A Implementation Details

To promote reproducibility, we provide all necessary implementation details in this Appendix. Statistics regarding all the datasets we used in our experiments are provided in Table 6. To train deep neural networks, we use the open-source PyTorch library (Paszke et al., 2019). For adversarial training, we use the open-source Robustness library (Engstrom et al., 2019) which was developed by Madry Lab. For autoPGD attack evaluation, we use the AutoAttack official code.[1] For training and evaluation using the randomized smoothing framework, we use the code provided by Jeong *et al.* (Jeong & Shin, 2020).[2] All the code and instructions for replicating our experiments are available at `[REDACTED]`.[3]

Table 6: Statistics for all datasets used in our experiments.

Dataset	**# Train Images**	**# Classes**	**# Test Images**	**Skip**	**# Certified Images**
ImageNet	1,281,167	1,000	50,000	100	500
Food	75,750	101	25,250	50	505
CIFAR-10/100	50,000	10/100	10,000	20	500
Birdsnap	32,677	500	8,171	16	511
☼	19,850	397	19,850	39	509
Caltech-256	15,420	257	15,189	30	506
Cars	8,141	196	8,041	16	503
Aircraft	6,667	100	3,333	6	556
DTD	3,760	47	1,880	4	470
Pets	3,680	37	3,669	7	524
Caltech-101	3,030	101	5,647	11	513
Flowers	2,040	102	6,149	12	512

Input Pre-processing. For all experiments, we fix the dimension of the input image to 224×224. For cases where the image is of smaller resolution (*i.e.,* CIFAR-10 and CIFAR-100), we upscale it first during the input pre-processing stage. The complete set of pre-processing steps we perform are as follows:

```
TRAIN_TRANSFORMS = transforms.Compose([
    transforms.RandomSizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize()
])

TEST_TRANSFORMS = transforms.Compose([
    transforms.Scale(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize()
])
```

We follow prior works (He et al., 2019; Kornblith et al., 2019; Salman et al., 2020) and only use normalization for the ImageNet, CIFAR-10, and CIFAR-100 datasets.

Training. When training from scratch, we perform hyperparameter tuning by performing grid search over lr $\in \{0.1, 0.01, 0.05, 0.001\}$, batch size $\in \{256, 128, 64, 32\}$, and weight decay $\in$

[1] `https://github.com/fra31/auto-attack`

[2] `https://github.com/jh-jeong/smoothing-consistency`

[3] Redacted to honor the double-blind review process.

$\{1e-04, 1e-03, 1e-02\}$. Before terminating training, the learning rate is decayed twice by a factor of 0.1 when the performance on validation set doesn't improve for 30 epochs. For ImageNet pre-training, we use publicly available weights for Adversarial Training[4] and SimCLR.[5] Since ImageNet pre-trained weights are not publicly available for the Consistency Regularization method, we generate them ourselves using hyperparameter details provided by the authors (Jeong & Shin, 2020). For all training, we use the Stochastic Gradient Descent (SGD) optimizer.

Certification Using Randomized Smoothing. During certification, we use $\sigma = 0.5$ and follow Cohen *et al.* (Cohen et al., 2019) for all other hyperparameters, *i.e.,* $N_0 = 100$, $N = 100,000$, and failure probability $\alpha = 0.001$. Also following prior works, we certify about 500 test images for each dataset, by skipping every n^{th} image in the complete test set (see Table 6 for skip factor used).

[4] https://github.com/microsoft/robust-models-transfer

[5] https://github.com/facebookresearch/vissl/blob/main/MODEL_ZOO.md