2c. SegCertify with different testing corrections for various numbers of components $N$ with error rate $\gamma = 0.05$.

*Figure 5.* Unsmoothed versions of Fig. 2. We empirically investigate the power (ability to avoid type II errors – false abstention) of multiple algorithms on synthetic data. The $y$-axis shows the rate of certified (rather than abstained) components. An optimal algorithm would achieve 1.0 or 0.99 in all plots.




(a) $\alpha = 0.1$.




(b) $\alpha = 0.001$.

*Figure 6.* Evaluation where an error budget of up top $b$ type I errors is allowed. Potentially allowing a small amount of errors greatly increases the power of the test. $N$ varies along the $x$-axis, $\gamma = 0.05$ and $k$ components with error rate $5\gamma$.

levels of $N$. Fig. 6a uses $\alpha = 0.1$ and Fig. 6b $\alpha = 0.001$. We see that allowing a single error leads to a huge gain in power for the $N = 10^6$ setting. Similarly, allowing 1 percent or 1 permille errors greatly strengthen the method.

**False Discovery Rate** Similarly, SegCertify can employ false discovery rate (FDR) control rather than ($k$-)FWER control. FDR control limits the expected number of type I errors. Since this is a much weaker statement than FWER it allows for less type II errors. However, while useful this kind of control leaves the area of (statistical) certified robustness for a more relaxed probabilistic setting which we do not investigate here further.

### B.3. Additional Results for §6.2

Table 8 shows an extended version of Table 1. Both of these tables use Holm correction. Table 4 shows the difference to Table 8 if instead Bonferroni correction was used. Generally this difference is very small in this setting as it appears the true $p_A$ are far from $\tau$. However in some settings (such as multi resolution) the effect of Holm correction can be up to 2%. Since for a particular base classifier or $\tau$ this gain can quickly go from neglectable to significant and since the additional evaluation time ($< 0.1s$) is neglectable compared to the time for sampling we believe Holm correction to be preferable in most cases.

*Table 4.* Difference when Bonferroni correction rather than Holm correction is used in Table 8. Only differences $\geq 10^{-4}$ are shown. We observed no such differences on the Pascal Context dataset.

| | | | | Cityscapes | | |
|---|---|---|---|---|---|---|
| scale | | $\sigma$ | $R$ | acc. | mIoU | %⊘ |
| 0.5 | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | -0.0008 | -0.0005 | 0.0019 |
| | | 0.33 | 0.22 | -0.0014 | -0.0010 | 0.0024 |
| | | 0.50 | 0.34 | -0.0021 | -0.0019 | 0.0031 |
| 0.5 | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | -0.0009 | -0.0014 | 0.0014 |
| | | 0.33 | 0.22 | -0.0012 | -0.0020 | 0.0016 |
| | | 0.50 | 0.34 | -0.0005 | -0.0004 | 0.0006 |
| 1.0 | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | -0.0012 | -0.0032 | 0.0016 |
| | | 0.33 | 0.22 | -0.0020 | -0.0018 | 0.0024 |
| | | 0.50 | 0.34 | -0.0000 | -0.0000 | 0.0001 |
| | SEGCERTIFY $n = 300, \tau = 0.90$ | 0.25 | 0.17 | 0.0001 | 0.0002 | -0.0002 |
| multi | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | -0.0125 | -0.0185 | 0.0173 |

**Consistency Training** Here we investigate a naive instantiation of training approach from Jeong & Shin (2020).

Jeong & Shin (2020) improve the training for classification models used as base models in randomized smoothing by adding a consistency regularization term in training. We use this same term, but compute it for every pixel and average the results. To obtain the results in Table 5 we used $m = 2$, $\lambda = 1$, $\eta = 0.5$ and $\sigma = 0.25$. Depending on the scale we see either slightly better or slightly worse results than with standard Gaussian data augmentation in training. This shows the promise of the method but also highlights the need for potential further specialization to the segmentation setting, particularly by considering the effect of scaling.

### B.4. Additional Results for §6.3

Table 6 shows the change when executing the experiments in Table 2 with Bonferroni correction instead of Holm correction.

### B.5. Certification beyond $\ell_p$

As outlined in §5.2, SEGCERTIFY can be easily adapted to non-$\ell_2$-settings. Here we show that we can certify against and adversary rotating 3d point clouds. A 3d rotation is parameterized by 3 angles which we will denote $\epsilon \in \mathbb{R}^3$ and define $\psi_\epsilon(\boldsymbol{x})\colon \mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times 3}$ as

$$(\psi_\epsilon(\boldsymbol{x}))_i = \boldsymbol{R}_\epsilon \boldsymbol{x}_i, \tag{2}$$

where $\boldsymbol{R}_\epsilon$ denotes the 3d rotation matrix specified by $\epsilon$. The randomized smoothing approach of Fischer et al. (2020) allows to certify robustness in this case: $f(\boldsymbol{x}) = f(\psi_\delta(\boldsymbol{x}))$ for $\delta$ with $\|\delta\|_2 \leq R$, by sampling rotations $\psi_\epsilon(\boldsymbol{x})$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The parameter robustness radius $R$ is computed the same as throughout the paper. When applied to points with a normal vector, Eq. (2) can be extended to apply $R_\epsilon$ to the point coordinates as well as the normal vector.

Using one of the model from §6.3, $f^n_{\sigma=0.5}$ , we perform this version of randomized smoothing.

The results are shown in Table 7. Since the models was not specifically trained to be robust under rotations, the performance quickly deteriorates. Nevertheless we can certify robustness to rotations with a parameter radius $R$ of 0.17 and 0.085 for $\sigma$ of 0.25 and 0.125 respectively.

The same approach can be applied to models that are empirically invariant to most rotations while not formally rotation invariant. In these cases we need to certify a radius of $R = \sqrt{3}\pi$ (when measuring angles in radians). When using a fixed $\tau$, an appropriate $\sigma$ can be chosen as $\sigma = \frac{\sqrt{3}\pi}{\Phi^{-1}(\tau)}$. While this is relativity large $\sigma$, this does not pose an obstacle for a mostly robust base model.

*Table 7.* Results for point cloud part segmentation under 3d rotation. The baseline and base model is $f^n_{\sigma=0.5}$. SEGCERTIFY uses $\tau = 0.75$, $n_0 = 100$, $n = 1000$ and $\alpha = 0.0001$.

| model / $\sigma$ | acc | %⊘ | $t$ |
|---|---|---|---|
| baseline | 0.77 | 0.00 | 0.72 |
| 0.125 | 0.69 | 0.16 | 74.13 |
| 0.25 | 0.61 | 0.26 | 74.51 |

*Table 5.* Same setting as Table 8 but using a model trained with consitency regularization.

| | | | | Cityscapes | | |
|---|---|---|---|---|---|---|
| scale | | $\sigma$ | $R$ | acc. | mIoU | %⊘ |
| 0.25 | base model | - | - | 0.87 | 0.40 | 0.00 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.83 | 0.42 | 0.07 |
| | | 0.33 | 0.22 | 0.84 | 0.42 | 0.09 |
| | | 0.50 | 0.34 | 0.82 | 0.43 | 0.14 |
| 0.50 | base model | - | - | 0.91 | 0.53 | 0.00 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.89 | 0.57 | 0.06 |
| | | 0.33 | 0.22 | 0.89 | 0.57 | 0.07 |
| | | 0.50 | 0.34 | 0.86 | 0.57 | 0.11 |
| 1.00 | base model | - | - | 0.92 | 0.62 | 0.00 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.88 | 0.63 | 0.12 |
| | | 0.33 | 0.22 | 0.79 | 0.46 | 0.20 |
| | | 0.50 | 0.34 | 0.39 | 0.06 | 0.32 |

*Table 6.* Difference when Bonferroni correction rather than Holm correction is used in Table 2.

| | $n$ | $\tau$ | $\sigma$ | acc | %⊘ |
|---|---|---|---|---|---|
| $f_{\sigma=0.25}$ | | | | | |
| | 1000 | 0.75 | 0.250 | -0.0012 | 0.0019 |
| | 1000 | 0.85 | 0.250 | -0.0023 | 0.0029 |
| | 10000 | 0.95 | 0.250 | -0.0013 | 0.0009 |
| | 10000 | 0.99 | 0.250 | -0.0008 | 0.0009 |
| $f^n_{\sigma=0.25}$ | | | | | |
| | 1000 | 0.75 | 0.250 | -0.0020 | 0.0026 |
| | 1000 | 0.85 | 0.250 | -0.0026 | 0.0032 |
| | 10000 | 0.95 | 0.250 | -0.0011 | 0.0010 |
| | 10000 | 0.99 | 0.250 | -0.0013 | 0.0011 |
| $f_{\sigma=0.5}$ | | | | | |
| | 1000 | 0.75 | 0.250 | -0.0007 | 0.0010 |
| | 1000 | 0.75 | 0.500 | -0.0002 | 0.0010 |
| | 1000 | 0.85 | 0.500 | -0.0017 | 0.0028 |
| | 10000 | 0.95 | 0.500 | -0.0007 | 0.0010 |
| | 10000 | 0.99 | 0.500 | -0.0003 | 0.0003 |
| $f^n_{\sigma=0.5}$ | | | | | |
| | 1000 | 0.75 | 0.250 | -0.0009 | 0.0017 |
| | 1000 | 0.75 | 0.500 | -0.0015 | 0.0024 |
| | 1000 | 0.85 | 0.500 | -0.0019 | 0.0028 |
| | 10000 | 0.95 | 0.500 | -0.0043 | 0.0013 |
| | 10000 | 0.99 | 0.500 | -0.0008 | 0.0007 |

*Table 8.* Extended version of Table 1. Segmentation results for 100 images. acc. shows the mean per-pixel accuracy, mIoU the mean intersection over union, %⊘ abstentions and $t$ runtime in seconds. All SEGCERTIFY ($n_0 = 10, \alpha = 0.001$) results are certifiably robust at radius $R$ w.h.p. multiscale uses $0.5, 0.75, 1.0, 1.25, 1.5, 1.75$ as well as their flipped variants for Cityscapes and additionally $2.0$ for Pascal. All numbers are obtained via Holm correction.

| | | | | Cityscapes | | | | Pascal Context | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| scale | | $\sigma$ | $R$ | acc. | mIoU | %⊘ | $t$ | acc. | mIoU | %⊘ | $t$ |
| 0.25 | non-robust model | - | - | 0.93 | 0.60 | 0.00 | 0.38 | 0.59 | 0.24 | 0.00 | 0.12 |
| | base model | - | - | 0.87 | 0.42 | 0.00 | 0.37 | 0.33 | 0.08 | 0.00 | 0.13 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.84 | 0.43 | 0.07 | 70.00 | 0.33 | 0.08 | 0.13 | 14.16 |
| | | 0.33 | 0.22 | 0.84 | 0.44 | 0.09 | 70.21 | 0.34 | 0.09 | 0.17 | 14.20 |
| | | 0.50 | 0.34 | 0.82 | 0.43 | 0.13 | 71.45 | 0.23 | 0.05 | 0.27 | 14.23 |
| | SEGCERTIFY $n = 300, \tau = 0.90$ | 0.25 | 0.32 | 0.83 | 0.43 | 0.10 | 143.37 | 0.32 | 0.08 | 0.23 | 24.33 |
| | | 0.33 | 0.42 | 0.82 | 0.43 | 0.12 | 143.30 | 0.32 | 0.09 | 0.29 | 24.42 |
| | | 0.50 | 0.64 | 0.79 | 0.40 | 0.18 | 143.54 | 0.20 | 0.04 | 0.43 | 24.13 |
| | SEGCERTIFY $n = 500, \tau = 0.95$ | 0.25 | 0.41 | 0.83 | 0.42 | 0.11 | 229.37 | 0.29 | 0.01 | 0.30 | 33.64 |
| | | 0.33 | 0.52 | 0.83 | 0.42 | 0.12 | 230.69 | 0.26 | 0.01 | 0.39 | 33.79 |
| | | 0.50 | 0.82 | 0.77 | 0.38 | 0.20 | 230.09 | 0.10 | 0.00 | 0.61 | 33.44 |
| | SEGCERTIFY $n = 10000, \tau = 0.99$ | 0.25 | 0.58 | - | - | - | - | 0.25 | 0.07 | 0.48 | 557.29 |
| | | 0.33 | 0.77 | - | - | - | - | 0.24 | 0.07 | 0.58 | 557.34 |
| | | 0.50 | 1.17 | - | - | - | - | 0.11 | 0.03 | 0.77 | 557.32 |
| 0.5 | non-robust model | - | - | 0.96 | 0.76 | 0.00 | 0.39 | 0.74 | 0.38 | 0.00 | 0.16 |
| | base model | - | - | 0.89 | 0.51 | 0.00 | 0.39 | 0.47 | 0.13 | 0.00 | 0.14 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.88 | 0.54 | 0.06 | 75.59 | 0.48 | 0.16 | 0.09 | 16.29 |
| | | 0.33 | 0.22 | 0.87 | 0.54 | 0.08 | 75.99 | 0.50 | 0.17 | 0.11 | 16.08 |
| | | 0.50 | 0.34 | 0.86 | 0.54 | 0.10 | 75.72 | 0.36 | 0.10 | 0.21 | 16.14 |
| | SEGCERTIFY $n = 300, \tau = 0.90$ | 0.25 | 0.32 | 0.87 | 0.53 | 0.08 | 143.40 | 0.46 | 0.15 | 0.17 | 27.17 |
| | | 0.33 | 0.42 | 0.86 | 0.52 | 0.10 | 145.90 | 0.47 | 0.16 | 0.21 | 27.17 |
| | | 0.50 | 0.64 | 0.83 | 0.50 | 0.15 | 144.61 | 0.31 | 0.10 | 0.38 | 27.32 |
| | SEGCERTIFY $n = 500, \tau = 0.95$ | 0.25 | 0.41 | 0.86 | 0.52 | 0.09 | 228.63 | 0.45 | 0.19 | 0.21 | 38.27 |
| | | 0.33 | 0.52 | 0.85 | 0.51 | 0.11 | 228.38 | 0.46 | 0.16 | 0.26 | 38.43 |
| | | 0.50 | 0.82 | 0.82 | 0.49 | 0.16 | 228.73 | 0.30 | 0.09 | 0.44 | 38.37 |
| 0.75 | non-robust model | - | - | 0.97 | 0.80 | 0.00 | 0.46 | 0.76 | 0.41 | 0.00 | 0.15 |
| | base model | - | - | 0.90 | 0.59 | 0.00 | 0.47 | 0.55 | 0.18 | 0.00 | 0.15 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.88 | 0.57 | 0.09 | 82.69 | 0.53 | 0.19 | 0.15 | 16.78 |
| | | 0.33 | 0.22 | 0.86 | 0.56 | 0.12 | 82.87 | 0.54 | 0.20 | 0.20 | 16.83 |
| | | 0.50 | 0.34 | 0.64 | 0.27 | 0.31 | 82.19 | 0.29 | 0.07 | 0.33 | 16.83 |
| | SEGCERTIFY $n = 300, \tau = 0.90$ | 0.25 | 0.32 | 0.84 | 0.54 | 0.13 | 177.44 | 0.51 | 0.19 | 0.22 | 29.48 |
| | | 0.33 | 0.42 | 0.84 | 0.52 | 0.15 | 177.22 | 0.51 | 0.20 | 0.28 | 29.58 |
| | | 0.50 | 0.64 | 0.60 | 0.24 | 0.37 | 177.67 | 0.25 | 0.06 | 0.45 | 29.53 |
| 1.0 | non-robust model | - | - | 0.97 | 0.81 | 0.00 | 0.52 | 0.77 | 0.42 | 0.00 | 0.18 |
| | base model | - | - | 0.91 | 0.57 | 0.00 | 0.52 | 0.53 | 0.18 | 0.00 | 0.18 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.88 | 0.59 | 0.11 | 92.75 | 0.55 | 0.22 | 0.22 | 18.53 |
| | | 0.33 | 0.22 | 0.78 | 0.43 | 0.20 | 92.85 | 0.46 | 0.18 | 0.34 | 18.57 |
| | | 0.50 | 0.34 | 0.34 | 0.06 | 0.40 | 92.48 | 0.17 | 0.03 | 0.41 | 18.46 |
| | SEGCERTIFY $n = 300, \tau = 0.90$ | 0.25 | 0.32 | 0.86 | 0.56 | 0.14 | 204.82 | 0.53 | 0.21 | 0.29 | 33.83 |
| | | 0.33 | 0.42 | 0.75 | 0.40 | 0.24 | 204.58 | 0.42 | 0.17 | 0.44 | 33.78 |
| | | 0.50 | 0.64 | 0.31 | 0.05 | 0.47 | 204.57 | 0.15 | 0.03 | 0.52 | 33.43 |
| multi | non-robust model | - | - | 0.97 | 0.82 | 0.00 | 8.98 | 0.78 | 0.45 | 0.00 | 4.21 |
| | base model | - | - | 0.92 | 0.60 | 0.00 | 9.04 | 0.56 | 0.19 | 0.00 | 4.22 |
| | SEGCERTIFY $n = 100, \tau = 0.75$ | 0.25 | 0.17 | 0.88 | 0.57 | 0.09 | 1040.55 | 0.52 | 0.21 | 0.29 | 355.00 |

## C. Details on Attacks & Fig. 1

To produce the visualization in Fig. 1 we used a custom PGD attack described below and produced an $\ell_2$ adversarial example within a range of $0.16$. We performed certification with $n = 100, \alpha = 0.001, \sigma = 0.25$ and $\tau = 0.75$, which certifies robustness at an radius of $R = \sigma\Phi^{-1}(\tau) = 0.1686$. We perform this certification on the clean input and thus show robustness to the attack. As the non-robust model for Fig. 1c we used a pretrained HrNet from the same repository as outlined in App. A.1.

We use $k = 100$ steps for the attack and step size $s = \frac{10*R}{k}$. While scaling can be simply incorporated into the attack, we use scale $1.0$ for both the attacked an the certified classifier.

We used an Nvidia Titan RTX to perform these attacks as the memory requirement exceeds that of a single Nvidia GeForce RTX 2080 Ti.

Here, in Figs. 7 and 8 we provide further visualizations like Fig. 1. The visualization in Fig. 7 is hand-picked (like Fig. 1), while the ones in Fig. 8 are chosen randomly from the evaluated images.

### C.1. PGD for Segmentation

Following the work of Madry et al. (2018), Arnab et al. (2018) and Xie et al. (2017) we use a slightly generalized form of the untargeted $\ell_2$ projected gradient decent (PGD) attack. This version is the same as untargeted PGD (Madry et al., 2018) in the classification setting, but we adapt the loss to be the average of all pixel losses as in Xie et al. (2017). Formally, for an input $\boldsymbol{x} \in \mathcal{X}^N = [0,1]^{N\times 3}$ with ground truth segmentation $\boldsymbol{y} \in \mathbb{Y}^N$ and model $f$ a radius $R$ and stepsize $s \in \mathbb{R}$ we produce an adversarial example $\boldsymbol{x}'_k$ after $k$ steps.

$$\begin{aligned}\boldsymbol{x}'_0 &= \boldsymbol{x} + \epsilon, \qquad \epsilon \sim [0,1] \text{ with } \|\epsilon\|_2 \leq R \\ \boldsymbol{x}'_{i+1} &= c_{R,\boldsymbol{x}}(\boldsymbol{x}'_i + s\nabla_{\boldsymbol{x}'_i}\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}))\end{aligned}$$

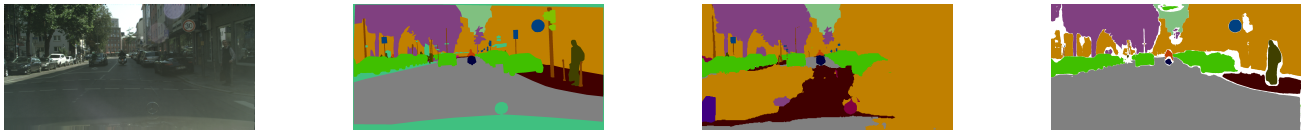
with clamping function

$$\begin{aligned}c_R &\colon \mathbb{R}^{N\times 3} \to [0,1]^{N\times 3} \\ c_{R,\boldsymbol{x}}(\boldsymbol{x}') &= \left[\boldsymbol{x} + R\frac{\boldsymbol{x}' - \boldsymbol{x}}{\|\boldsymbol{x}' - \boldsymbol{x}\|_2}\right],\end{aligned}$$

where $[\cdot]$ denotes component-wise clamping to $[0,1]$, and loss

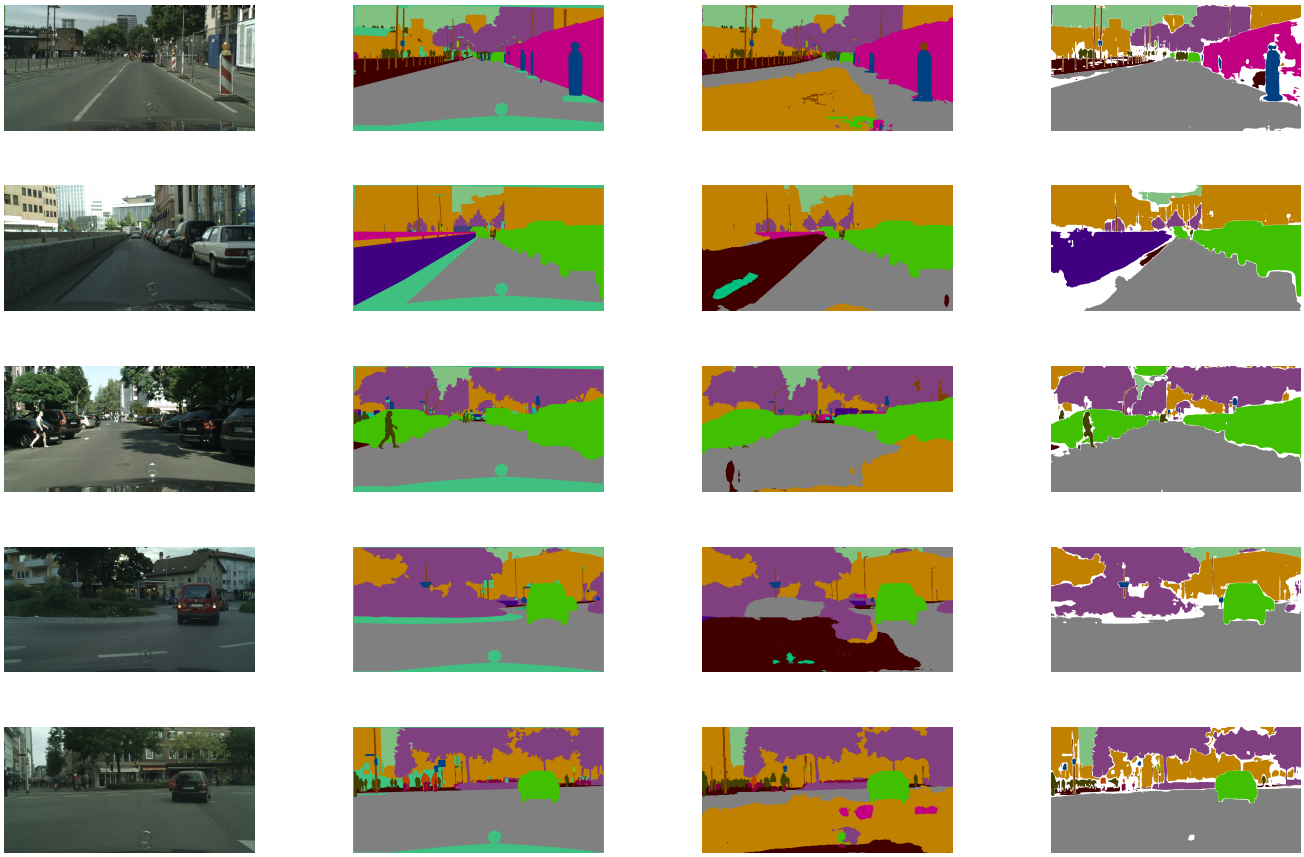
$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{H}(f_i(\boldsymbol{x}), \boldsymbol{y}_i), \tag{3}$$

where $\mathcal{H}$ denotes the cross entropy function.



(a) Attacked image (b) Ground truth seg. (c) Attacked segmentation (d) Certified segmentation

*Figure 7.* Another hand-picked example like Fig. 1.



(a) Attacked image (b) Ground truth seg. (c) Attacked segmentation (d) Certified segmentation

*Figure 8.* Randomly chosen examples like Fig. 1.