InAugment: Improving Classifiers via Internal Augmentation Supplemental Material

Moab Arar[1], Ariel Shamir[2], and Amit Bermano[1]

[1]Tel-Aviv University
[2]The Interdisciplinary Center Herzliya

1. Alternative Implementations

In our implementation, we copy different patches and apply the same image transformation on the copied patches. In this section, we give a detailed explanation of our implementation and also consider different implementations. In order to compare these implementations, we train a PreAct-ResNet18 [2, 1] on the CIFAR-100 [3] dataset. All reported results are obtained by averaging seven different runs. In all experiments, we copy two patches of size 32×32. The first patch is not resized, while the second patch's dimension is scaled by 0.5. It is worth noting that this is not the best configuration obtained for the PreAct-ResNet18 [2, 1] model; however, we want to observe the effect InAugment has on the network's performance when multiple patches are used.

Setup 1 (AA $\rightarrow$ InAugment): in this setup, the input image undergoes the standard augmentation, i.e., random-crop and random flip, and a random sub-policy of Auto Augment is then applied. Thereafter we apply InAugment by copying patches from the augmented image.

Setup 2 (InAugment $\rightarrow$ AA): in this setup, we first apply the standard augmentation. Afterward, we apply InAugment, and finally, we perform the Auto Augment augmentation. We also applied InAugment first (in the order of operations) but found that the former order performed better.

Setup 3 (InAugment & AA on patches): in this setup, we first copy patches from the input image. Before pasting the patches, we apply different augmentations on the patches and base image. We found that it is better not applying random-crop on the patches. Therefore, the base image undergoes the standard CIFAR augmentation plus Auto Augment. The patches, on the other hand, only undergo random flipping and Auto Augment.

	Setup 1	Setup 2	Setup 3	Setup 4 (Ours)
Exp. 1	80.03	79.84	79.98	80.36
Exp. 2	80.07	80.01	80.26	80.30
Exp. 3	79.72	79.83	80.48	80.20
Exp. 4	80.36	79.72	79.88	80.20
Exp. 5	80.07	79.71	80.09	80.00
Exp. 6	**80.50**	80.26	79.80	79.74
Exp. 7	79.79	80.24	80.24	**80.57**
avg. $\pm$ std.	80.07 $\pm$ .30	79.94 $\pm$ .23	80.10 $\pm$ 0.24	**80.19** $\pm$ **.25**

Table 1: Top-1 accuracy report for different InAugment implementations. We report the best top-1 accuracy obtained for each setup on seven independent runs. In the last row, we report the average accuracy and the standard deviation of all the corresponding setup experiments.

Setup 4 (InAugment & same AA on patches): in this setup, we copy patches from the input image. The copied patches and the base image undergo the same augmentation, except for random-cropping, which is not performed on the copied patches. Unlike Setup-3, we apply the same augmentation on both the patches and base image, including the same random-flip and Auto Augment sub-policy. It is important to note that the operations in Auto-Augment are applied randomly. However, we found that if an image transformation operation is applied, it should be performed on all images (i.e., the copied patches and the base image).

Results: the results for each experiment setup are reported in Table 1. As can be seen, applying InAugment on the input image before augmentation (Auto Augment) achieves the worst accuracy. The second worst implementation is when we first apply Auto Augment and then perform InAugment on the augmented image. We believe that because, in this setup, the image right-before InAugment contains many padded pixels (e.g., due to rotation, cropping), and there is a good chance that the copied patches will also contain these pixels. Therefore, the patches will

not contain meaningful information. Specifically, it is best to copy patches from the input image before any augmentation. Also, note that applying different augmentations on the patches (setup 3) yields the second-best performance (the performance drop for larger networks is even more evident). We believe that since sub-policies in Auto Augment are applied with some pre-defined probability, applying different sub-policies for each patch independently will result in a patch that is most likely not augmented at all, which will bias the network towards identifying un-augmented patches.

2. Additional Out-of-distribution Samples

In the paper, we showed that incorporating InAugment as an image augmentation technique yields robust models to scale. The idea is that exposing the network during training to a multi-scale view of the same image encourages the network to become scale-invariant. To test this hypothesis, we evaluated our network on rescaled versions of the ImageNet [4] validation set. We consider three alternatives: (1) zero-padding, (2) tiling, and (2) symmetric-padding with Gaussian blur to scale the entire validation split. We briefly discuss each rescaling method, and we also give an overview of the standard ImageNet validation set pre-processing step:

- **The standard pre-processing of the validation set:** images in the validation set are center-cropped to a size that is roughly 90% of the input image. The cropped images are then resized to a fixed dimension of 224×224.
- **Zero-Padding:** the input images are zero padded (in each direction) by a fixed padding size $D \in [64, 128, 256, 512]$. The images are then transformed using the standard validation set transformation.
- **Tiling:** rescaling by tiling is done as follows: each image is tiled x-times along each dimension, and the tiled image is then rescaled using the standard ImageNet transformation. In general, tiling the image x-times will scale the final image by $1/x$ since the output size is always fixed to 224×224. For tiling, we consider 4 versions, where for each version, we tile each the image T times, such that $T \in [1.5, 2, 3, 4]$. Note that for $T = 1.5$, we tile only half the image in each direction.
- **Symmetric-padding & Gaussian blur:** similar to zero-padding, we pad each image along both dimensions. However, we use symmetric-padding and the padded pixels are blurred using the Gaussian-filter, of size 51×51 and $\sigma = 10$. We use the same padding values as in the zero-padding case, i.e., $D \in [64, 128, 256, 512]$.

Metric: to measure the robustness of the model on multi-scale images, we evaluate the ratio between the accuracy of the model on the scaled validation set and the original set. This metric measures the performance preservation of the network when tested on an out-of-distribution image scale.

Results: we trained a ResNet50 [1] network from scratch on two setups: training using Auto Augment as the only augmentation method and training the network with Auto Augment and InAugment as the augmentation methods. The accuracy preservation of the networks is reported in Figure 1. We also show Grad-CAM [5] visualization for the network in Figure 2. As shown in Table 1, incorporating InAugment in the training process yields a more robust model for all validation sets. It is also evident from Figure 2 that using InAugment yields a model that is insensitive to artifacts. See, for example, in Figure 2a, it is evident that the baseline model associates the padded pixels to be the border of a Television. Furthermore, in Figure 2, we see that using InAugment improves the localization of the model.

3. Code

Our code will be made publicly available and all pre-trained models will be uploaded as well.

References

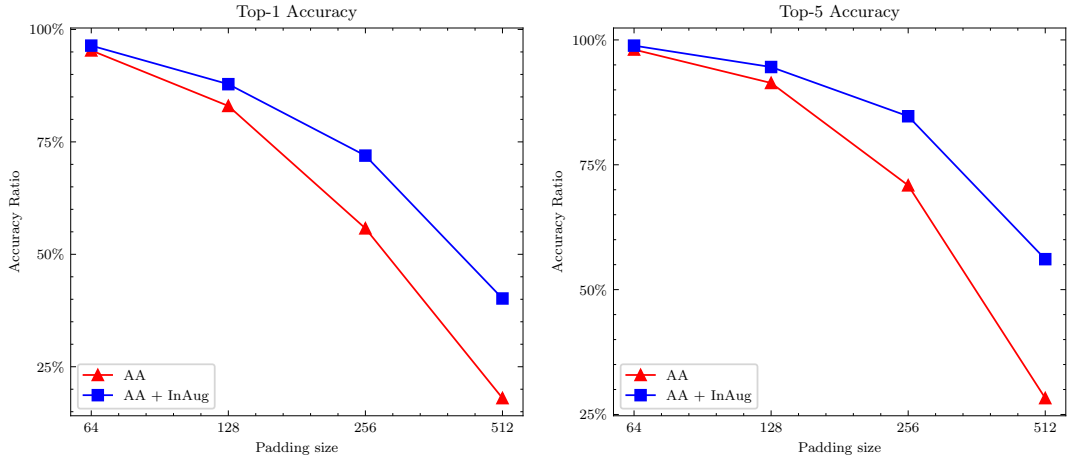
[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 2, 3

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016. 1

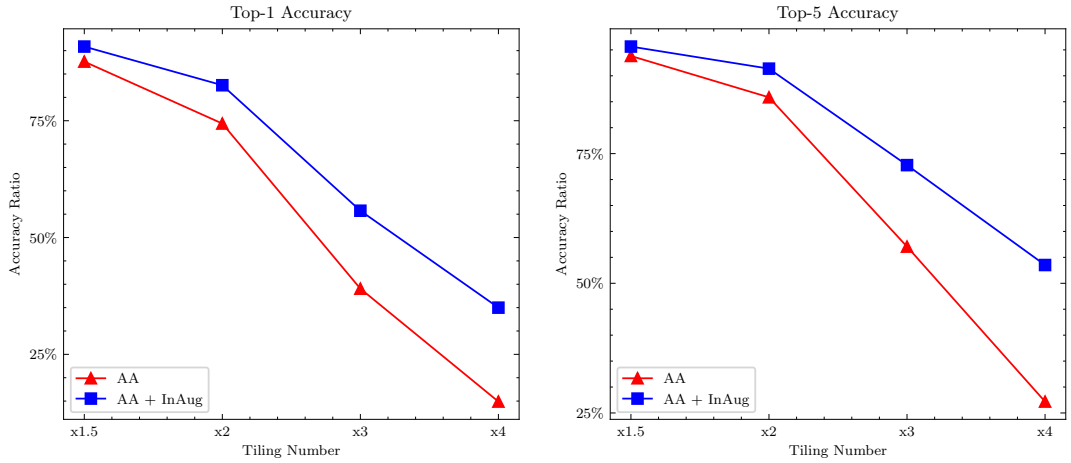
[3] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 1

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2

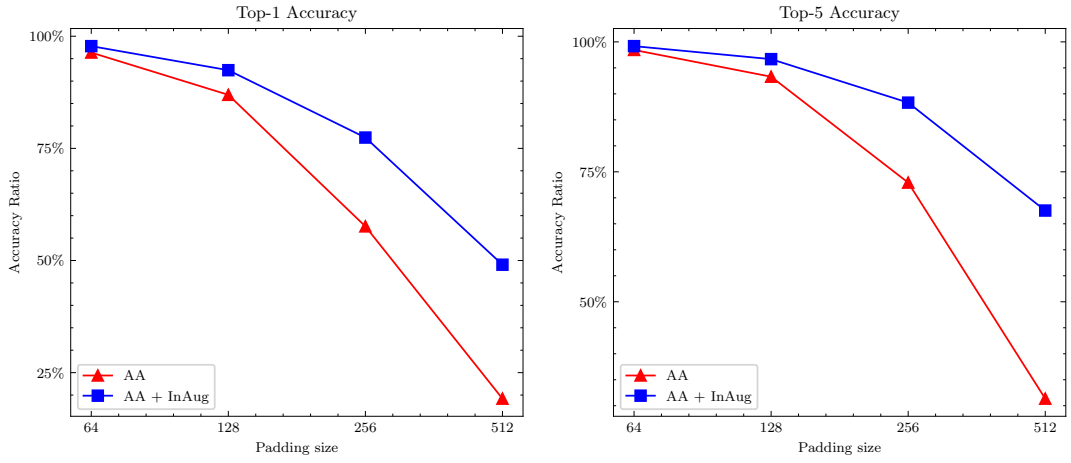
[5] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 618–626. IEEE Computer Society, 2017. 2, 4



(a) Zero-padding

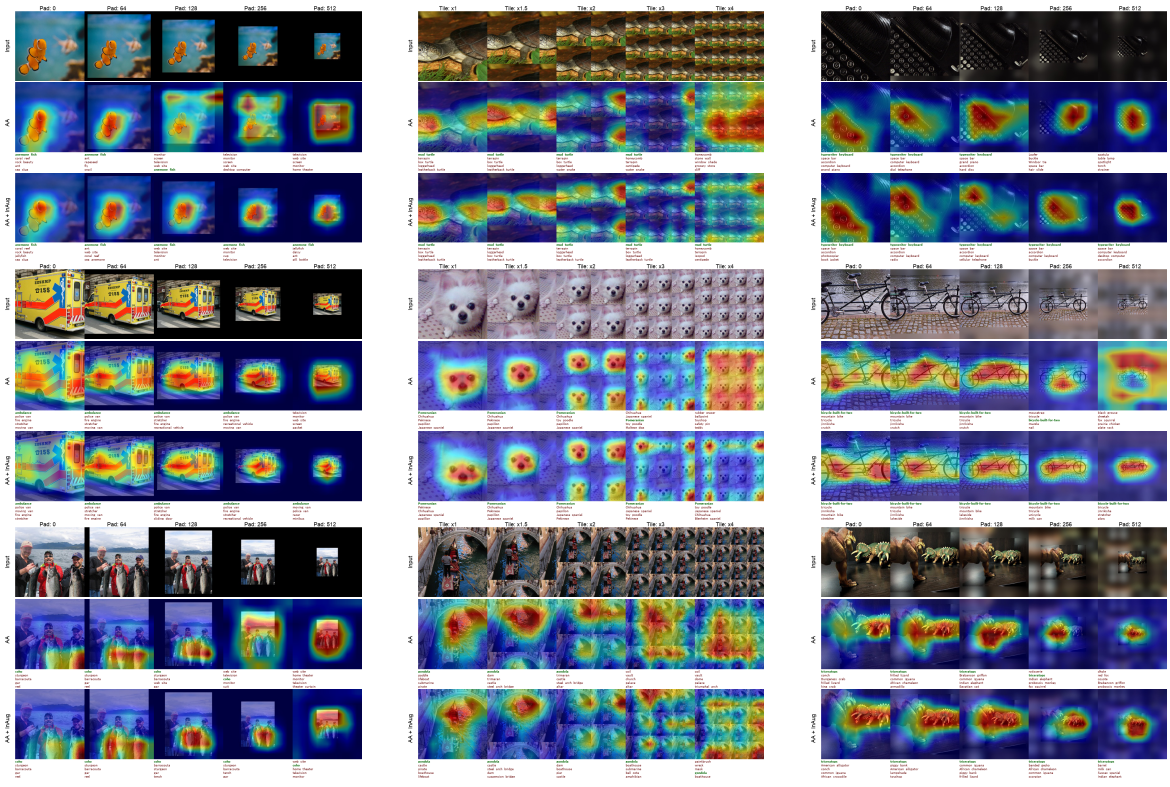


(b) Tile



(c) Symmetric-padding & blur

Figure 1: ResNet50 [1] robustness to scale.



(a) Zero-padding (b) Tiling (c) Symmetric-padding & blur

Figure 2: Grad-CAM [5] visualization for ResNet50 (**best viewed when zoomed-in**).