Appendix to Increasing the robustness of DNNs against image corruptions by playing the Game of Noise

A Architectures of the noise generators

The architectures of the noise generators are displayed in Tables 1 and 2. The number of color channels is indicated by C. The noise generator displayed in Table 1 only uses kernels with a size of 1 and thus produces spatially uncorrelated noise. With the stride being 1 and no padding, the spatial dimensions are preserved in each layer. The noise generator displayed in Table 2 has one layer with 3x3 convolutions and thus produces noise samples with a correlation length of 3x3 pixels.

Layer	Shape
Conv + ReLU	$20 \times 1 \times 1$
Conv + ReLU	$20 \times 1 \times 1$
Conv + ReLU	$20 \times 1 \times 1$
Conv	C $\times 1 \times 1$

Table 1: Architecture of the noise generator producing uncorrelated noise.

Layer	Shape
Conv + ReLU	$20 \times 1 \times 1$
Conv + ReLU	$20 \times 3 \times 3$
Conv + ReLU	$20 \times 1 \times 1$
Conv	C $\times 1 \times 1$

Table 2: Architecture of the noise generator producing locally correlated noise.

B Implementation details and hyper-parameters

We use PyTorch [7] for all of our experiments.

Preprocessing MNIST images are preprocessed such that their pixel values lie in the range $[0, 1]$. Preprocessing for ImageNet is performed in the standard way for PyTorch ImageNet models from the model zoo by subtracting the mean $[0.485, 0.456, 0.406]$ and dividing by the standard deviation $[0.229, 0.224, 0.225]$. We add Gaussian, adversarial and Speckle noise before the preprocessing step, so the noisy images are first clipped to the range $[0, 1]$ of the raw images and then preprocessed before being fed into the model.

ImageNet experiments For all ImageNet experiments, we used a pretrained ResNet50 architecture from `https://pytorch.org/docs/stable/torchvision/models.html`. We fine-tuned the model with SGD-M using an initial learning rate of 0.001, which corresponds to the last learning rate of the PyTorch model training, and a momentum of 0.9. After convergence, we decayed the learning rate once by a factor of 10 and continued the training. Decaying the learning rate was highly beneficial for the model performance. We tried decaying the learning rate a second time, but this did not bring any benefits in any of our experiments. For GNT, we also tried training from scratch, i.e. starting with a large learning rate of 0.1 and random weights, and trained for 120 epochs, but we got worse results compared to merely fine-tuning the model provided by torchvision. We used a batch size of 70 for all our experiments. We have also tried to use the batch sizes 50 and 100, but did not observe any difference.

Gaussian noise We trained the models until convergence. The total number of training epochs varied between 30 and 90 epochs.

Speckle noise We used the Speckle noise implementation from `https://github.com/hendrycks/robustness/blob/master/ImageNet-C/create_c/make_imagenet_c.py`, line 270. The model trained with Speckle noise converged faster than with Gaussian data augmentation and therefore, we only trained the model for 10 epochs.

Adversarial Noise Training The adversarial noise generator was trained with the Adam optimizer with a learning rate of 0.0001. We have replaced the noise generator every 0.33 epochs. For ANT^{1x1}, we set the ϵ-sphere to control the size of the perturbation to 135.0 which on average corresponds to the ℓ_2-size of a perturbation caused by additive Gaussian noise sampled from $\mathcal{N}(0, 0.5^2 \cdot \mathbb{1})$. We have trained the classifier until convergence for 80 epochs. For ANT^{3x3}, we set the ϵ-sphere to 70.0 and trained the classifier for 80 epochs. We decreased the ϵ-sphere for ANT^{3x3} to counteract giving the noise generator more degrees of freedom to fool the classifier to maintain a similar training losses and accuracies for ANT^{1x1} and ANT^{3x3}.

MNIST experiments For the MNIST experiments, we used the same model architecture as [6] for our ANT^{1x1} and GNT. For ANT^{1x1}, our learning rate for the generator was between 10^{-4} and 10^{-5}, and equal to 10^{-3} for the classifier. We used a batch size of 300. As an optimizer, we used SGD-M with a momentum of 0.9 for the classifier and Adam [5] for the generator. The splitting of batches in clean, noisy and history was equivalent to the ImageNet experiments. The optimal ϵ hyper-parameter was determined with a line search similar to the optimal σ of the Gaussian noise; we found $\epsilon = 10$ to be optimal. The parameters for the Gaussian noise experiments were equivalent. Both models were trained until convergence (around 500-600 epochs). GNT and ANT^{1x1} were performed on a pretrained network.

C Detailed results on the evaluation of corruption robustness due to regular adversarial training

We find that standard adversarial training against minimal adversarial perturbations in general does not increase robustness against common corruptions. While some early results on CIFAR-10 by [1] and Tiny ImageNet-C by [3] suggest that standard adversarial training might increase robustness to common corruptions, we here observe the opposite: Adversarially trained models have lower robustness against common corruptions. An adversarially trained ResNet152 with an additional denoising layer[1] from [12] has lower accuracy across almost all corruptions except Snow and Pixelations. On some corruptions, the accuracy of the adversarially trained model decreases drastically, e.g. from 49.1% to 4.6% on Fog or 42.8% to 9.3% on Contrast. Similarly, the adversarially trained ResNet50[2] from [Shafahi et al., 2019] shows a substantial decrease in performance on common corruptions compared with a vanilla trained model.

An evaluation of a robustified version of AlexNet[2] [10] that was trained with the Universal Adversarial Training scheme on ImageNet-C shows that achieving robustness against universal adversarial perturbations does not noticeably increase robustness towards common corruptions (22.2%) compared with a vanilla trained model (21.1%).

[1] Model weights from `https://github.com/facebookresearch/ImageNet-Adversarial-Training`
[2] Model weights were kindly provided by the authors.

		Noise (Compressed)			Blur (Compressed)			
Model	All	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom
Vanilla RN50	39.2	29.3	27.0	23.8	38.7	26.8	38.7	36.2
AT [9]	29.1	20.5	19.1	12.4	21.4	30.8	30.4	31.4
Vanilla RN152	45.0	35.7	34.3	29.6	45.1	32.8	48.4	40.5
AT [12]	35.0	35.2	34.4	24.8	22.1	31.7	30.9	32.0
Vanilla AlexNet	21.1	11.4	10.6	7.7	18.0	17.4	21.4	20.2
UAT [10]	22.2	20.1	19.1	16.2	13.1	21.6	19.7	19.2

	Weather (Compressed)				Digital (Compressed)			
Model	Snow	Frost	Fog	Brightness	Contrast	Elastic	Pixelate	JPEG
Vanilla RN50	32.5	38.1	45.8	68.0	39.1	45.2	44.8	53.4
AT [9]	24.4	25.6	5.8	51.1	7.8	45.4	53.4	56.3
Vanilla RN152	38.7	43.9	49.1	71.2	42.8	51.1	50.5	60.5
AT [12]	42.0	40.4	4.6	58.8	9.3	47.2	54.1	58.0
Vanilla AlexNet	13.3	17.3	18.1	43.5	14.7	35.4	28.2	39.4
UAT [10]	13.8	18.3	4.3	36.5	4.8	36.8	42.3	47.1

Table 3: Average Top-1 accuracy over 5 severities of common corruptions on ImageNet-C in percent. A high accuracy on a certain corruption type indicates high robustness of a classifier on this corruption type, so higher accuracy is better. Adversarial training (AT) decreases the accuracy on common corruptions, especially on the corruptions Fog and Contrast. Universal Adversarial Training (UAT) slightly increases the overall performance.

D Detailed ImageNet-C results

We show detailed results on individual corruptions in Table 4 in accuracy and in Table 5 in mCE for differently trained models. In Fig. 1, we show the degradation of accuracy for different severity levels. To avoid clutter, we only show results for a vanilla trained model, for the previous state of the art SIN+IN [2], for several Gaussian trained models and for the overall best model ANT^{3x3}+SIN.

The Corruption Error [3] is defined as

$$\mathrm{CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f \right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}} \right), \tag{1}$$

where $E_{s,c}^f$ is the Top-1 error of a classifier f for a corruption c with severity s. The mean Corruption error (mCE) is taken by averaging over all corruptions.

		Noise			Blur				Weather				Digital			
model	mean	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Vanilla RN50	39	29	27	24	39	27	39	36	33	38	46	68	39	45	45	53
Shift Inv	42	36	34	30	40	29	38	39	33	40	48	68	42	45	49	57
Patch GN	44	45	43	42	38	26	39	38	30	39	54	67	39	52	47	56
SIN+IN	45	41	40	37	43	32	45	36	41	42	47	67	43	50	56	58
AugMix	48	41	41	38	**48**	35	**54**	**49**	40	44	47	69	**51**	52	57	60
Speckle	46	55	58	49	43	32	40	36	34	41	46	68	41	47	49	58
GNT$_{\mathrm{mult}}$	49	**67**	65	**64**	43	33	41	37	34	42	45	68	41	48	50	60
GNT$\sigma_{0.5}$	49	58	59	57	47	38	43	42	35	44	44	68	39	50	55	**62**
ANT^{1x1}	51	65	**66**	**64**	47	37	43	40	36	46	44	**70**	43	49	55	**62**
ANT^{1x1}+SIN	52	64	65	63	46	38	46	39	42	47	49	69	47	50	57	60
ANT^{1x1} w/o EP	49	59	59	57	46	37	43	40	34	43	43	68	39	49	55	61
ANT^{3x3}	50	65	64	**64**	44	36	42	38	39	46	44	69	41	49	55	61
ANT^{3x3}+SIN	**53**	62	61	60	41	**39**	46	37	**48**	**52**	**55**	68	49	**53**	**59**	59

Table 4: Average Top-1 accuracy over 5 severities of common corruptions on ImageNet-C in percent obtained by different models; higher is better.

E MNIST-C results

Similar to the ImageNet-C experiments, we are interested how vanilla, adversarially and noise trained models perform on MNIST-C.

The adversarially robust MNIST model by [11] was trained with a robust loss function and is among the state of the art in certified adversarial robustness. The other baseline models were trained with Adversarial Training in ℓ_2 (DDN) by [8] and ℓ_∞ (PGD) by [6]. Our GNT and ANT^{1x1} trained versions are trained as described in the main paper and Appendix B.2. The results are shown in Table 6. Similar to ImageNet-C, the models trained with GNT and ANT^{1x1} are significantly better than our vanilla trained baseline. Also, regular adversarial training has severe drops and does not lead to significant robustness improvements.

As for ImageNet and GNT, we have treated σ as a hyper-parameter. The accuracy on MNIST-C for different values of σ is displayed in Fig. 2 and has a maximum around $\sigma = 0.5$ like for ImageNet.

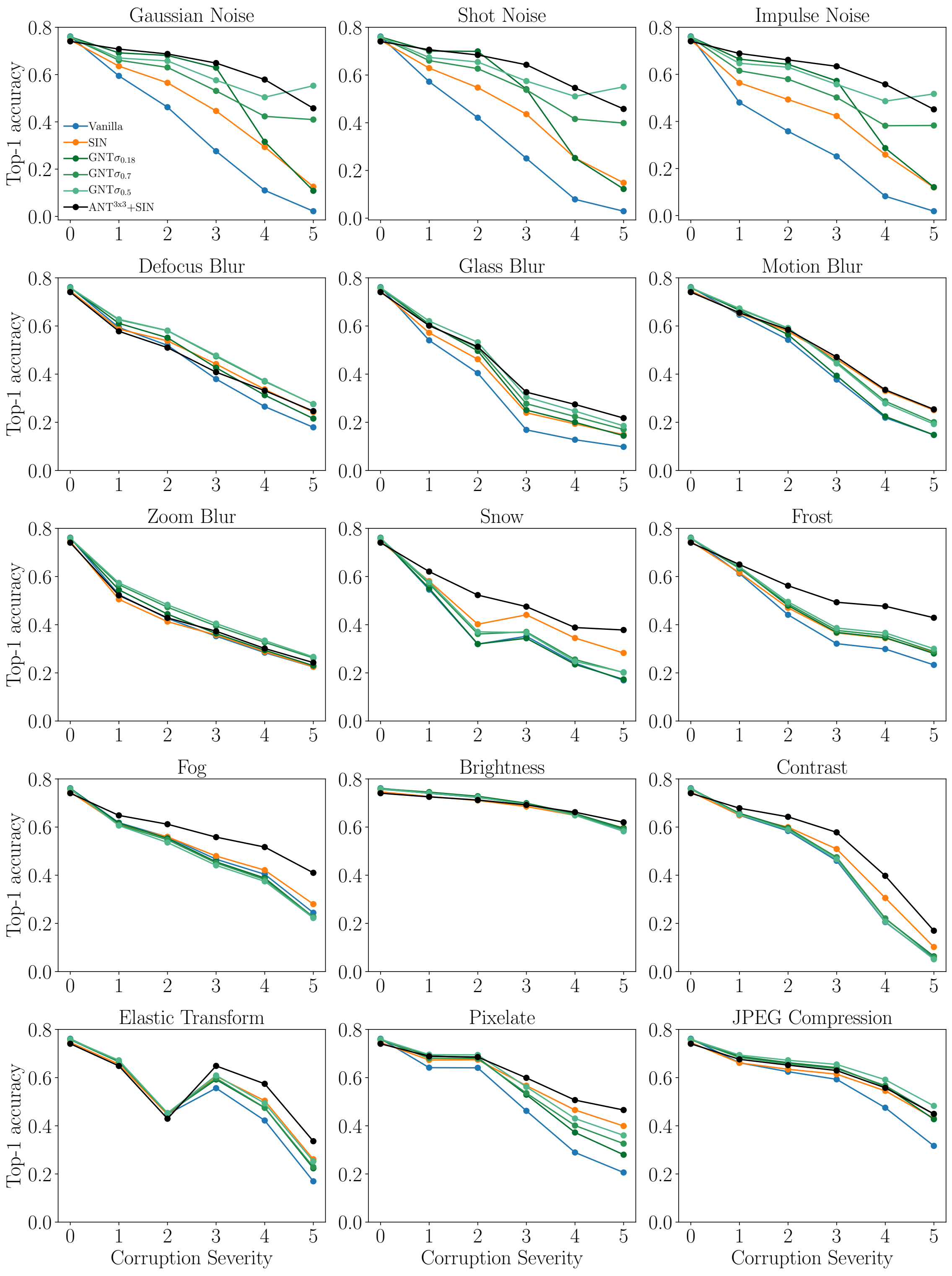

Fig. 1: Top-1 accuracy for each corruption type and severity on ImageNet-C.

		Noise			Blur				Weather				Digital			
model	mCE	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Vanilla	77	80	82	83	75	89	78	80	78	75	66	57	71	85	77	77
SIN	69	66	67	68	70	82	69	80	68	71	65	58	66	78	62	70
Patch GN	71	62	63	62	75	90	78	78	81	74	57	59	71	74	74	72
Shift Inv.	73	73	74	76	74	86	78	77	77	72	63	56	68	86	71	71
AugMix	65	67	66	68	**64**	79	**59**	**64**	69	68	65	54	**57**	74	60	65
Speckle	68	51	47	55	70	83	77	80	76	71	66	57	70	82	71	69
GNT$_{\text{mult}}$	65	**37**	39	**39**	69	81	76	79	76	70	67	56	69	81	69	66
GNT$\sigma_{0.5}$	64	46	46	47	65	75	72	74	75	68	69	57	71	78	63	63
ANT$^{1\text{x}1}$	62	39	**38**	**39**	65	77	72	75	74	66	68	**53**	67	78	62	**62**
ANT$^{1\text{x}1}$+SIN	**61**	40	39	40	65	76	69	76	67	64	62	55	63	77	59	66
ANT$^{1\text{x}1}$ w/o EP	65	46	46	47	66	76	73	75	76	69	70	57	72	79	63	64
ANT$^{3\text{x}3}$	63	39	40	**39**	68	78	73	77	71	66	68	55	69	79	63	64
ANT$^{3\text{x}3}$+SIN	**61**	43	44	43	71	**74**	69	79	**60**	**58**	**55**	56	59	**73**	**57**	67

Table 5: Average mean Corruption Error (mCE) obtained by different models on common corruptions from ImageNet-C; lower is better.

model	clean acc	mean	Shot	Impulse	Glass Blur	Motion Blur	Shear	Scale	Rotate	Brightness	Translate	Stripe	Fog	Splatter	Dotted Line	Zig Zag	Canny Edges
Vanilla	99.1	86.9	98	96	96	94	98	95	92	88	57	88	50	97	96	86	72
[6]	98.5	75.6	98	55	94	94	97	88	92	27	53	40	63	96	78	74	84
Vanilla	98.8	74.3	98	91	96	88	95	80	89	34	45	41	23	96	96	80	63
[11]	98.2	68.6	97	65	93	93	94	87	89	11	40	20	25	96	89	61	68
Vanilla	99.5	89.8	98	96	95	97	98	96	94	95	61	89	79	98	98	90	63
DDN Tr [8]	99.0	87.0	99	97	96	94	98	91	93	72	55	92	64	99	98	91	66
Vanilla	99.1	86.9	98	96	96	94	98	95	92	88	57	88	50	97	96	86	72
GNT$\sigma_{0.5}$	99.3	92.4	99	99	98	97	98	95	93	98	56	91	91	99	99	96	78
ANT$^{1\text{x}1}$	99.4	92.4	99	99	98	97	98	95	93	98	55	89	91	99	99	96	80

Table 6: Accuracy in percent for the MNIST-C dataset for adversarially robust ([11], [6], DDN [8]) and our noise trained models (GNT and ANT$^{1\text{x}1}$). Vanilla always denotes the same network architecture as its adversarially or noise trained counterpart but with standard training. Note that we used the same network architecture as [6].

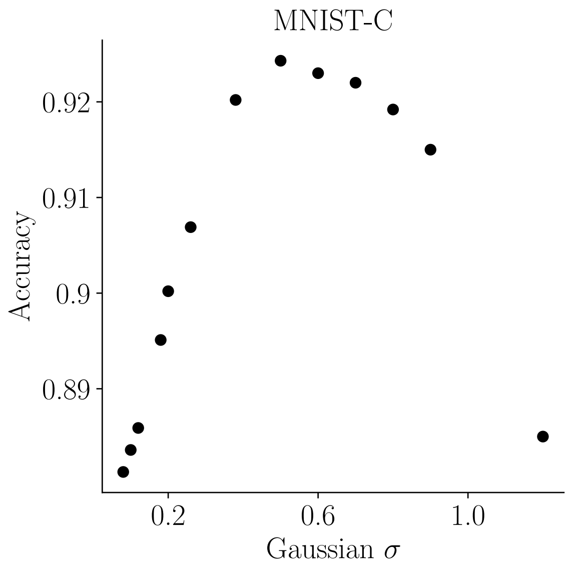

Fig. 2: Average accuracy on MNIST-C over all severties and corruptions for different values of sigma σ of the Gaussian noise training (GNT) during training. Each point corresponds to one converged training.

F Evaluation of adversarial robustness of models trained via GNT and ANT^{1x1}

ImageNet To evaluate adversarial robustness on ImageNet, we used PGD [6] and DDN [8]. For the ℓ_∞ PGD attack, we allowed for 200 iterations with a step size of 0.0001 and a maximum sphere size of 0.001. For the DDN ℓ_2 attack, we also allowed for 200 iterations, set the sphere adjustment parameter γ to 0.02 and the maximum epsilon to 0.125. We note that for both attacks increasing the number of iterations from 100 to 200 did not make a significant difference in robustness of our tested models. The results on adversarial robustness on ImageNet can be found in the main paper in Table 4.

MNIST To evaluate adversarial robustness on MNIST, we also used PGD [6] and DDN [8]. For the ℓ_∞ PGD attack, we allowed for 100 iterations with a step size of 0.01 and a maximum sphere size of 0.1. For the DDN ℓ_2 attack, we also allowed for 100 iterations, set the sphere adjustment parameter γ to 0.05 and the maximum epsilon to 1.5. All models have the same architecture as [6]. The results on adversarial robustness on MNIST can be found in Table 7.

model	clean acc. [%]	ℓ_2 acc. [%]	ℓ_∞ acc. [%]
Vanilla	99.1	73.2	55.8
GNT$\sigma_{0.5}$	99.3	89.2	73.6
ANT^{1x1}	99.4	90.4	76.3

Table 7: Adversarial robustness on MNIST on ℓ_2 ($\epsilon = 1.5$) and ℓ_∞ ($\epsilon = 0.1$) compared to a Vanilla CNN.

G Example images for additive Gaussian noise

Example images with additive Gaussian noise of varying standard deviation σ are displayed in Fig. 3. The considered σ-levels correspond to those studied in section 4.2. in the main paper.

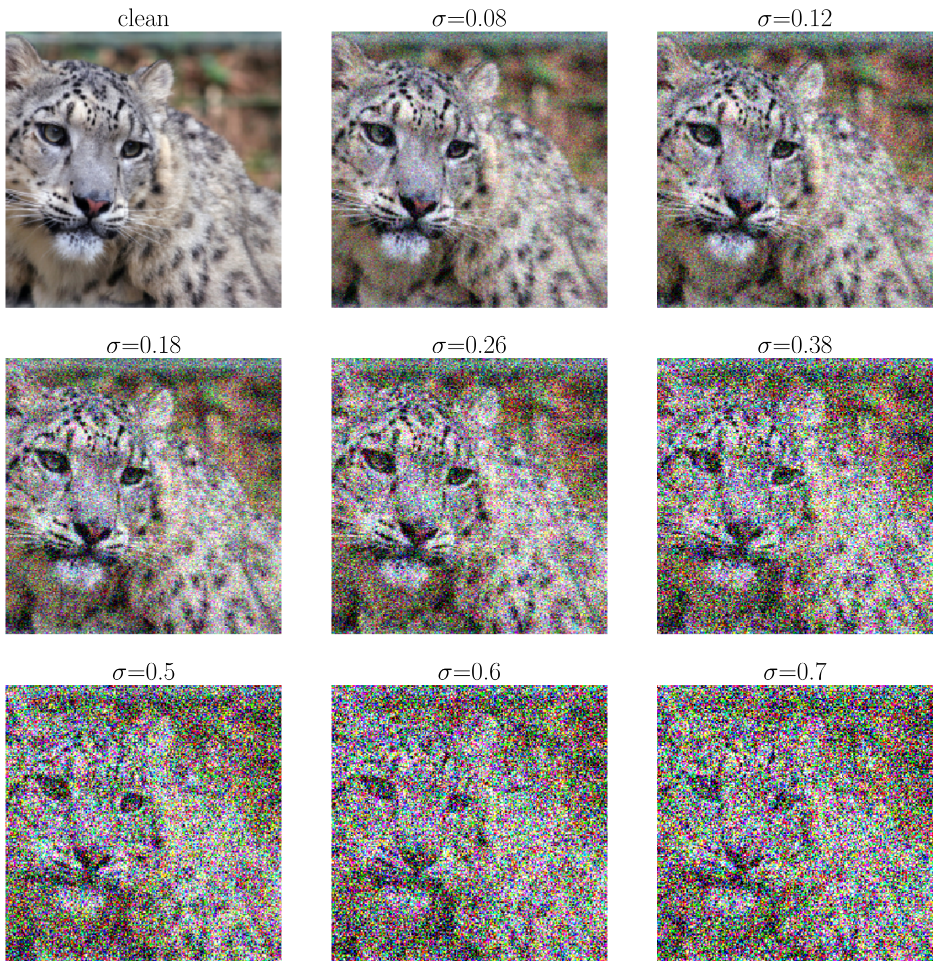

Fig. 3: Example images with different σ-levels of additive Gaussian noise on ImageNet.

H Comparison to Ford et al.

Ford et al. trained an InceptionV3 model from scratch both on clean data from the ImageNet dataset and on data augmented with Gaussian noise [1]. Since we use a very similar approach, we compare our approach to theirs directly. The results for comparison on ImageNet both for the vanilla and the Gaussian noise trained model can be found in Table 8. Since we use a pretrained model provided by PyTorch and fine-tune it instead of training a new one, the performance of our vanilla trained model differs from the performance of their vanilla trained model, both on clean data and on ImageNet-C. The accuracy on clean data is displayed in Table 9. Another difference between our training and theirs is that we split every batch evenly in clean and data augmented by Gaussian noise with one standard deviation whereas they sample σ uniformly between 0 and one specific value. With our training scheme, we were able to outperform their model significantly on all corruptions except for Elastic, Fog and Brightness.

		Noise (Compressed)			Blur (Compressed)			
model	All	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom
Vanilla InceptionV3 [1]	38.8	36.6	34.3	34.7	31.1	19.3	35.3	30.1
Gaussian ($\sigma = 0.4$) [1]	42.7	40.3	38.8	37.7	32.9	29.8	35.3	33.1
Vanilla InceptionV3 [ours]	41.6	42.0	40.3	38.5	33.5	27.1	36.1	28.8
GNT$\sigma_{0.4}$ [ours]	49.5	60.8	59.6	59.4	43.8	37.0	42.8	38.4
GNT$\sigma_{0.5}$ [ours]	**50.2**	**61.6**	**60.9**	**60.8**	**44.6**	**37.3**	**44.0**	**39.3**

	Weather (Compressed)				Digital (Compressed)			
model	Snow	Frost	Fog	Brightness	Contrast	Elastic	Pixelate	JPEG
Vanilla InceptionV3 [1]	33.1	34.0	52.4	66.0	35.9	47.8	38.2	50.0
Gaussian ($\sigma = 0.4$) [1]	36.6	43.5	**52.3**	**67.1**	35.8	**52.2**	47.0	55.5
Vanilla InceptionV3 [ours]	33.5	39.6	42.2	64.2	41.0	43.5	57.4	56.9
GNT$\sigma_{0.4}$ [ours]	35.6	43.7	43.3	64.8	43.0	49.0	59.3	61.7
GNT$\sigma_{0.5}$ [ours]	**37.1**	**44.2**	43.6	64.6	**43.3**	49.4	**59.6**	**61.9**

Table 8: ImageNet-C accuracy for InceptionV3.

model	clean accuracy [%]
Vanilla InceptionV3 [1]	75.9
Gaussian ($\sigma = 0.4$) [1]	74.2
Vanilla InceptionV3 [ours]	77.2
GNT$\sigma_{0.4}$ [ours]	78.1
GNT$\sigma_{0.5}$ [ours]	77.9

Table 9: Accuracy on clean data for differently trained models.

I Visualization of images with different perturbation sizes

In the main paper, we measure model robustness by calculating the median perturbation size ϵ^* and report the results in Table 2. To provide a better intuition for the noise level in an image for a particular ϵ^*, we display example images in Fig. 4.

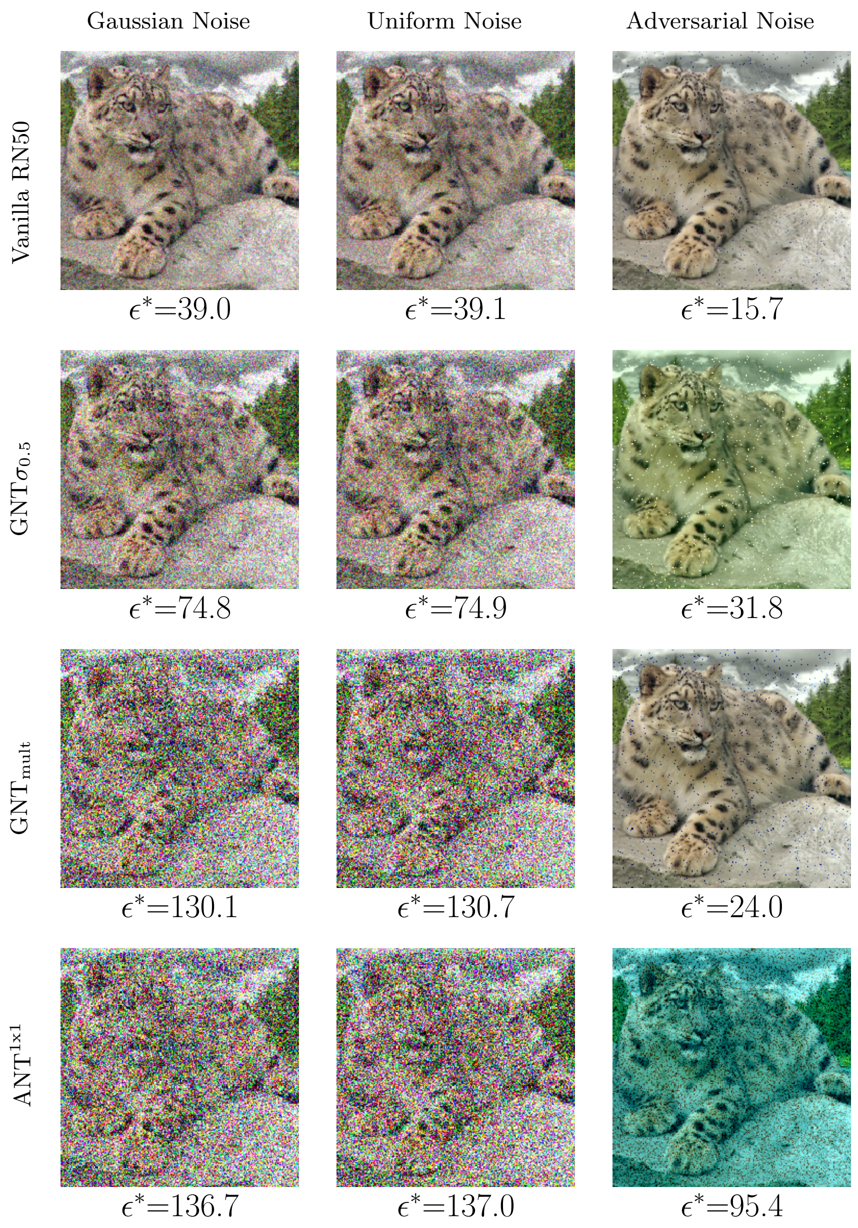

Fig. 4: Example images for the different perturbation sizes ϵ^* and different noise types on ImageNet corresponding to the ϵ^* values in Table 2 in the main paper.

J Visualization of Posterize vs JPEG

AugMix [4] uses Posterize as one of their operations for data augmentation during training. In Fig. 5, we show the visual similarity between the Posterize operation and the JPEG corruption from ImageNet-C.

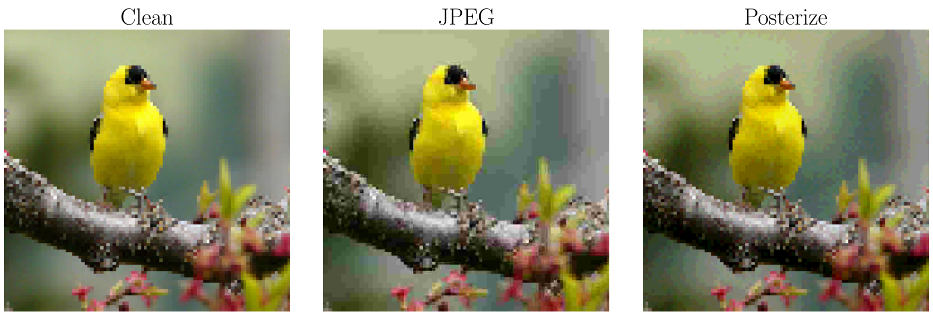

Fig. 5: Example images for the JPEG compression from ImageNet-C and the `PIL.ImageOps.Posterize` operation.

K Additional results

Adversarial Noise Training with a DenseNet121 architecture To test in how far our results generalize to other backbones, we have trained a DenseNet121 model with ANT^{1x1}. The DenseNet121 model was finetuned from the checkpoint provided by torchvision. A DenseNet121 has $7.97886 \cdot 10^6$ trainable parameters whereas a ResNet50 has $2.5557 \cdot 10^7$. Our results and a comparison to ANT^{1x1} with a ResNet50 model is shown in Table 10: ANT^{1x1} increases robustness on full ImageNet-C and ImageNet-C without noises for the DenseNet121 model, showing that adversarial noise training generalizes to other backbones.

	IN	IN-C		IN-C w/o noises	
model	clean acc.	Top-1	Top-5	Top-1	Top-5
Vanilla RN50	76.1	39.2	59.3	42.3	63.2
ANT^{1x1} RN50	76.0	(51.1)	(72.2)	47.7	68.8
Vanilla DN121	74.4	42.1	63.4	44.0	65.5
ANT^{1x1} DN121	74.3	50.3	71.6	46.8	68.3

Table 10: Average accuracy on clean data, average Top-1 and Top-5 accuracies on full ImageNet-C and ImageNet-C without the noise category (higher is better); all values in percent. We compare the results obtained by ANT^{1x1} for a ResNet50 (RN50) architecture to a DenseNet121 (DN121) architecture.

Results for different parameter counts of the noise generator Here, we study the effect of different parameter counts of the adversarial noise generator on ANT^{1x1}. We provide the results in Table 11. We indicate the depth of the noise generator with a subscript. All experiments in this paper apart from this ablation study were performed with a default depth of 4 layers. We observe that while depth is a tunable hyper-parameter, the performances of ANT^{1x1} with the studied noise generators do not differ by a lot. Only the most shallow noise generator with a depth of one layer and only 12 trainable parameters results in a roughly 1% lower accuracy than its deeper counterparts. We note that a GNT$\sigma_{0.5}$ model has an accuracy of 49.4% on full ImageNet-C and an accuracy of 47.1% on ImageNet-C without noises which roughly corresponds to the respective accuracies of ANT^{1x1} with the most shallow noise generator.

model	Number of parameters	IN clean acc.	IN-C Top-1	IN-C w/o noises Top-1
Vanilla RN50	-	76.1	39.2	42.3
ANT^{1x1} RN50 NG$_1$	12	75.1	(49.5)	46.6
ANT^{1x1} RN50 NG$_2$	143	75.5	(50.8)	47.2
ANT^{1x1} RN50 NG$_3$	563	75.3	(50.7)	47.2
ANT^{1x1} RN50 NG$_4$	983	76.0	(51.1)	47.7
ANT^{1x1} RN50 NG$_5$	1403	74.0	(50.7)	47.0

Table 11: Number of trainable parameters of different noise generators, average accuracy on clean data, ImageNet-C and ImageNet-C without the noise category (higher is better); all values in percent. We compare the results obtained by ANT^{1x1} with noise generators of different depth. Note that a depth of 4 layers was used in all experiments in this paper apart from this ablation study.

References

1. Ford, N., Gilmer, J., Carlini, N., Cubuk, D.: Adversarial examples are a natural consequence of test error in noise. ICML (2019)
2. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=Bygh9j09KX`
3. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=HJz6tiCqYm`
4. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. In: International Conference on Learning Representations (2020), `https://openreview.net/forum?id=S1gmrxHFvB`
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
7. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017)
8. Rony, J., Hafemann, L.G., Oliveira, L.S., Ayed, I.B., Sabourin, R., Granger, E.: Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4322–4330 (2019)
9. Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T.: Adversarial training for free! arXiv preprint arXiv:1904.12843 (2019)
10. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J.P., Davis, L.S., Goldstein, T.: Universal adversarial training. CoRR **abs/1811.11304** (2018), `http://arxiv.org/abs/1811.11304`
11. Wong, E., Schmidt, F.R., Metzen, J.H., Kolter, J.Z.: Scaling provable adversarial defenses. CoRR **abs/1805.12514** (2018), `http://arxiv.org/abs/1805.12514`
12. Xie, C., Wu, Y., van der Maaten, L., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. CVPR (2019)