Jedi: Entropy-based Localization and Removal of Adversarial Patches - Supplementary Materials

1. Parameters exploration

The first parameter considered is the size of the sliding window used for local entropy calculation. The recovery rate and average precision for each window size considered is shown in Figure 1.

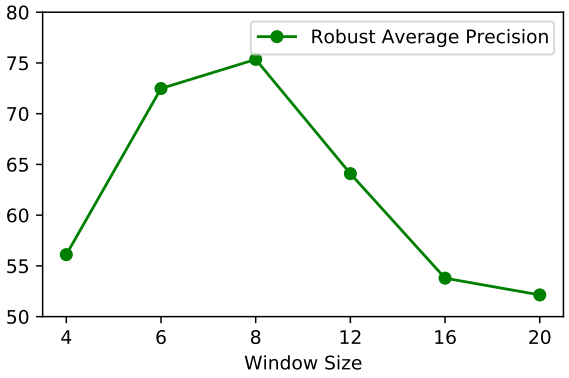


Figure 1. Robust Average Precision for different window sizes

These results show that beyond a certain size, the chosen window generates an imprecise inpainting mask, leading to lower recovery rates. This phenomenon is also observed for very small window sizes, as the entropy of smaller windows is more likely to be below the inpainting threshold. These results show that there is an optimal window size that corresponds to the image's resolution. Therefore, using a static window size is counterproductive; We adopt a dynamic window size equal to 1% of the images largest dimension, with a minimum size of 8 pixels.

The second parameter we explore is the stride of the local entropy window. We use the same sample dataset as the previous experiment and a window size of 8 pixels. The results are show in Table 1.

The results show that lowering the window stride leads to a better performing defense, as a more granular local entropy calculation results in a more accurate inpainting mask. However, beyond a certain point, the improvements brought by lowering the window stride are quickly outpaced by the negative effect of exponentially increasing the number of entropy calculations. Therefore, we set the window stride to be equal to half of the window size as this value keeps most of the performance gains from lowering the window stride while keeping calculation time reasonable.

Table 1. Effect of window stride on defense performance

Stride	**1**	**4**	**8**
Clean	100%	100%	100%
Patch Success Rate	72.20%	72.20%	72.20%
Detected Patches	92,20%	90.50%	87.45%
Robust Avg Precision	81.50%	79.73%	75.35%

2. Alternate method for creating the patch mask

An earlier version of our work investigated an alternate method of creating the mask that locates the adversarial patch. This method follows the same framework as our main proposed approach, but the methods for determining the high entropy kernels and obtaining the full shape of the patch differ:

First, to determine the locations of high entropy kernels, a local entropy heatmap is established using a sliding window, and the kernels are located by keeping only the peaks of this heatmap: Only windows whose entropy value exceeds the top-$k\%$ of the maximum entropy value are considered as high entropy kernels, the rest of the heatmap is discarded.

Next, instead of using an autoencoder to determine the final mask, we "expand" our kernels to fit the shape of the adversarial patch: We compare each high entropy kernel with the windows that surround it, in an n-window radius using a similarity metric. If the neighboring window is similar to the high entropy kernel, it is added to the final patch mask.

The similarity metric we use in this method is mutual information. For each kernel, we determine the self-mutual information of the kernel as a baseline, and we compare the mutual information between the high entropy kernel and candidate window for expansion. If the ratio of this mutual information value relative to the self-mutual information is

Table 2. Performance of the alternative patch localization method with comparison to the main method

Experiment	Metric	Main	Alternative
ImageNet + [3] + InceptionV3	Robust Accuracy	55.26%	51.36%
	Recovery Rate	75.26%	70.45%
Pascal VOC 07 + [1] + ResNet50	Robust Accuracy	66.40%	66.60%
	Recovery Rate	82.74%	83.51%
CASIA + [7] + YOLOv2	Robust Avg Precision	88.21%	70.52%
	Recovery Rate	94.38%	84.15%

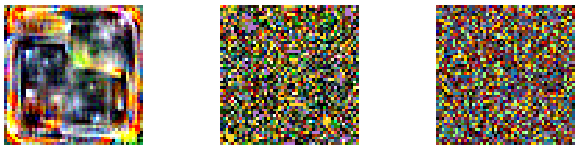
higher than the threshold. The candidate window is considered as part of the adversarial patch.

These two steps produce a mask that shows the location of the adversarial patch, we then filter this mask and use it to apply the inpainting method in the same way as discussed in Section 3.3 of the main paper that presents our main proposed approach.

The performance of this alternative approach and the difference in recovery rates when compared against our main method are show in table 2:

This method's results are inferior in most cases to our main method's results, and equal in the best of cases. Furthermore, this approach may have some scalability issues, especially in images with higher resolutions and when using smaller window sizes or when there is a large number of high entropy kernels in the image. Nonetheless, this alternative approach may have its merits in situations where it is impractical to use an autoencoder or if preparing the clean dataset entropy distribution is not possible.

3. Illustration of entropy budget on the generated patches



(a) No entropy limit (b) Entropy = 7 (c) Entropy = 5

Figure 2. Samples of the adaptive patch under entropy limits

4. Evaluation for different patch sizes

Adversarial patches' size is a hyper-parameter fixed in the noise generation process, it has a significant impact on the attack efficiency. Therefore it is necessary to evaluate our defense against a variety of patch sizes. In Figure 3 we repeat the experiments on CASIA with the different patch sizes relative to the bounding box width.

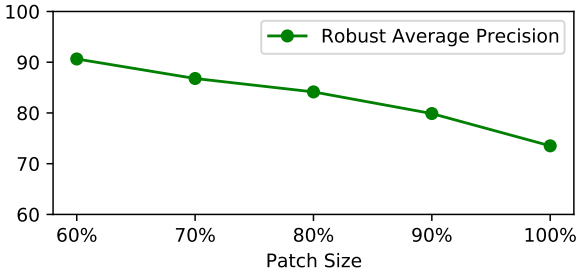


Figure 3. Robust Average Precision for different patch sizes

As expected, patch success rate increases with patch size, however while the recovery rate is lower for bigger patch size, the drop is small relative to the patch size: The recovery rate is still high even when the adversarial patch is extremely large. Notice that excessively large patches can be impractical in a real-life scenario.

5. Full result tables

In this subsection of the supplementary materials, we include the full tables with detailed results of each CNN and adversarial patch we have applied to the four datasets used in our evaluation:

- ImageNet results are shown in table 3
- Pascal VOC 07 results are shown in table 4
- CASIA results are shown in table 5
- INRIA results are shown in table 6

Table 3. Evaluation results for the experiments performed on ImageNet

Patch: [1], trained on Imagenet						
Classifier	VGG-16	VGG-19	ResNet-18	ResNet-50	ResNet-101	InceptionV3
Clean Accuracy	68.80%	69.87%	66.56%	74.10%	75.53%	67.65%
Patch Success Rate	31.16%	31.96%	29.34%	49.02%	31.05%	23.10%
Adversarial Accuracy	50.04%	50.37%	50.45%	39.26%	54.45%	54.93%
Patch Detection Rate	85.31%	85.31%	85.31%	85.31%	85.31%	85.31%
Robust Accuracy	57.86%	58.53%	58.20%	64.34%	66.28%	59.09%
Recovery Rate	45.18%	44.75%	49.87%	66.95%	55.22%	43.72%
Lost Predictions	5.38%	4.88%	5.15%	2.77%	3.48%	5.98%
Patch: [3], trained on Imagenet						
Classifier	VGG-16	VGG-19	ResNet-18	ResNet-50	ResNet-101	InceptionV3
Clean Accuracy	68.80%	69.87%	66.56%	74.10%	75.53%	67.65%
Patch Success Rate	25.86%	29.33%	28.00%	19.57%	20.52%	93.22%
Adversarial Accuracy	54.30%	52.43%	51.31%	62.61%	62.76%	4.82%
Patch Detection Rate	89.22%	89.22%	89.22%	89.22%	89.22%	89.22%
Robust Accuracy	58.73%	58.50%	58.37%	65.72%	66.68%	55.26%
Recovery Rate	38.54%	41.29%	46.65%	36.03%	39.47%	75.26%
Lost Predictions	5.10%	6.09%	3.78%	4.28%	4.63%	3.33%

Table 4. Evaluation results for the experiments performed on Pascal VOC 07

Patch: [1], trained on Imagenet						
Classifier	VGG-16	VGG-19	ResNet-18	ResNet-50	ResNet-101	InceptionV3
Clean Accuracy	77.36%	77.12%	70.78%	72.17%	74.09%	70.17%
Patch Success Rate	16.86%	11.99%	17.95%	18.63%	18.29%	19.25%
Adversarial Accuracy	66.98%	70.78%	61.41%	61.65%	63.61%	59.77%
Patch Detection Rate	89.51%	89.51%	89.51%	89.51%	89.51%	89.51%
Robust Accuracy	71.53%	72.27%	67.33%	67.73%	70.30%	64.82%
Recovery Rate	53.10%	38.21%	58.51%	59.31%	64.68%	60.39%
Lost Predictions	3.39%	2.23%	2.54%	2.54%	2.77%	4.63%
Patch: [3], trained on Imagenet						
Classifier	VGG-16	VGG-19	ResNet-18	ResNet-50	ResNet-101	InceptionV3
Clean Accuracy	77.36%	77.12%	70.78%	72.17%	74.09%	70.17%
Patch Success Rate	10.34%	9.03%	13.10%	13.63%	15.32%	23.08%
Adversarial Accuracy	71.79%	72.80%	65.21%	65.43%	65.67%	56.89%
Patch Detection Rate	91.33%	91.33%	91.33%	91.33%	91.33%	91.33%
Robust Accuracy	72.37%	72.94%	66.96%	67.43%	70.23%	64.36%
Recovery Rate	28.79%	24.06%	43.79%	44.76%	60.68%	58.23%
Lost Predictions	2.68%	2.62%	2.99%	2.79%	3.51%	3.82%
Patch: [1], trained on Pascal VOC 07						
Classifier	VGG-16	VGG-19	ResNet-18	ResNet-50	ResNet-101	InceptionV3
Clean Accuracy	77.36%	77.12%	70.78%	72.17%	74.09%	70.17%
Patch Success Rate	22.81%	20.84%	35.35%	63.88%	26.55%	22.42%
Adversarial Accuracy	62.32%	63.67%	48.30%	26.94%	56.91%	57.65%
Patch Detection Rate	92.62%	92.62%	92.62%	92.62%	92.62%	92.62%
Robust Accuracy	72.29%	72.15%	67.14%	66.40%	69.97%	65.53%
Recovery Rate	65.90%	60.68%	76.43%	82.74%	70.64%	62.77%
Lost Predictions	2.94%	2.38%	2.25%	1.32%	2.41%	3.71%

Table 5. Evaluation results for the experiments performed on CASIA

Patch: [7], trained on Wildtrack			
Detector	YoloV2	YoloV3	YoloV4
Clean Average Precision	91,47%	92.47%	87.85%
Patch Success Rate	26.10%	60.55%	18.18%
Adversarial Average Precision	39.60%	20.31%	58.46%
Patch Detection Rate	95.31%	95.31%	95.31%
Robust Average Precision	88.21%	76.17%	85.10%
Recovery Rate	94.38%	79.98%	89.47%
Lost Predictions	0.07%	4.36%	0.26%
Patch: [2], trained on CASIA			
Detector	YoloV2	YoloV3	YoloV4
Clean Average Precision	91,47%	92.47%	87.85%
Patch Success Rate	7.09%	63.61%	19.72%
Adversarial Average Precision	76.56%	23.53%	60.61%
Patch Detection Rate	95.59%	95.59%	95.59%
Robust Average Precision	85.22%	72.68%	83.45%
Recovery Rate	82.83%	76.95%	83.70%
Lost Predictions	0.48%	5.03%	0.50%

Table 6. Evaluation results for the experiments performed on INRIA

Patch: [7], trained on Wildtrack			
Detector	YoloV2	YoloV3	YoloV4
Clean Average Precision	51.11%	81.51%	85.77%
Patch Success Rate	61.15%	33.88%	37.02%
Adversarial Average Precision	12.17%	41.80%	38.44%
Patch Detection Rate	38.80%	38.80%	38.80%
Robust Average Precision	28.03%	53.07%	50.71%
Recovery Rate	41.88%	43.61%	32.57%
Lost Predictions	1.14%	3.39%	2.03%
Patch: [2], trained on INRIA			
Detector	YoloV2	YoloV3	YoloV4
Clean Average Precision	51.11%	81.51%	85.77%
Patch Success Rate	34.22%	29.40%	28.94%
Adversarial Average Precision	33.04%	52.22%	48.71%
Patch Detection Rate	56.56%	56.56%	56.56%
Robust Average Precision	47.04%	65.11%	65.49%
Recovery Rate	68.39%	51.27%	53.92%
Lost Predictions	2.35%	2.33%	2.40%

6. Sample images of the Jedi defense

The final section of the supplementary materials showcases some samples from our various experiments:

- ImageNet samples are shown in figure 4
- Pascal VOC 07 samples are shown in figure 5
- CASIA samples are shown in figure 6
- INRIA samples are shown in figure 7
- A traffic sign dataset sample is shown in figure 8

7. Additional Certified Defense evaluations

In addition to the certified defenses evaluated in the main paper (Derandomized Smoothing [4], Patchguard [8], Smoothed-Vit [6]), we also evaluate another certified defense: ViP [5]. The additional certified defense is also evaluated on ImageNet using [1]. Table 7 shows the robust accuracy of all the evaluated certified defenses.

Table 7. Robust accuracy of evaluated certified defenses in comparison to Jedi

Defense	**Robust Accuracy**
Jedi	64.34%
[4]	35.02%
[8]	30.96%
[6]	40.38%
[5]	58.29%

Figure 4. Sample images of Jedi on the Imagenet dataset

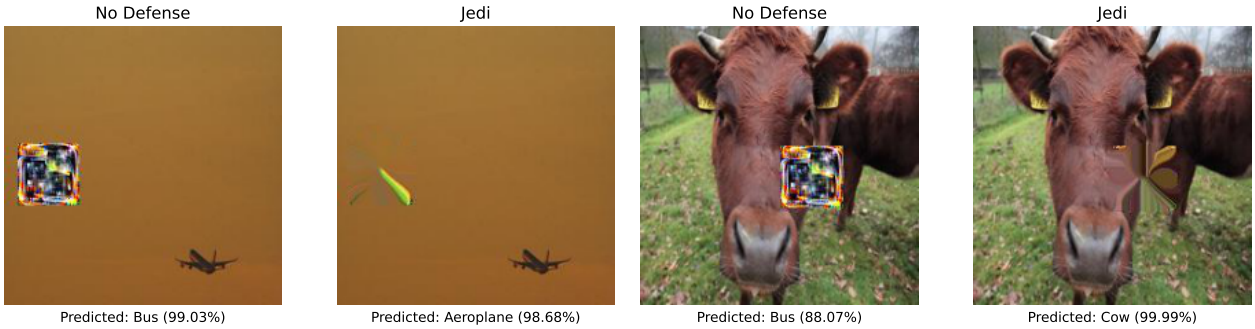


Figure 5. Sample images of Jedi on the Pascal VOC 07 dataset

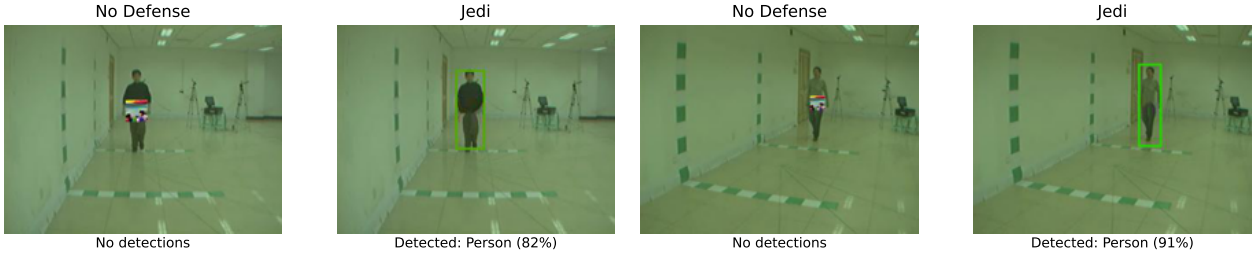


Figure 6. Sample images of Jedi on the CASIA dataset

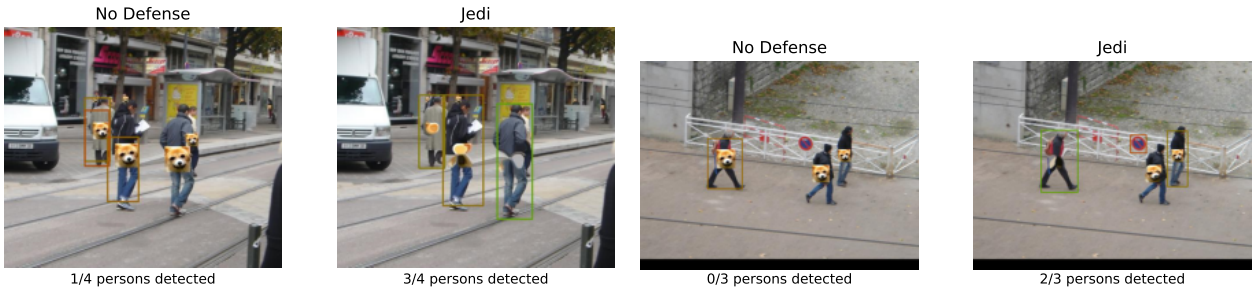


Figure 7. Sample images of Jedi on the INRIA dataset

Figure 8. Sample images of Jedi on a traffic sign dataset

References

[1] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. 2, 3, 4, 6

[2] Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7848–7857, October 2021. 5

[3] Danny Karmon, Daniel Zoran, and Yoav Goldberg. LaVAN: Localized and visible adversarial noise. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2507–2515. PMLR, 10–15 Jul 2018. 2, 3, 4

[4] Alexander Levine and Soheil Feizi. (de)randomized smoothing for certifiable defense against patch attacks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6465–6475. Curran Associates, Inc., 2020. 6

[5] Junbo Li, Huan Zhang, and Cihang Xie. Vip: Unified certified detection and recovery for patch attack with vision transformers. In *ECCV*, 2022. 6

[6] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified patch robustness via smoothed vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15137–15147, June 2022. 6

[7] Simen Thys, Wiebe Van Ranst, and Toon Goedeme. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 2, 5

[8] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *30th USENIX Security Symposium (USENIX Security)*, 2021. 6