Pyramid Adversarial Training Improves ViT Performance

SUPPLEMENTARY MATERIALS

Thanks for viewing the supplementary material, in which we provide a detailed explanation of the pyramid structure in Section 1, detailed experiments for different backbones in Section 2, more ablation study in Section 3, additional analysis in Section 4, visualizations in Section 5, and finally a discussion on the effect of optimizers in Section 6.

1. Pyramid Attack Details

In this section, we provide a conceptual description and pseudocode for the pyramid attack.

Description Scale s determines the size of the patch that an individual perturbation parameter will be applied to; e.g. for $s = 16$, we learn and add a single adv parameter to each non-overlapping patch of size 16x16. The application is equivalent to a nearest neighbor resize on the 14x14 adv tensor to the image size of 224x224 and then addition. The scales s and multipliers m_s used by PyramidAT are hyperparameters.

Code We provide a minimal implementation of our technique in Fig. 1.

```
import jax
import jax.numpy as jnp

H=224,lr=1/255, M=[20,10,1], S=[32,16,1], BOUNDS=[0, 1], n_steps=5
def get_attacked_image(model, loss_fn, image):
  def get_perturbed_image(delta):
    return image + sum(
      M[i]*jax.image.resize(delta[i], (H,H,3), 'nearest')
      for i in delta)
  def get_perturbed_loss(delta):
    return loss_fn(model(get_perturbed_image(delta)))
  delta = {i: jnp.zeros((H/s, H/s, 3)) for (i,s) in enumerate(S)}
  for _ in range(n_steps):
    delta_grad = jax.grad(get_perturbed_loss)(delta)
    delta = {i: delta[i]+lr*jnp.sign(delta_grad[i]) for i in delta}
  perturbed_image = get_perturbed_image(delta)
  return jnp.clip(perturbed_image, BOUNDS[0], BOUNDS[1])
```

Figure 1. Implementation of our technique in JAX.

2. Discussing Backbones

2.1. ResNets

Table 1 shows results for multiple variants of ResNet [18]: ResNet-50, ResNet-101, and ResNet-200. As the capacity of the network increases, we observe larger gains from both PixelAT and PyramidAT. PyramidAT performs the best on all evaluation sets.

For these runs, we follow the training protocol and network details set up in [61]. We use the proposed split BN and standard ResNet training: 90 epochs, cosine learning rate at 0.1 with linear warmup for 7 epochs, minimal augmentations (left right flip and Inception crop). For network optimization, we use SGD with momentum and for the adversarial steps, we use SGD.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
ResNet-50	76.70	83.11	4.49	74.90	26.47	64.31	36.24	23.44	6.41
+PixelAT	77.37	84.11	6.03	66.88	27.80	65.59	41.75	27.04	8.13
+PyramidAT	**77.48**	**84.22**	**6.24**	**66.77**	**27.91**	**65.96**	**43.32**	**28.55**	**8.83**
ResNet-101	78.44	84.29	6.16	69.75	29.31	66.54	38.74	26.15	7.19
+PixelAT	79.57	85.73	9.55	59.92	31.00	67.95	44.63	29.83	10.23
+PyramidAT	**79.69**	**85.82**	**9.96**	**59.15**	**31.57**	**68.12**	**46.50**	**31.74**	**10.94**
ResNet-200	79.47	85.25	8.65	65.58	31.86	67.36	40.55	28.21	7.81
+PixelAT	80.51	86.34	12.93	56.99	33.84	69.73	46.32	32.74	9.14
+PyramidAT	**80.92**	**86.59**	**14.17**	**55.72**	**34.04**	**70.08**	**48.05**	**34.54**	**11.48**

Table 1. For all variants of ResNet, PyramidAT leads to improvements. Note that this is with the standard training of ResNet.

2.2. ViT Tiny/16

ViT Ti/16 has the same overall structure and design as ViT B/16 (the primary model used in our main paper) but is significantly smaller, at 5.8 million parameters (as opposed to the 86 million parameters of B/16). More specifically, Ti/16 has a width of 192 (instead of 768), MLP size of 768 (instead of 3072), and 3 heads (instead of 12). In total, this decrease in parameters and model size leads to a substantial decrease in capacity. We experiment with ViT Ti/16 primarily in order to understand the impact of this decreased capacity on our adversarial training methods.

We start with an exploration of the impact of the random augmentation's strength on the overall performance of the model. Table 2 shows the performance of Ti/16 models with different RandAugment parameters (the two parameters are in order the number of transforms applied and the magnitude of the transforms); this table suggests that the network's lower capacity benefits from weaker random augmentation. Specifically, the best RandAugment parameters for the majority of the evaluation datasets is (1,0.8), which is considerably lower than the RandAugment parameters tuned for B/16 (2, 15).

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
RA=(2,10)	61.07	68.77	3.95	85.84	12.88	48.50	21.95	11.21	4.84
RA=(2,5)	64.62	72.57	4.59	80.79	15.44	52.37	25.68	**14.48**	8.36
RA=(1,10)	63.64	71.26	4.64	83.04	14.53	51.37	23.67	13.14	7.27
RA=(1,5)	64.96	72.54	**4.80**	81.32	14.94	52.05	25.03	13.69	**8.98**
RA=(1,3)	64.88	72.66	4.80	79.04	15.61	52.59	25.43	13.54	8.13
RA=(1,0.8)	**65.33**	**73.19**	4.79	**77.08**	**16.16**	**53.03**	**25.98**	14.15	8.98
RA=(1,0.4)	64.27	72.17	4.69	78.10	15.46	52.18	24.99	13.47	8.59
RA=(1,0.1)	63.58	71.41	4.80	79.23	15.39	51.43	23.66	12.54	8.36

Table 2. ViT Ti/16 baseline training with different random augmentations. In this table, RA denotes the two RandAugment parameters: number of applied transforms, and mangitude of transforms. Note that weaker augmentation tends to perform better.

In Table 3 and 4, we pick several of the better performing RandAugment parameters and then show results from adversarial training with steps of 1 and 3, respectively.

Table 3 shows that the performance of adversarial training depends heavily on both the random augmentation and the type of attack. Note, RAm refers to the RandAugment mangitude parameter. As shown by RAm=0.1, pixel attacks can improve performance for in-distribution evaluation datasets when the random augmentation strength is low. However, at higher random augmentation, RAm=0.4 and RAm=0.8, PixelAT leads to the commonly observed trade-off between clean performance and adversarial robustness. In contrast, pyramid tends to improve performance across the board regardless of the starting augmentation (for all RAm of 0.1, 0.4, and 0.8). Interestingly, PixelAT exhibits better robustness properties (out-of-distribution performance) than PyramidAT for Ti/16. We hypothesize that the limited capacity can be "spent" on either in-distribution or out-of-distribution representations and that pyramid tends to bias the network towards in-distribution as opposed to pixel which has a bias towards out-of-distribution.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Ti/16 RAm=0.1	63.58	71.41	4.80	79.23	15.39	51.43	23.66	12.54	8.36
+PixelAT steps=1	64.66	72.75	4.39	74.54	14.61	52.05	**32.52**	**17.65**	**14.77**
+PyramidAT steps=1	**65.49**	**73.53**	**5.16**	**74.30**	**16.15**	**53.08**	29.18	16.55	12.58
Ti/16 RAm=0.4	64.27	72.17	4.69	78.10	15.46	52.18	24.99	13.47	8.59
+PixelAT steps=1	62.78	70.53	4.05	77.67	13.89	50.37	**29.75**	**16.35**	11.80
+PyramidAT steps=1	**65.61**	**73.68**	**4.80**	**74.72**	**15.97**	**52.88**	28.89	16.14	**11.95**
Ti/16 RAm=0.8	65.33	73.19	4.79	77.08	**16.16**	53.03	25.98	14.15	8.98
+PixelAT steps=1	63.49	71.64	3.80	75.68	13.79	51.13	**31.80**	**17.65**	**13.91**
+PyramidAT steps=1	**65.67**	**73.73**	**4.84**	**75.25**	15.71	**53.43**	29.08	16.41	12.97

Table 3. ViT Ti/16 adversarial training experiments with steps=1. RAm gives the RandAugment magnitude parameter; all experiments have RandAugment number of transforms equal to 1. Pixel's performance on 0.4 and 0.8 is consistent with earlier work that suggests that adversarial training causes a trade-off between in distribution and out of distribution datasets. However, we show that a low random augmentation starting point can break this trade-off and lead to gains. Pyramid tends to outperform pixel on in-distribution performance for all random augmentations. However, pixel performs well for out-of-distribution datasets.

Table 4 shows that the strength of the adversarial attack also matters substantially to the overall performance of the model. Both attacks, pixel and pyramid, with 3 steps tend to degrade the model's performance on in-distribution evaluation datasets. Adversarial training still provides some benefits for out-of-distribution, with pyramid performing the best in terms of robustness. We hypothesize that pyramid outperforms pixel in the steps=3 runs because pyramid is a weaker out-of-distribution augmentation and pixel at 3 steps has over-regularized the network, leading to decreased performance. Note that pyramid at steps=3 produces the best out-of-distribution performance out of any Ti/16 runs including strong random augmentation and PixelAT at steps=1.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Ti/16 RAm=0.1	**63.58**	**71.41**	**4.80**	79.23	**15.39**	**51.43**	23.66	12.54	8.36
+PixelAT steps=3	59.91	67.77	3.39	81.44	12.04	47.28	28.52	14.78	11.02
+PyramidAT steps=3	62.71	71.08	4.08	**74.49**	14.72	50.25	**35.05**	**20.67**	**18.67**
Ti/16 RAm=0.8	**65.33**	**73.19**	**4.79**	77.08	**16.16**	**53.03**	25.98	14.15	8.98
+PixelAT steps=3	60.10	67.88	3.36	80.48	11.90	47.78	29.83	15.21	13.44
+PyramidAT steps=3	62.92	71.11	4.05	**74.70**	14.76	50.77	**34.74**	**20.35**	**18.13**

Table 4. ViT Ti/16 adversarial training experiments with steps=3. RAm gives the RandAugment magnitude parameter; all experiments have RandAugment number of transforms equal to 1. All techniques degrade from the baseline suggesting that 3 adversarial steps produces augmentation that is too strong for Ti's capacity.

2.3. MLP-Mixer

As shown in the main paper, we observe gains across the board for MLP-Mixer with PyramidAT. Here, we show that the gain is robust to a change in the LR schedule and that the gain is, again, affected by the starting augmentation.

Table 5 shows baseline and adversarially trained models for two different training schedules of MLP-Mixer, one with the default LR schedule of 10k warm-up steps and then linear decay to an end learning rate (LR) of $1e-5$ and another with a more aggressive end learning rate of $1e-7$. We show that this change in LR schedule does not affect the gains from adversarial training.

Table 6 shows that, similar to ViT Ti/16, the gains are improved when the random augmentation is weakened. However, in this case, the gain is not enough to overcome the drop in performance from using the weaker augmentation.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
MLP-Mixer [54] end LR=1e-5 (default)	78.27	83.64	10.84	58.50	25.90	64.97	38.51	29.00	10.08
+PixelAT	77.17	82.99	9.93	57.68	24.75	64.03	44.43	33.68	**15.31**
+PyramidAT	**79.29**	**84.78**	**12.97**	**52.88**	**28.60**	**66.56**	**45.34**	**34.79**	14.77
MLP-Mixer [54] end LR=1e-7	75.92	81.28	9.45	64.29	22.13	62.17	33.70	25.15	7.27
+PixelAT	74.96	80.81	7.81	61.85	20.87	60.94	39.82	28.59	12.27
+PyramidAT	**77.98**	**83.60**	**11.17**	**56.19**	**25.59**	**64.92**	**41.65**	**31.99**	**12.66**

Table 5. MLP-Mixer ablations with different training. The pyramid gains are preserved even with different training schedules.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
MLP-Mixer [54] end LR=1e-7, RA=(2,15)	75.92	81.28	9.45	64.29	22.13	62.17	33.70	25.15	7.27
+PixelAT	74.96	80.81	7.81	61.85	20.87	60.94	39.82	28.59	12.27
+PyramidAT	**77.98**	**83.60**	**11.17**	**56.19**	**25.59**	**64.92**	**41.65**	**31.99**	**12.66**
MLP-Mixer [54] end LR=1e-7, RA=(2,9)	73.56	79.01	7.79	67.18	18.09	60.30	31.47	22.80	7.58
+PixelAT	72.44	78.27	7.80	63.38	16.96	58.40	37.18	25.78	12.27
+PyramidAT	**76.19**	**81.47**	**10.61**	**57.71**	**21.86**	**63.03**	**39.52**	**29.40**	**14.37**

Table 6. MLP-Mixer ablations with random augmentation magnitude. Weaker random augmentations lead to larger gains from the adversarial training but with these parameters do not lead to better overall performance than the strong random augmentation plus adversarial training. Note that for both the starting point and weaker random augmentation, the gain from pyramid is substantial compared to the gain from pixel.

3. Additional Ablations

3.1. Dropout

One of the key findings of this paper is the importance of "matched" Dropout [50] and stochastic depth [23]. Here we describe numerous ablations on these Dropout terms and list several detailed findings including:

- Matching the Dropout and stochastic depth matters significantly for balanced clean performance and robustness.
- Running without Dropout in the adversarial training branch can improve robustness even more.
- Dropout matters more than Stochastic Depth

Note that in the tables below, we use the term "dropparams" to refer to a tuple of the Dropout probability and stochastic depth probability. Clean dropparams (abbreviated as c_dp) refer to the dropparams used for the clean training branch; adversarial dropparams (abbreviated as a_dp) refer to the dropparams used for the adversarial training branch; and matched dropparams (abbreviated as m_dp) refer to dropparams used for both clean and adversarial branches. So c_dp $= (10, 0)$ means that the clean training branch had a 10% probability of Dropout but a 0% probability of stochastic depth.

Table 7 explores different possible values for adversarial dropparams. In general, lower values of Dropout and stochastic depth in the adversarial branch improve out-of-distribution performance while hurting in-distribution performance; however, the opposite is not true: higher levels of Dropout and stochastic depth in the adversarial branch do not improve in-distribution performance. In-distribution performance seems to peak when the params for the adversarial and clean branches match.

Table 8 explores different possible values for matched dropparams. In general, the dropparams determined by RegViT [51] seem to be roughly optimal for both the baselines and the adversarially trained models, with some variation for some datasets.

Table 9 explores if one of these parameters is more important than the others. To do so, we set clean dropparams to $(10, 10)$ for the entire table (besides the included baselines) and only vary the adversarial dropparams. For both PixelAT and PyramidAT, the Dropout parameter seems to be more important for clean, in-distribution performance. Without Dropout, the top-1 of ImageNet drops 0.41 for PixelAT and 0.92 for PyramidAT. However, no Dropout does give a substantial boost to out-of-distribution performance, with Rendition gains of 11.59 for PixelAT and 15.68 for PyramidAT and Sketch gains of 7.49 for PixelAT and 11.96 for PyramidAT. Without stochastic depth, the adversarially trained models seem to perform

Method	ImageNet	Real	A	C↓	Out of Distribution Robustness Test ObjectNet	V2	Rendition	Sketch	Stylized
Baseline c_dp $= (10, 10)$	79.92	85.14	17.48	52.46	29.30	67.49	38.24	29.08	11.02
PixelAT a_dp $= (0, 0)$	79.13	84.75	14.37	52.73	29.01	66.94	**49.94**	**37.03**	**22.34**
PixelAT a_dp $= (5, 5)$	78.41	84.39	14.00	54.50	28.04	66.30	49.29	36.84	21.80
PixelAT a_dp $= (10, 10)$	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
PixelAT a_dp $= (20, 20)$	**81.00**	**86.16**	**21.43**	**46.07**	**30.85**	**69.60**	46.00	34.73	18.36
PixelAT a_dp $= (30, 30)$	76.53	82.56	16.20	66.09	24.22	64.40	42.93	34.89	15.00
Baseline c_dp $= (10, 10)$	79.92	85.14	17.48	52.46	29.30	67.49	38.24	29.08	11.02
PyramidAT a_dp $= (0, 0)$	79.31	85.13	13.95	55.09	29.98	67.43	**53.69**	**41.07**	23.44
PyramidAT a_dp $= (5, 5)$	79.13	84.90	13.43	54.08	28.99	67.50	52.39	40.23	**24.84**
PyramidAT a_dp $= (10, 10)$	**81.71**	**86.82**	22.99	**44.99**	**32.92**	**70.82**	47.66	36.77	19.14
PyramidAT a_dp $= (20, 20)$	81.67	86.76	**23.33**	45.19	32.74	70.58	48.15	38.01	17.50
PyramidAT a_dp $= (30, 30)$	74.26	80.57	11.55	69.85	22.87	61.46	41.59	29.89	17.66

Table 7. Ablation on the values of Dropout and stochastic depth for adversarial training branch. All c_dp are kept constant at $(10, 10)$ in the adversarial section.

Method	ImageNet	Real	A	C↓	Out of Distribution Robustness Test ObjectNet	V2	Rendition	Sketch	Stylized
Baseline c_dp $= (0, 0)$	75.69	80.92	12.68	61.78	23.65	62.49	34.23	24.38	7.73
Baseline c_dp $= (5, 5)$	79.05	84.19	16.52	54.32	28.27	66.40	37.38	28.08	9.53
Baseline c_dp $= (10, 10)$	**79.92**	**85.14**	17.48	**52.46**	**29.30**	**67.49**	38.24	**29.08**	11.02
Baseline c_dp $= (15, 15)$	79.35	84.74	**17.71**	52.96	29.11	67.47	**39.13**	28.67	**11.41**
Baseline c_dp $= (20, 20)$	78.40	84.24	16.16	54.93	27.80	66.07	37.85	27.77	9.84
PixelAT m_dp $= (0, 0)$	79.13	84.20	19.71	49.68	29.77	67.02	43.34	32.14	15.23
PixelAT m_dp $= (5, 5)$	**80.82**	85.59	**22.12**	**46.15**	**31.27**	**68.99**	44.86	33.31	16.95
PixelAT m_dp $= (10, 10)$	80.42	**85.78**	19.15	47.68	30.11	68.78	45.39	**34.40**	18.28
PixelAT m_dp $= (15, 15)$	79.03	84.88	15.16	50.67	27.72	66.53	**46.43**	34.21	**21.95**
PixelAT m_dp $= (20, 20)$	77.89	84.12	13.99	52.20	26.28	66.06	45.21	32.75	21.95
PyramidAT m_dp $= (0, 0)$	79.54	84.66	18.91	50.02	30.10	67.44	44.43	33.46	13.67
PyramidAT m_dp $= (5, 5)$	**81.80**	86.59	**23.47**	45.52	32.76	70.36	46.62	36.50	17.27
PyramidAT m_dp $= (10, 10)$	81.71	**86.82**	22.99	**44.99**	**32.92**	**70.82**	**47.66**	36.77	**19.14**
PyramidAT m_dp $= (15, 15)$	80.95	86.49	21.33	46.18	31.62	69.70	47.21	**36.83**	18.91
PyramidAT m_dp $= (20, 20)$	79.56	85.77	18.35	49.14	29.73	68.13	45.72	35.41	18.36

Table 8. Ablation on the values of Dropout and stochatic depth for matched attacks.

roughly as well as with stochastic depth, exhibitly marginally more clean accuracy for PyramidAT than the model with both Dropout and stochastic depth. Our main takeaway is that Dropout seems to be the primary determinant in whether the gains are balanced between in-distribution and out-of-distribution or primarily focused on out-of-distribution. In fact, no Dropout PyramidAT performs so well on out-of-distribution that it sets new state-of-the-art numbers for Rendition and Sketch.

Table 10 explores parameter settings where the Dropout and stochastic depth are not equal. In general, there does not seem to be a consistent trend or recognizable pattern for the overall performance, though some patterns exist for specific attacks and evaluation datasets. For example, increasing stochastic depth probability for pixel attacks tend to improve Real, ImageNet-C, and ObjectNet performance.

Table 11 explores the effects of adversarial training with different dropparams without Dropout or stochastic depth in the main branch. In general, the lack of Dropout and stochastic depth in the clean branch has a substantial negative effect on the performance of the model and all of the resulting models under-perform their counterparts with non-zero clean dropparams. In this setting, adversarial training does provide substantial improvements for both in-distribution ($+5.18$ for clean using PyramidAT) and out-of-distribution performances ($+12.29$ for Rendition using PyramidAT and $+11.68$ for Sketch using PyramidAT), but not enough to offset the poor starting performance of the baseline model.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Baseline c_dp $= (0, 0)$	75.69	80.92	12.68	61.78	23.65	62.49	34.23	24.38	7.73
Baseline c_dp $= (10, 10)$	**79.92**	**85.14**	**17.48**	**52.46**	**29.30**	**67.49**	**38.24**	**29.08**	**11.02**
PixelAT a_dp $= (0, 10)$	79.40	84.98	14.32	52.17	28.87	67.24	**49.83**	**36.57**	**21.88**
PixelAT a_dp $= (10, 0)$	79.51	85.12	15.65	49.38	29.45	67.50	47.32	34.64	21.17
PixelAT a_dp $= (10, 10)$	**80.42**	**85.78**	**19.15**	**47.68**	**30.11**	**68.78**	45.39	34.40	18.28
PyramidAT a_dp $= (0, 10)$	79.00	85.02	13.69	55.58	29.38	66.96	**53.92**	**41.04**	**24.22**
PyramidAT a_dp $= (10, 0)$	**81.80**	86.67	**23.51**	45.00	**33.37**	70.52	47.82	37.09	19.38
PyramidAT a_dp $= (10, 10)$	81.71	**86.82**	22.99	**44.99**	32.92	**70.82**	47.66	36.77	19.14

Table 9. Ablation on the values of Dropout and stochastic depth for unmatched attacks. For the adversarial techniques, clean dropparams will be the same as RegVit at (10, 10). For the adversarial training rows, either Dropout or stochastic depth will be 0 and the other will be the base value. This table explores whether one of these parameters is more important than the other. For both PixelAT and PyramidAT, Dropout appears to be more important in determining the balance between in-distribution and out-of-distribution performance. PyramidAT no Dropout is SOTA for Rendition and Sketch.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Baseline c_dp $= (10, 10)$	**79.92**	**85.14**	**17.48**	**52.46**	**29.30**	**67.49**	**38.24**	**29.08**	**11.02**
PixelAT a_dp $= (0, 10)$	79.40	84.98	14.32	52.17	28.87	67.24	**49.83**	**36.57**	**21.88**
PixelAT a_dp $= (10, 10)$	**80.42**	**85.78**	19.15	**47.68**	**30.11**	**68.78**	45.39	34.40	18.28
PixelAT a_dp $= (20, 10)$	74.58	80.74	13.07	69.14	23.93	61.82	42.64	33.54	17.42
PixelAT a_dp $= (30, 10)$	80.14	85.59	**20.91**	48.99	29.74	68.50	44.62	35.06	16.02
PixelAT a_dp $= (10, 0)$	79.51	85.12	15.65	49.38	29.45	67.50	**47.32**	34.64	**21.17**
PixelAT a_dp $= (10, 10)$	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
PixelAT a_dp $= (10, 20)$	**81.14**	86.22	**21.37**	45.91	31.53	**69.75**	46.24	**35.14**	19.45
PixelAT a_dp $= (10, 30)$	80.94	**86.26**	21.37	**45.19**	**31.63**	69.69	45.76	34.69	19.61
PyramidAT a_dp $= (0, 10)$	79.00	85.02	13.69	55.58	29.38	66.96	**53.92**	**41.04**	**24.22**
PyramidAT a_dp $= (10, 10)$	**81.71**	**86.82**	**22.99**	**44.99**	**32.92**	**70.82**	47.66	36.77	19.14
PyramidAT a_dp $= (20, 10)$	75.36	81.40	14.03	70.76	23.65	63.14	39.36	26.92	15.00
PyramidAT a_dp $= (30, 10)$	79.18	84.91	17.43	56.44	28.95	67.81	43.64	29.91	19.45
PyramidAT a_dp $= (10, 0)$	**81.80**	86.67	**23.51**	45.00	**33.37**	70.52	47.82	37.09	**19.38**
PyramidAT a_dp $= (10, 10)$	81.71	**86.82**	22.99	**44.99**	32.92	**70.82**	47.66	36.77	19.14
PyramidAT a_dp $= (10, 20)$	81.50	86.58	23.13	45.72	32.45	70.48	**48.08**	36.91	17.66
PyramidAT a_dp $= (10, 30)$	81.53	86.75	21.87	45.59	32.80	70.34	47.73	**37.12**	17.89

Table 10. Ablation on settings where the Dropout parameter and stochastic depth parameter are not equal for adversarial training branch. All c_dp are kept constant at $(10, 10)$ in the adversarial section.

3.2. Pyramid Structure

In the main paper, Table 8 presented an abridged version (with only a subset of the evaluation datasets) of an ablation on the structure of the pyramid used in the pyramid adversarial training. We present the full version (complete with all the evaluation datasets) of this ablation in Table 12. This table remains consistent with the description and explanation in the main table: adding more layers to the pyramid tends to improve performance. In fact, Table 12 shows the full extent of the trade-off between the 3rd and 4th levels of the pyramid. Specifically, the 4th level seems to lead to a slight improvement in out-of-distribution performance and a slight decline in in-distribution performance. Note that 2-level Pyramid is simply the combination of Pixel and Patch.

Using the the scale notation established in 3.2 Pyramid Adversarial Training, the details of these layers are as follows in Table 13.

In Table 14, we explore different magnitudes for the patch level. We note that some of the gains from 2-level are from the

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Baseline c_dp $= (0, 0)$	75.69	80.92	12.68	61.78	23.65	62.49	34.23	24.38	7.73
PixelAT c_dp $= (0, 0)$ a_dp $= (5, 5)$	79.86	84.78	21.23	48.66	30.19	67.92	**43.94**	32.69	**15.00**
PixelAT c_dp $= (0, 0)$ a_dp $= (10, 10)$	79.82	84.81	21.32	**47.82**	30.84	67.99	43.25	32.24	14.92
PixelAT c_dp $= (0, 0)$ a_dp $= (20, 20)$	**80.03**	**84.96**	**21.69**	47.86	**31.34**	**68.16**	43.56	**32.78**	14.77
Baseline c_dp $= (0, 0)$	75.69	80.92	12.68	61.78	23.65	62.49	34.23	24.38	7.73
PyramidAT c_dp $= (0, 0)$ a_dp $= (5, 5)$	80.79	85.66	**23.65**	47.39	32.39	69.12	46.30	**36.40**	15.70
PyramidAT c_dp $= (0, 0)$ a_dp $= (10, 10)$	80.69	85.55	23.25	47.14	**32.74**	69.21	46.49	36.09	15.78
PyramidAT c_dp $= (0, 0)$ a_dp $= (15, 15)$	**80.87**	**85.74**	23.63	**46.68**	32.67	**69.27**	**46.52**	36.06	**17.19**

Table 11. Ablation on the values of Dropout and stochastic depth for adversarial training branch for a clean branch with no Dropout or stochastic depth; specifically, all c_dp are kept constant at $(0, 0)$. The loss of Dropout and stochastic depth causes poor performance across the board.

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Pixel	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
Patch	81.20	86.10	21.33	50.30	31.87	68.98	42.87	33.75	15.00
2-level Pyramid	81.65	86.69	22.79	45.27	32.46	69.86	47.00	36.71	19.06
3-level Pyramid	**81.71**	**86.82**	22.99	**44.99**	**32.92**	**70.82**	47.66	36.77	19.14
4-level Pyramid	81.66	86.68	**23.21**	45.29	32.85	70.56	**47.68**	**37.41**	**20.47**

Table 12. Pyramid structure ablations. This shows the effect of the number of layers of the pyramid. Adding coarser layers with larger magnitudes generally improves performance.

Layer	Name	s	m_s
Layer 1	Pixel	1	1
Layer 2	Patch	16	10
Layer 3	2×2 patches	32	20
Layer 4	Global	224	25

Table 13. Pyramid details.

higher magnitude for the coarse level.

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Pixel $m = 1$	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
Patch $m = 1$	80.09	85.09	18.40	52.17	29.78	68.09	40.46	30.46	13.83
Patch $m = 10$	80.95	85.77	20.01	50.62	31.37	68.86	42.46	32.65	11.64
Patch $m = 20$	81.20	86.10	21.33	50.30	31.87	68.98	42.87	33.75	15.00
2-level Pyramid $m = [10, 1]$	**81.65**	**86.69**	**22.79**	**45.27**	**32.46**	**69.86**	**47.00**	**36.71**	**19.06**

Table 14. Pyramid structure ablations where m is the multiplicative term of the perturbation. Shows that the combination of patch and pixel is better than only patch, even when patch is tested at different magnitudes.

We additionally include Table 15 which shows a random subset of pyramid structures tested. The best pyramids tend to be structured based on the patches of the ViT.

3.3. More epochs for baseline

We tested the effect of additional epochs for the baseline training. We found that going from 300 epochs to 500 (with the learning rate being adjusted accordingly) did not provide any benefits to the network's performance. In fact, Table 16 shows

Scale factor	Strengths	ImageNet [12]	Real [12]	A [22]	C [20]↓	Rendition [19]	Sketch [16]	Stylized [58]
$[224, 16, 1]$	$[20, 10, 1]$	81.37	86.50	21.65	45.84	46.72	36.33	17.97
$[32, 16, 1]$	$[20, 10, 1]$	**81.71**	**86.82**	**22.99**	**44.99**	**47.66**	36.77	**19.14**
$[224, 32, 16, 1]$	$[20, 10, 5, 1]$	81.49	86.66	21.93	45.89	47.08	37.22	18.98
$[16, 1]$	$[10, 1]$	81.65	86.69	22.79	45.27	47.00	36.71	19.06
$[16, 4, 1]$	$[20, 10, 1]$	81.43	86.59	21.83	49.15	47.49	**37.85**	17.19

Table 15. Random set of pyramid configurations.

that the longer run performs worse in most evaluation datasets than the shorter run.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Baseline 300 epoch.	**79.92**	**85.14**	17.48	**52.46**	**29.30**	**67.49**	**38.24**	**29.08**	**11.02**
Baseline 500 epoch.	79.34	78.83	**18.39**	54.28	28.43	66.69	37.64	27.60	10.31

Table 16. Exploration of the number of steps for the baseline.

3.4. Number of Attack Steps

We perform an ablation on the number of steps in the adversarial attack. AdvProp [61] uses 5 for their main paper; we also adopt this parameter as a reasonable balance between performance and train time (each additional step in the attack requires a forward and backward pass of the model and increases the train time accordingly). Table 17 shows that higher number of steps tends to lead to better performance for both pixel and pyramid.

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
PixelAT steps=1	80.46	85.48	17.96	49.59	30.12	68.66	43.31	32.12	13.59
PixelAT steps=3	80.39	85.62	17.71	48.50	29.67	68.58	**46.56**	33.84	17.50
PixelAT steps=5	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
PixelAT steps=7	**80.77**	**86.03**	**20.44**	**46.31**	**31.12**	**69.25**	45.95	**34.54**	**18.59**
PyramidAT steps=1	79.93	84.89	18.17	50.92	28.91	68.13	40.50	30.48	12.81
PyramidAT steps=3	81.47	86.46	22.39	46.21	32.33	70.11	45.07	34.89	18.20
PyramidAT steps=5	**81.71**	**86.82**	22.99	**44.99**	**32.92**	**70.82**	**47.66**	**36.77**	**19.14**
PyramidAT steps=7	81.65	86.69	**23.63**	45.33	32.53	70.47	47.29	36.52	16.95

Table 17. Ablation on the number of steps in the adversarial attack. For both PixelAT and PyramidAT, larger number of steps tend to give higher performance. Note that increasing the number of steps also increases the train time. We chose 5 for both PixelAT and PyramidAT as a reasonable tradeoff between performance and train time.

3.5. Magnitude

We also perform ablations on the magnitude of perturbations (specifically L2 of the difference between adversarial image and the original image) and show that there exists an inverted U curve for both PixelAT and PyramidAT where one perturbation setting tends to produce the best model for most evaluation datasets.

For PixelAT, we change the perturbation magnitude by editing the learning rate (lr) and the epsilon parameter (ϵ) which is used for the clipping function. Since we use the SGD optimizer, a larger learning rate and epsilon will naturally lead to larger perturbations. Table 18 shows the results of these experiments, which suggests that pixel attacks can very quickly become too large to help the overall network performance.

For PyramidAT, we adjust the perturbation size by editing the magnitude of the multiplicative terms. In Table 19, we perform an exhaustive sweep of these terms starting with an initial list of $[20, 10, 1]$ and multiplying the list by a constant.

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Baseline	79.92	85.14	17.48	52.46	29.30	67.49	38.24	29.08	11.02
PixelAT lr $= 5/255, \epsilon = 20/255$	**81.25**	**86.61**	**21.59**	**44.07**	32.06	**69.75**	**47.59**	**37.29**	**20.39**
PixelAT lr $= 10/255, \epsilon = 40/255$	80.93	86.35	21.43	45.58	**32.88**	69.57	45.40	35.01	18.44
PixelAT lr $= 20/255, \epsilon = 80/255$	80.59	85.90	20.31	46.11	30.83	69.05	45.08	33.39	18.28
PixelAT lr $= 40/255, \epsilon = 160/255$	80.27	85.59	18.95	48.61	30.34	68.40	42.12	32.97	15.16

Table 18. Ablation on the magnitude of PixelAT. PixelAT tends to degrade with higher lr and ϵ.

This table shows that there also exists an inverted U curve where the performance will degrade if the perturbation magnitude is either too small or too big.

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Baseline	79.92	85.14	17.48	52.46	29.30	67.49	38.24	29.08	11.02
PyramidAT $m = [1, 0.5, 0.05]$	80.94	85.71	19.29	49.97	30.57	68.98	43.38	32.99	13.52
PyramidAT $m = [2, 1, 0.1]$	80.94	85.71	19.29	49.97	30.57	68.98	43.38	32.99	13.52
PyramidAT $m = [4, 2, 0.2]$	80.94	85.71	19.29	49.97	30.57	68.98	43.38	32.99	13.52
PyramidAT $m = [8, 4, 0.4]$	80.73	86.02	20.20	47.20	31.36	69.08	44.10	33.48	18.83
PyramidAT $m = [10, 5, 0.5]$	80.94	85.71	19.29	49.97	30.57	68.98	43.38	32.99	13.52
PyramidAT $m = [12, 6, 0.6]$	80.47	85.83	19.32	47.48	30.86	68.97	44.10	33.46	17.03
PyramidAT $m = [20, 10, 1]$	**81.71**	**86.82**	22.99	44.99	32.92	**70.82**	47.66	36.77	**19.14**
PyramidAT $m = [22, 11, 1.1]$	81.71	86.70	**23.55**	**44.84**	**32.98**	70.57	**47.81**	**37.93**	18.59
PyramidAT $m = [24, 12, 1.2]$	81.19	86.40	20.63	46.78	31.59	69.27	45.48	35.25	16.72
PyramidAT $m = [28, 14, 1.4]$	81.14	86.31	20.68	46.71	30.99	69.61	46.23	35.86	17.50

Table 19. Ablation on the magnitude of the pyramid adversarial training.

4. Additional Analysis

4.1. Positional embedding

In Table 20, we explore training on a ViT model without the positional embedding in order to understand the effects of the PixelAT and PyramidAT. We observe that without the positional embedding, PixelAT and PyramidAT tend to perform similarly; in fact, the gap between PixelAT and PyramidAT for clean ImageNet decreases from 1.29 with the positional embedding to 0.17 without the positional embedding. This suggests that much of the improvements for in-distribution performance come from improved training of the positional embedding. However, even without the positional embedding, we observe improvements in the out-of-distribution datasets; e.g. going from pixel to pyramid results in a gain of 2.27 on Rendition and 2.37 on Sketch with the positional embedding and slightly smaller gains of 1.27 and 1.43 without the positional embedding. This suggests that PyramidAT is still improving the learned features used for out-of-distribution performance.

Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
			Out of Distribution Robustness Test						
Baseline with positional embedding	79.92	85.14	17.48	52.46	29.30	67.49	38.24	29.08	11.02
Pixel with positional embedding	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
Pyramid with positional embedding	**81.71**	**86.82**	**22.99**	**44.99**	**32.92**	**70.82**	**47.66**	**36.77**	**19.14**
Baseline without positional embedding	76.58	82.04	12.76	63.56	24.15	63.26	27.99	15.79	6.25
Pixel without positional embedding	78.30	83.85	14.52	**56.86**	26.19	65.50	33.09	19.93	**9.92**
Pyramid without positional embedding	**78.47**	**84.28**	**14.79**	57.11	**27.28**	**66.21**	**34.46**	**21.36**	9.38

Table 20. Analysis of the effect of adversarial training on a ViT without positional embedding. We observe that without the positional embedding, PixelAT and PyramidAT tend to perform similarly for many of the evaluation datasets.

4.2. Optimizing each level individually

In the pyramid attack, the different multiplicative magnitudes for each level mean that each level's parameter takes different sized steps; for example, with the default settings, a change of 1 in the patch parameter leads to a change of 10 on the final image, whereas a change of 1 in the pixel parameter leads to a change of 1. Here, we attempt to understand whether the gradients for the different levels of the pyramid can be informative in the presence of each other; specifically, if the patch level makes a step of 10 in one direction, will this invalidate the gradient in the pixel level which only makes a step of 1. To do this, we experiment with running each level of the pyramid separately, going from coarse to fine: for a given k, we run k steps of only the coarsest level, k steps of only the next coarsest level, etc. In this experiment, we try to keep the amount of training time roughly equal and select k so that the sum of k on each level is roughly equal to the steps taken in the pyramid method in the main paper (5). Table 21 shows the results from this experiment and suggests that the gradients from each individual level are still useful when combined and that separating this optimization does not in fact lead to performance improvements; note that $k = 2$ leads to more overall optimization steps (6 total steps) than the main technique (5 total steps).

			Out of Distribution Robustness Test						
Method	ImageNet	Real	A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
Pyramid Separate $k = 1$ (3 total)	81.17	86.06	19.41	49.10	30.35	69.27	42.94	33.41	15.55
Pyramid Separate $k = 2$ (6 total)	81.36	86.36	22.31	47.73	32.12	69.92	46.13	35.66	15.31
Pyramid (5 steps total)	**81.71**	**86.82**	**22.99**	**44.99**	**32.92**	**70.82**	**47.66**	**36.77**	**19.14**

Table 21. Ablation on CoarseThenFine. The separated gradients do not seem strictly better than simply running all levels at once.

4.3. Evaluation of white-box attacks

We evaluate the performance of the B/16 baseline, PixelAT, and PyramidAT models against pixel and pyramid PGD attacks. The results are given in table 22. Both adversarially trained models give the best performance when attacked in the setting in which they were trained. PyramidAT provides comparably more protection against pixel attacks (48.8% Top-1) than PixelAT against pyramid attacks (43.0% Top-1).

Method	Pixel PGD	Pyramid PGD
Baseline	13.7	26.1
+PixelAT	53.6	43.0
+PyramidAT	48.8	68.7

Table 22. Top-1 accuracy of 3 models (baseline, PixelAT, PyramidAT) against 2 types of white-box adversarial attacks (Pixel PGD, Pyramid PGD, both at 5 steps).

5. Additional Visualizations

5.1. Pixel attacks

In Figure 2, we include 4 additional visualizations of pixel attacks against the baseline and PixelAT models. Some structure is visible in the PixelAT model. Note that for pixel attacks, we would expect more structure to appear in the PixelAT model than the PyramidAT model since the attack is in-distribution for the PixelAT model but out-of-distribution for the PyramidAT.

5.2. Pyramid attacks

In Figure 3, we include 4 additional visualizations of pyramid attacks against the PyramidAT models. Note that in the finest level, more structure is visible.

5.3. Attention

We include the average attentions of baseline, PixelAT, and PyramidAT on the following datasets: ImageNet (Figure 4), ImageNet-A (Figure 5), ImageNet-ReaL (Figure 6), ImageNet-Rendition (Figure 7), ObjectNet (Figure 8), and Stylized ImageNet (Figure 9). The trend, as stated in the main paper, (PixelAT tightly focusing on the center and PyramidAT taking a more global perspective) stays consistent across the various evaluation datasets.

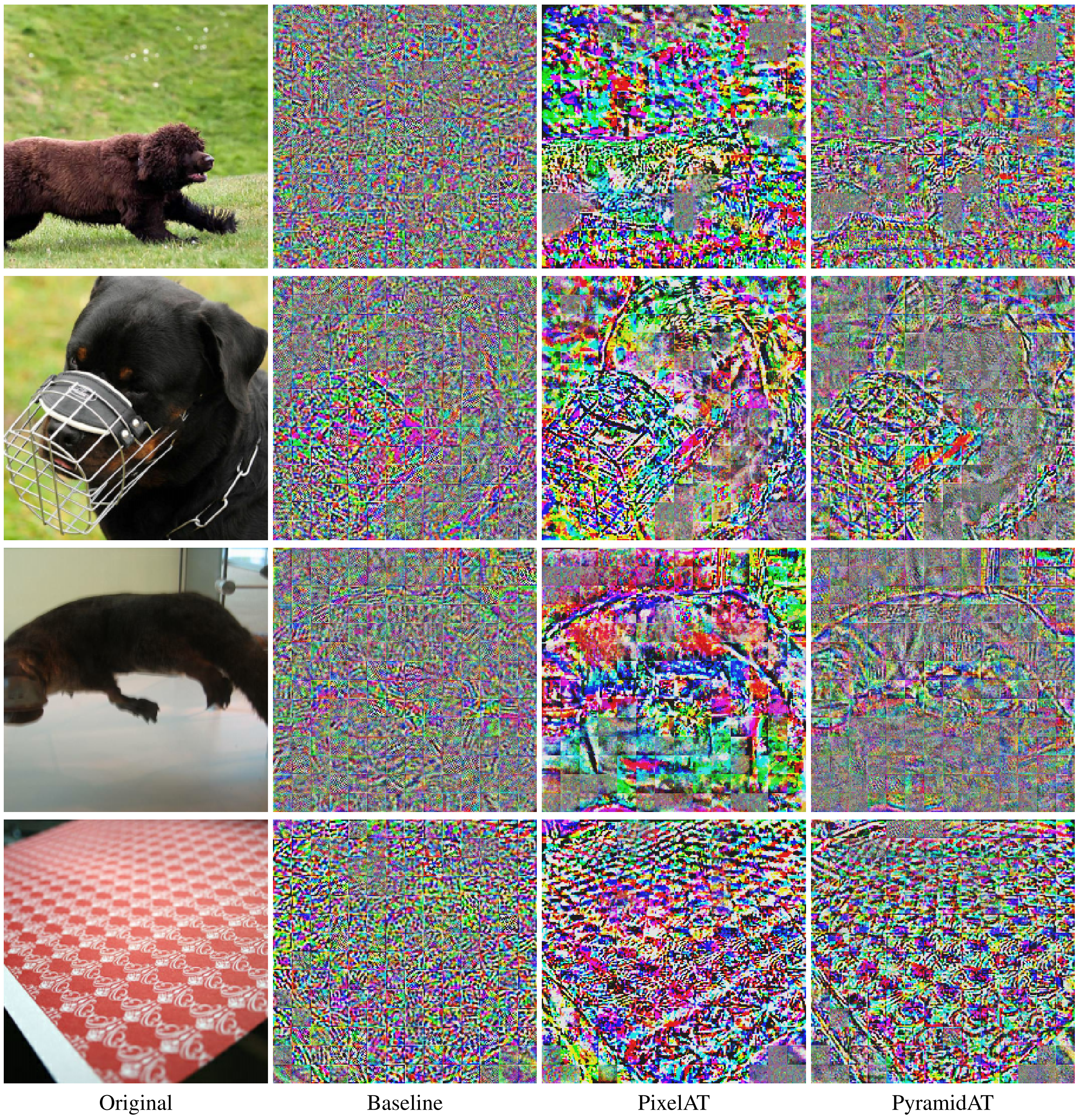

Figure 2. Visualizations of pixel attacks on different pre-trainings: baseline, PixelAT, and PyramidAT. Note that PixelAT models should have better defense against pixel attacks than PyramidAT models since the attack is in-distribution to the train data.

We also include 32 examples of the attention for individual images sampled from the following datasets: ImageNet (Figure 10), ImageNet-A (Figure 11), ImageNet-ReaL (Figure 12), ImageNet-Rendition (Figure 13), ObjectNet (Figure 14), and StylizedImageNet (Figure 15). The trend, as stated in the main paper, remains consistent through most of the examples. Baseline tends to be random and highlight both the object and background (particularly corners); PixelAT tries to aggressively crop to the object in the image, often cutting off parts of the object; and PyramidAT crops more closely than baseline but less aggressively than PixelAT. PyramidAT tends to take a more global perspective on the image and attends to both the object but also potentially relevant pieces of the background.

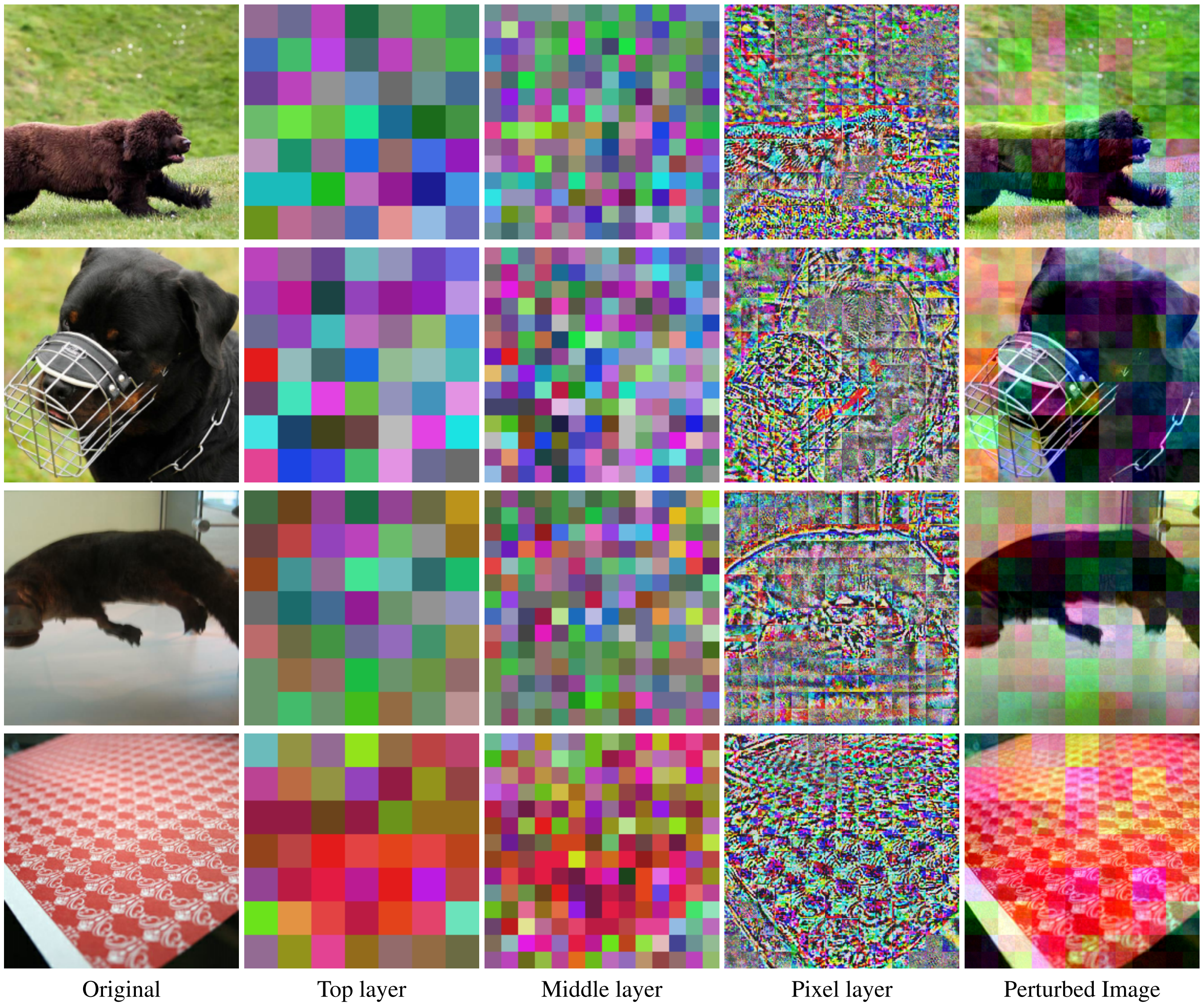

Figure 3. Visualizations of pyramid attacks on pyramid-trained model.

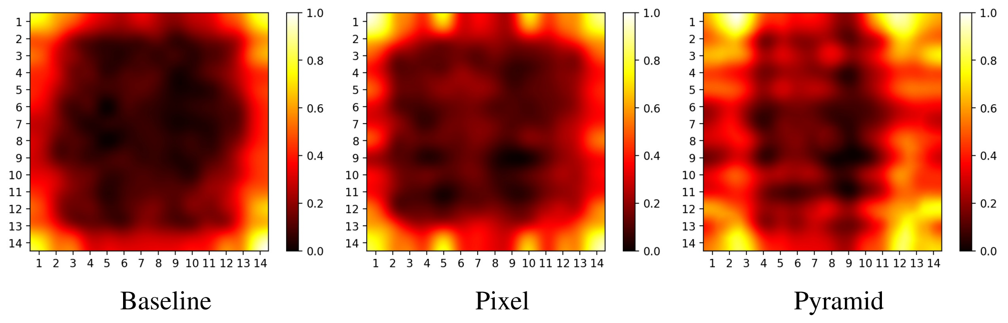

Figure 4. Visualizations of the average attention for different pre-trainings. Examples on dataset ImageNet.

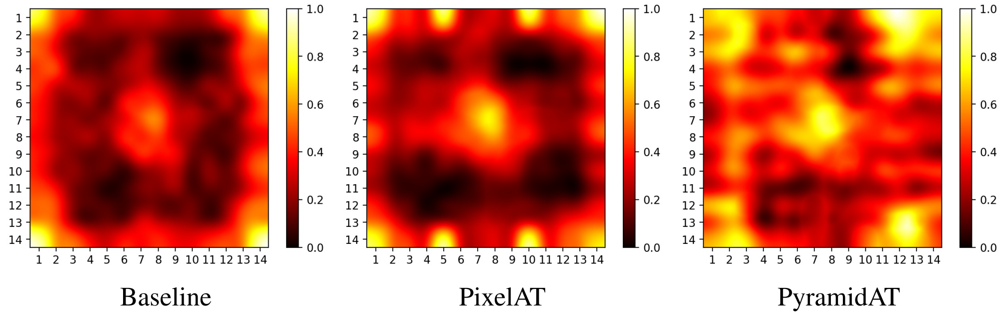
Figure 5. Visualizations of the average attention for different pre-trainings. Examples on dataset ImageNet-A.

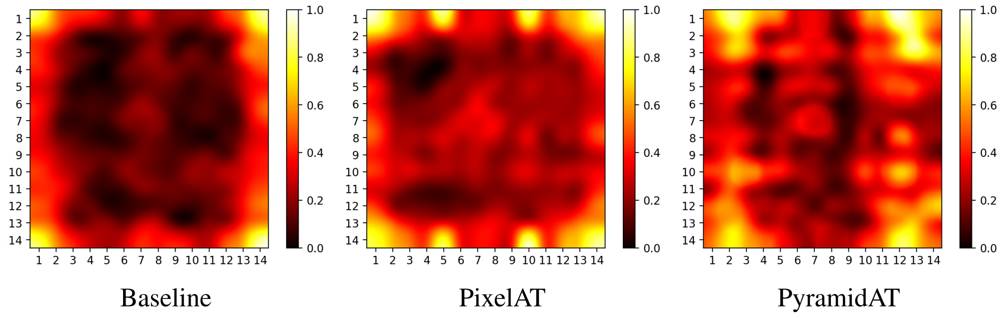
Figure 6. Visualizations of the average attention for different pre-trainings. Examples on dataset ImageNet2012-ReaL.

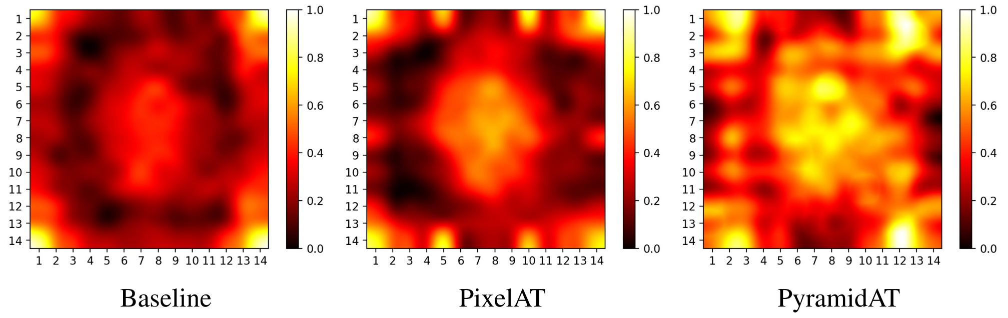
Figure 7. Visualizations of the average attention for different pre-trainings. Examples on dataset ImageNet-Rendition.

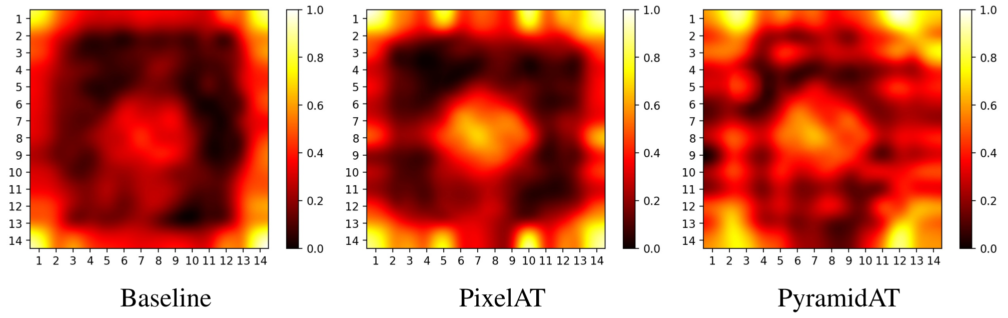
Figure 8. Visualizations of the average attention for different pre-trainings. Examples on dataset ObjectNet.

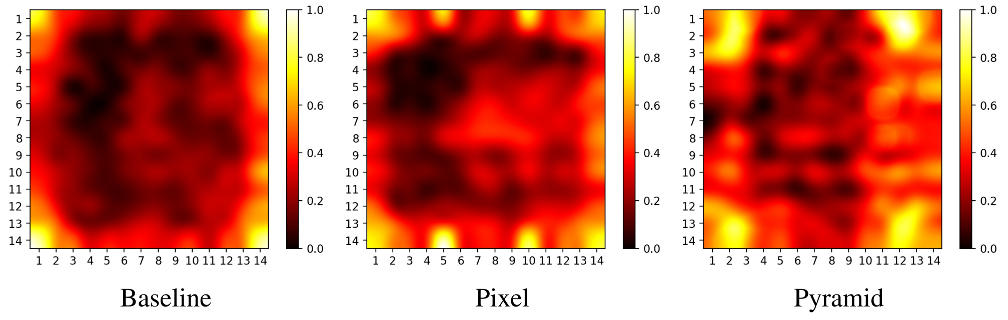

Figure 9. Visualizations of the average attention for different pre-trainings. Examples on dataset StylizedImageNet.

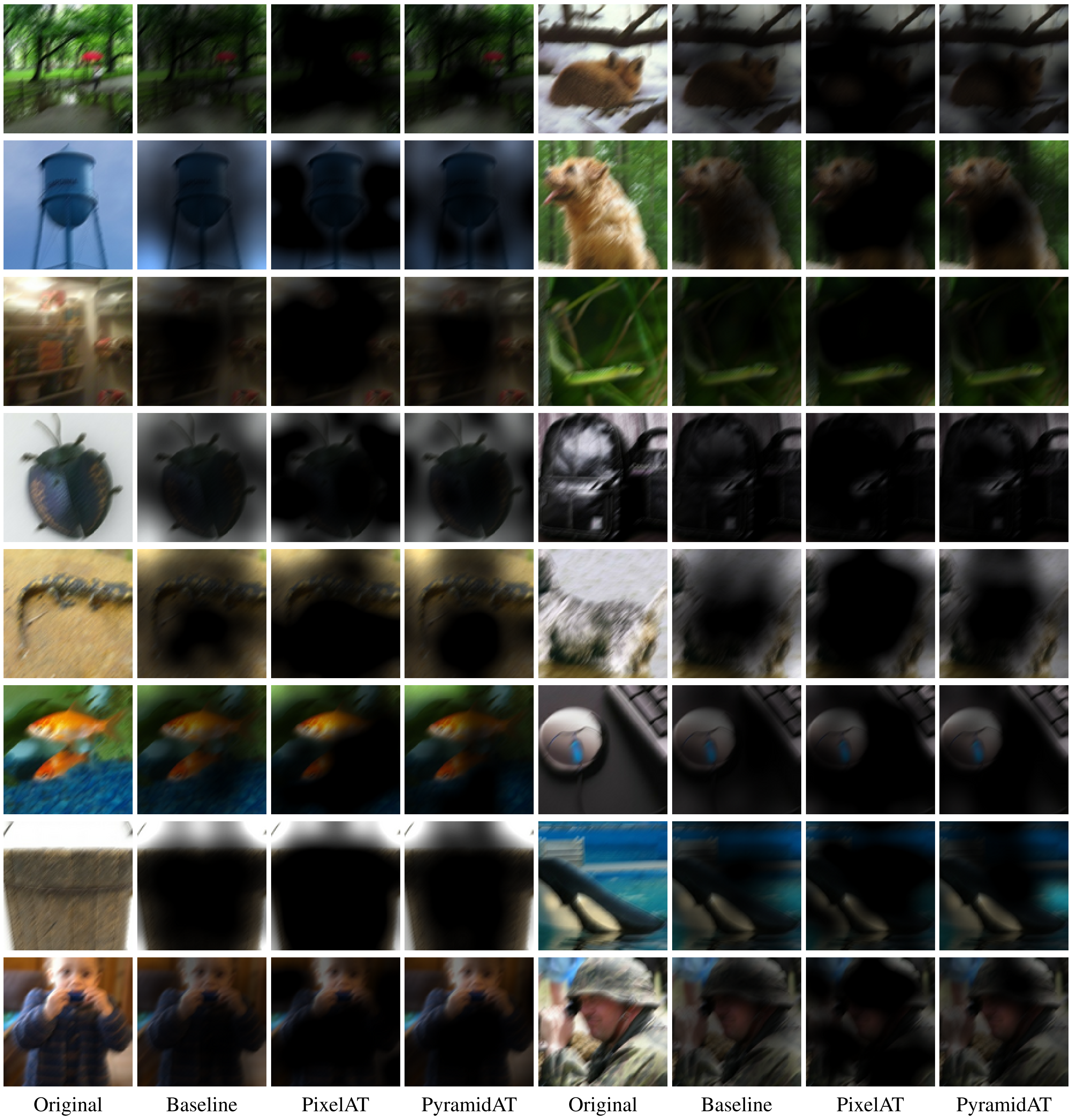

Figure 10. Visualizations of the attention for different pre-trainings. Examples on dataset ImageNet.

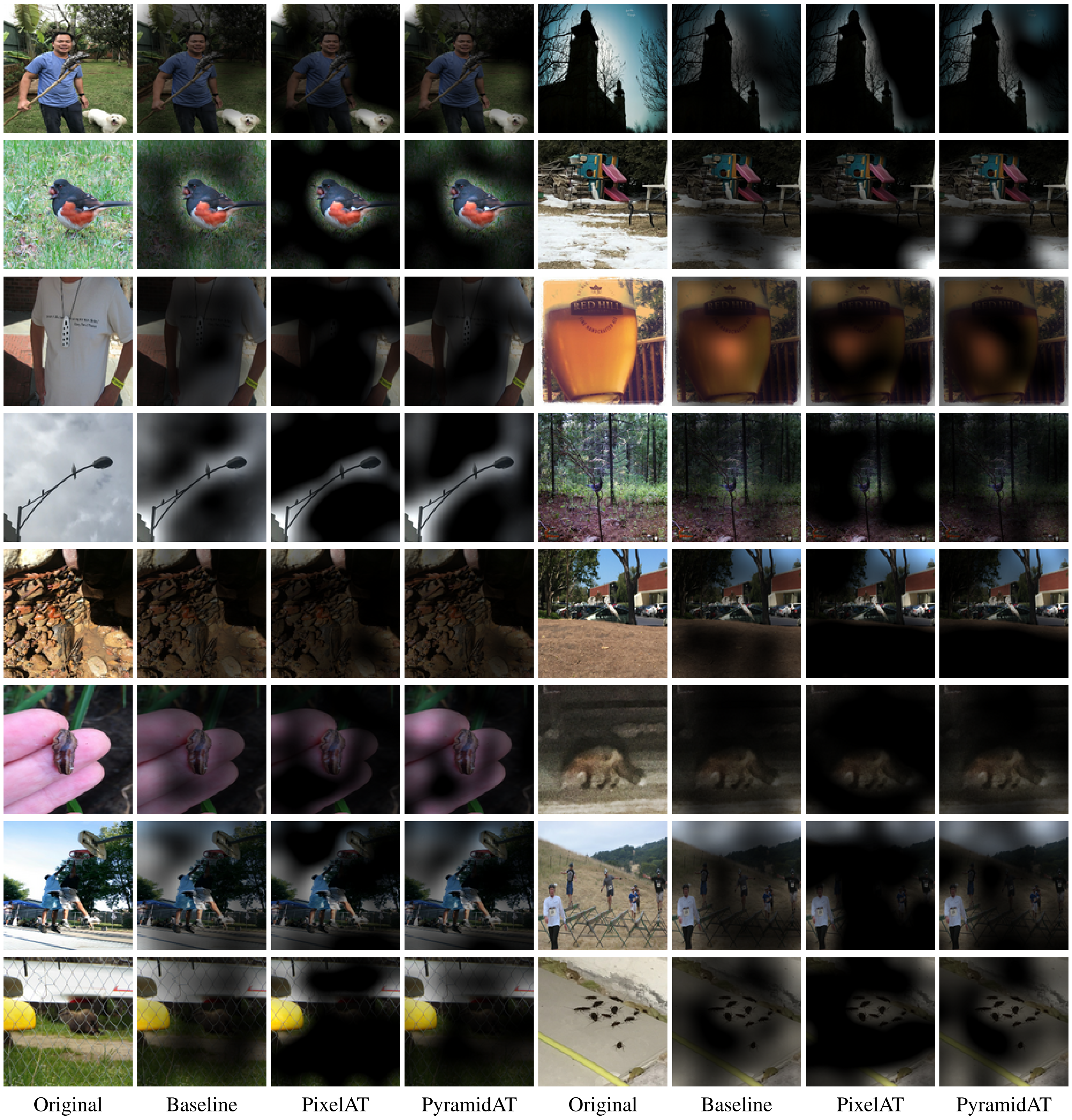

Figure 11. Visualizations of the attention for different pre-trainings. Examples on dataset ImageNet-A.

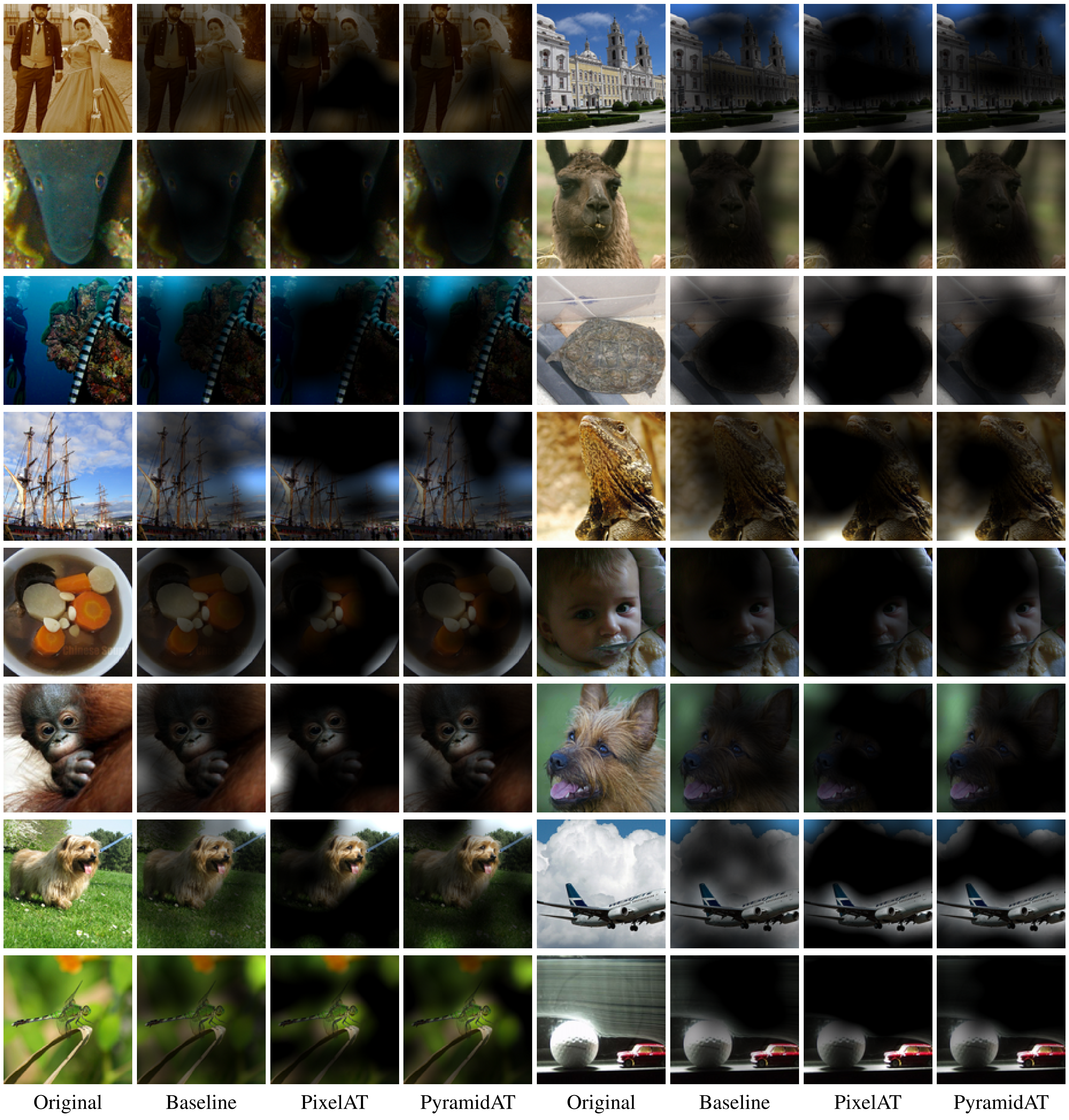

Figure 12. Visualizations of the attention for different pre-trainings. Examples on dataset ImageNet2012-ReaL.

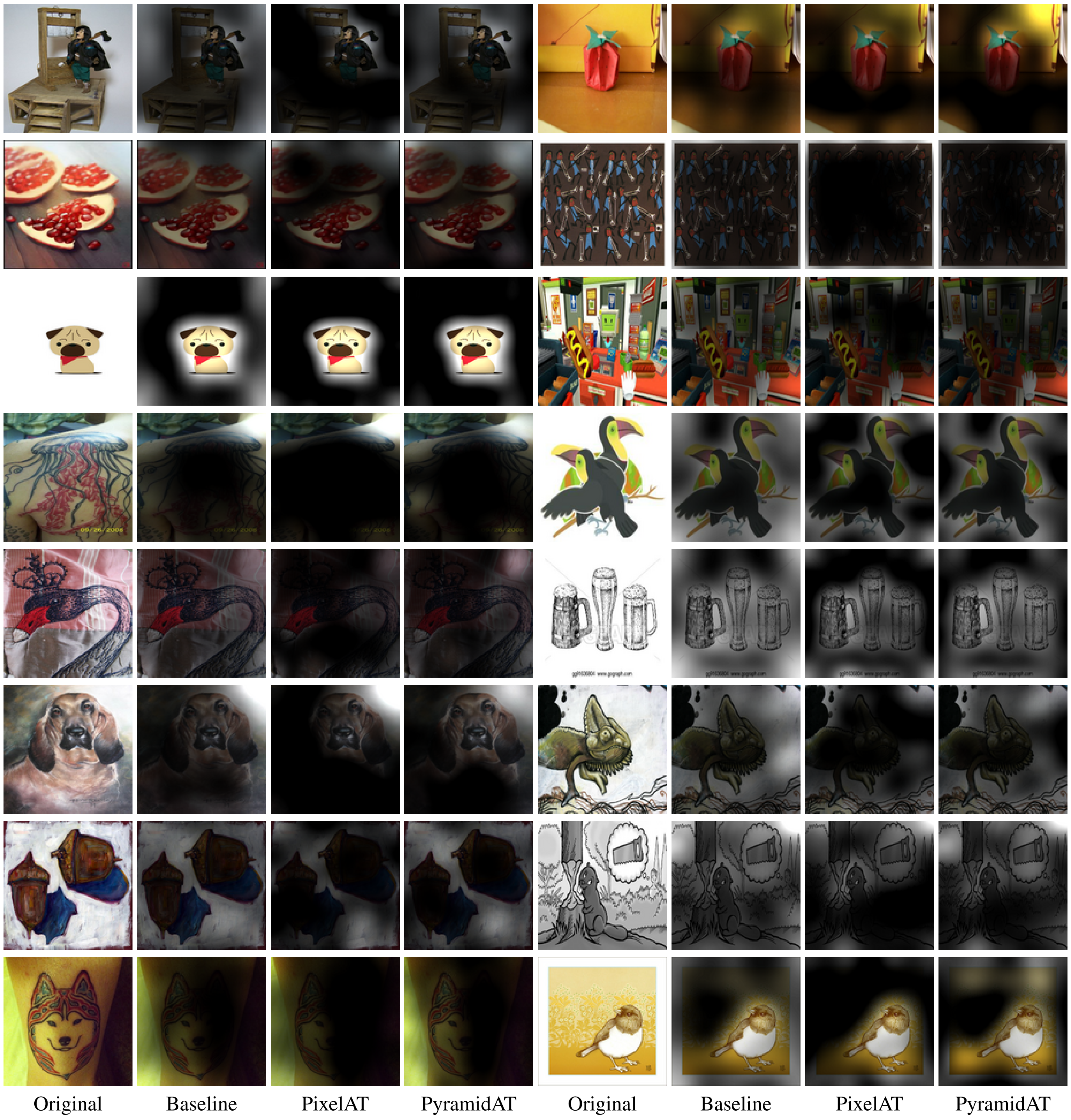

Figure 13. Visualizations of the attention for different pre-trainings. Examples on dataset ImageNet-Rendition.

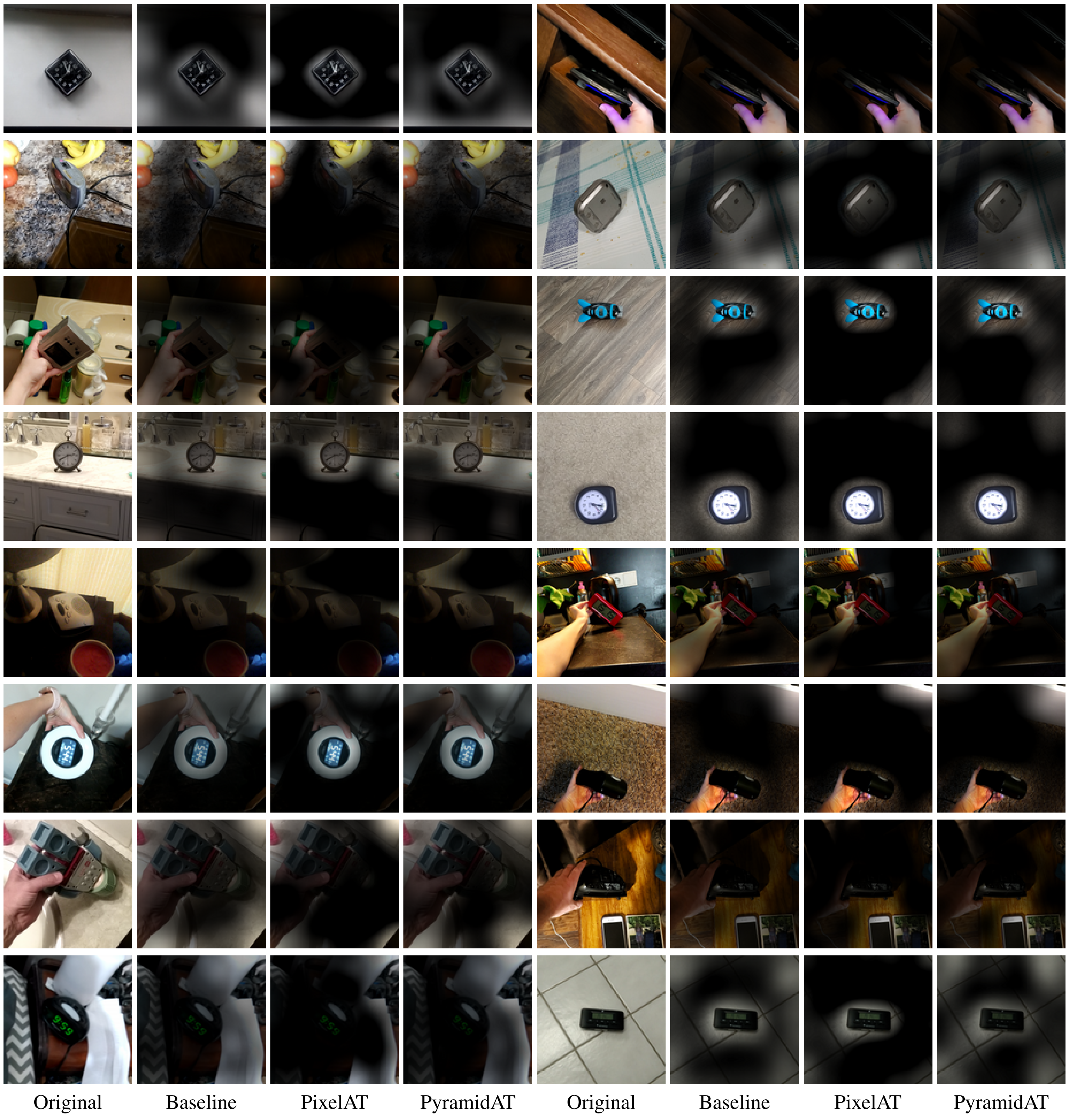

Figure 14. Visualizations of the attention for different pre-trainings. Examples on dataset ObjectNet.

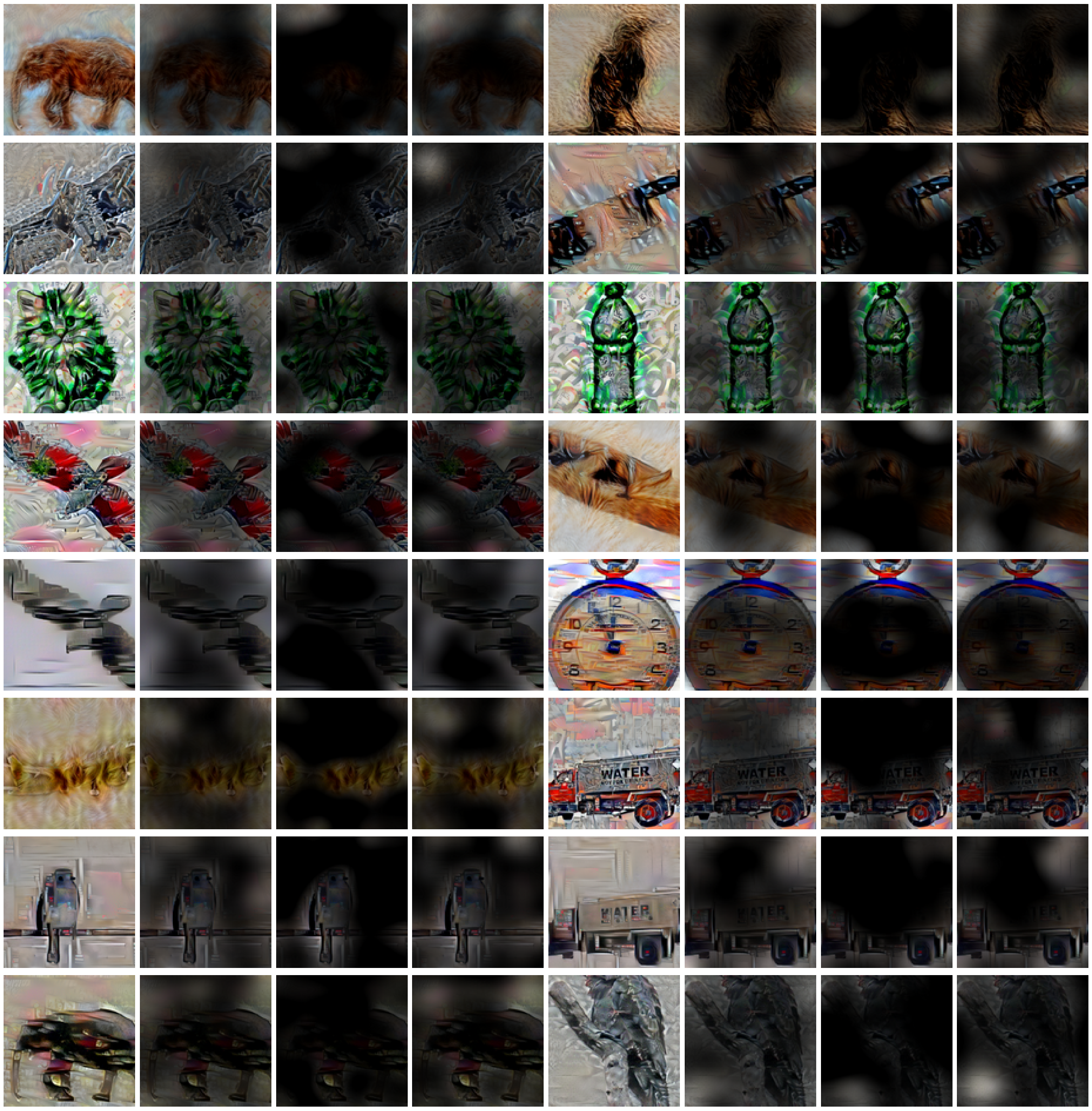

Figure 15. Visualizations of the attention for different pre-trainings. Examples on dataset StylizedImageNet.

6. Optimizers

We observe different behavior from adversarial training depending on the optimizer used in generating the attacks; note, discussion of optimizers was omitted from the main paper due to concerns regarding space and complexity. Throughout the main paper, we use SGD, the standard optimizer in the adversarial attack and training community. However after testing multiple optimizers (Adam [26], AdaBelief [66]), we observe significantly different behavior from AdaBelief. Specifically, as shown in Table 23, AdaBelief provides a significant improvement to PixelAT (0.71 to ImageNet, 1.72 in ImageNet-R) and a marginal improvement to PyramidAT (0.08 to ImageNet, 0.98 in ImageNet-R).

Method	ImageNet	Real	Out of Distribution Robustness Test						
			A	C↓	ObjectNet	V2	Rendition	Sketch	Stylized
PixelAT SGD	80.42	85.78	19.15	47.68	30.11	68.78	45.39	34.40	18.28
PixelAT AdaBelief	**81.13**	**86.40**	**21.45**	**45.25**	**32.03**	**69.97**	**47.11**	**36.18**	**20.55**
PyramidAT SGD	81.71	**86.82**	22.99	44.99	**32.92**	70.82	47.66	36.77	19.14
PyramidAT AdaBelief	**81.79**	86.79	**23.24**	**44.79**	32.81	**70.87**	**48.64**	**38.38**	**20.78**

Table 23. SGD vs AdaBelief

As shown in Figure 16, we also observe significant visual difference in the pixel attacks on the pixel-trained model with AdaBelief.

Shown in Figure 17, this visual difference is more apparent when looking at pixel attacks using AdaBelief on these four different pre-trainings. In the pixel attacks using AdaBelief on AdaBelief pixel-trained model, contours and edges are clearly visible and the edits to the texture are smoother and more consistent. Even beyond classification, this may provide a way to do semi-supervised segmentation (with only the class label). Currently, AdaBelief does not provide such visible changes or improvements to pyramid. We leave this adaptation to future work.

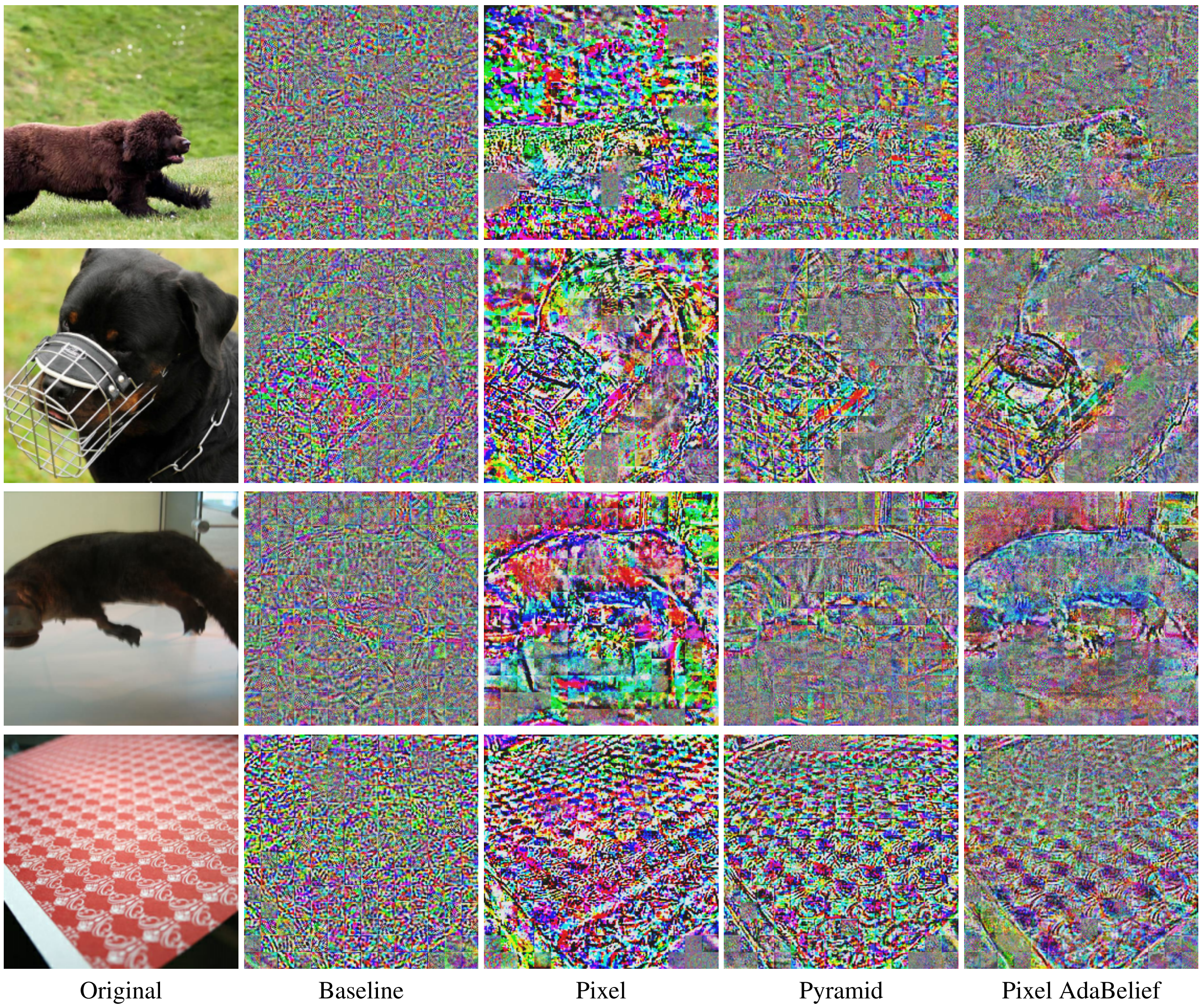

Figure 16. Visualizations of pixel attacks using SGD on different pre-trainings: baseline, pixel, pyramid, and pixel Adabelief.

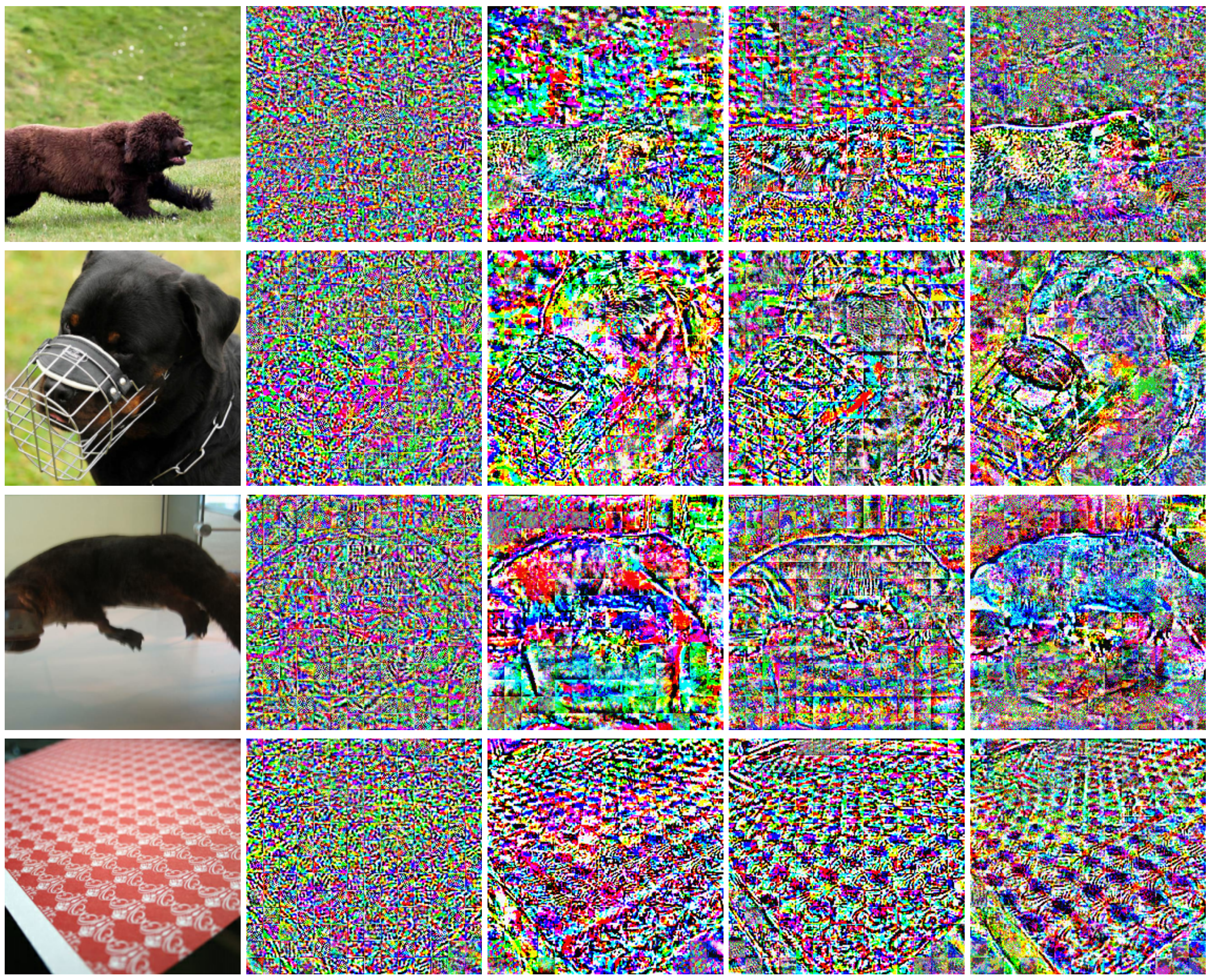

Figure 17. Visualizations of pixel attacks using AdaBelief on different pre-trainings: baseline, PixelAT SGD, PyramidAT SGD, and PixelAT Adabelief.

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, pages 274–283. PMLR, 2018.

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019.

[4] Lucas Beyer, Olivier J. Henaff, Alexander Kolesnikov, Xiaohua Zhai, and Aaron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2002.05709*, 2020.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (SP)*, pages 39–57. IEEE, 2017.

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[8] Yong Cheng, Lu Jiang, and Wolfgang Macherey. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*, 2019.

[9] Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. Advaug: Robust adversarial augmentation for neural machine translation. *arXiv preprint arXiv:2006.11834*, 2020.

[10] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[11] Mostafa Dehghani, Alexey Gritsenko, Anurag Arnab, Matthias Minderer, and Yi Tay. Scenic: A JAX library for computer vision research and beyond. *arXiv preprint arXiv:2110.11403*, 2021.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 19

[13] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[15] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning (ICML)*, pages 2286–2296. PMLR, 2021.

[16] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019. 19

[17] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 12

[19] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 19

[20] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019. 19

[21] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[22] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 19

[23] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 15

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[25] Insoo Kim, Seungju Han, Ji-won Baek, Seong-Jin Park, Jae-Joon Han, and Jinwoo Shin. Quality-agnostic image recognition via invertible decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12257–12266, June 2021.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015. 32

[27] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[28] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[29] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[30] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[31] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018.

[32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[33] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin Dogus Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *ArXiv*, abs/1906.02611, 2019.

[34] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

[35] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018.

[36] Kaleel Mahmood, Rigel Mahmood, and Marten van Dijk. On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7838–7847, October 2021.

[37] Chengzhi Mao, Lu Jiang, Mostafa Dehghani, Carl Vondrick, Rahul Sukthankar, and Irfan Essa. Discrete representations strengthen vision transformer robustness. In *International Conference on Learning Representations (ICLR)*, 2022.

[38] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer. *CoRR*, abs/2105.07926, 2021.

[39] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[40] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9078–9086, 2019.

[41] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[42] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[43] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[44] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[45] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. *CoRR*, abs/2002.10716, 2020.

[46] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, October 2021.

[47] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, pages 5389–5400. PMLR, 2019.

[48] 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.

[49] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of visual transformers. *CoRR*, abs/2103.15670, 2021.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 15

[51] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *CoRR*, abs/2106.10270, 2021. 15

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[53] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, 2019.

[54] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. 15

[55] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357. PMLR, 2021.

[56] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy, 2019.

[57] Abraham Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945.

[58] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10506–10518, 2019. 19

[59] Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar, and Zhangyang Wang. Augmax: Adversarial composition of random augmentations for robust training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[60] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[61] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 819–828, 2020. 13, 19

[62] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[63] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6023–6032, 2019.

[64] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[65] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 09–15 Jun 2019.

[66] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 18795–18806, 2020. 32