Simple Post-Training Robustness using Test Time Augmentations and Random Forest Supplementary Material

Gilad Cohen
Tel Aviv University
giladco1@post.tau.ac.il

Raja Giryes
Tel Aviv University
raja@tauex.tau.ac.il

1 Appendix Structure

In Appendix 2 we present adversarial robustness scores of our approach using other architectures: Resnet-50 and Resnet-101, demonstrating that the combination of VAT+ARF achieves SOTA robustness as shown also for Resnet-34 in the main paper.

Appendix 3 defines in detail the image transformations we used to calculate the Test Time Augmentations (TTAs) in the paper, listing their parameter distribution and randomized ordering protocol.

Appendix 4 lists the hardware (CPUs & GPUs) we used for training our DNNs and random forest classifiers.

Appendix 5 shows several parameter searches we conducted to optimize our baselines (TRADES/VAT) to the datasets and architecture in our experiments.

Appendix 6 shows ARF performance when replacing the random forest classifier with majority voting scheme or other machine learning models: Logistic regression and SVM.

In Appendix 7 we show that our ARF model is transferable, generalizing very well to new (unseen) attacks.

Appendix 8 provides in depth details on the steps we implemented to utilize the BPDA attack in our experiments, since the non-differential random forest had to be mimicked by a substitute model.

Appendix 9 is a continuation to the TTA size ablation study conducted in Section 5.2 in the main paper. Here, we add the same ”accuracy vs size” results for CIFAR-100 and SVHN.

Appendix 10 shows the mean L_2 distortion for some attacks we used in the paper.

Lastly, Appendix 11 displays adversarial images generated by the adaptive white-box BPDA attack against our defense method. We show that albeit BPDA can circumvent our defense for a vanilla DNN (w/o TRADES/VAT), its generated noise can be observed by the naked eye.

2 Robustness on Resnet-50 and Resnet-101

Section 5 in the main paper shows adversarial robustness results only on Resnet-34. In this section we repeat the results also for Resnet-50 and Resnet-101, shown in Table 1 and Table 2, respectively. We omit the black-box Boundary attack from these experiments because it utilizes thousands of search queries, which is not practical for large DNN architectures. The black-box Square attack is reported since it is fast and efficient.

Overall, the results on these DNNs have the same trend as in Resnet-34. ARF’s robustness is on par with TRADES and VAT, however when combined with VAT we surpass the robustness of the vanilla adversarial trained DNN by a large margin. In addition, contrary to TRADES, ARF exhibits very high normal accuracy.

Table 1: Comparison of accuracies (%) for various classifiers on CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet trained on Resnet-50. All attacks are detailed in Section 4 in the main paper. We boldface the best results. Ensemble is presented just as a reference as it has an unfair advantage.

Dataset	Method	Normal	FGSM^1	FGSM^2	JSMA	PGD^1	PGD^2	Deepfool	CW$_{L2}$	CW$_{L\infty}$	Square
CIFAR-10	Plain	94.80	59.12	40.00	76.24	8.16	0.00	3.44	0.08	16.00	51.08
	Ensemble	95.92	78.04	53.00	86.48	79.84	22.00	93.56	85.92	79.48	86.20
	TRADES	86.56	84.40	75.44	71.72	84.52	70.68	8.36	0.24	78.40	81.52
	VAT	**95.16**	80.40	63.44	84.88	72.56	5.08	3.56	1.16	25.24	79.24
	TTA	90.80	78.64	59.92	84.64	83.52	57.08	87.36	82.80	81.48	82.04
	ARF	93.28	81.92	63.16	85.04	87.76	60.96	**91.36**	86.24	86.64	85.44
	TRADES + ARF	84.16	82.20	75.96	78.64	82.00	74.60	67.48	67.16	80.16	80.20
	VAT + ARF	93.24	**89.36**	**77.64**	**90.76**	**90.48**	**78.36**	89.84	**88.00**	**87.32**	**88.80**
CIFAR-100	Plain	74.52	26.48	11.08	47.64	16.52	0.08	9.44	8.08	36.56	24.52
	Ensemble	78.44	53.60	23.08	56.72	64.48	32.64	76.88	53.84	62.40	60.36
	TRADES	55.08	53.92	44.60	45.92	54.08	48.64	11.08	5.52	52.64	47.96
	VAT	**71.56**	52.56	28.84	60.04	63.88	13.76	9.72	6.40	40.52	52.96
	TTA	67.32	49.60	26.56	54.84	60.76	46.08	64.20	49.52	58.00	52.56
	ARF	69.24	54.08	31.40	57.40	64.88	52.28	**65.68**	53.80	**62.00**	56.36
	TRADES + ARF	50.68	49.80	45.20	47.04	49.32	46.64	46.56	34.60	50.20	47.84
	VAT + ARF	67.28	**62.72**	**48.84**	**65.60**	**65.36**	**59.88**	64.24	**56.16**	61.92	**62.40**
SVHN	Plain	97.28	81.48	65.12	51.64	51.32	0.88	2.36	2.60	20.24	67.08
	Ensemble	97.88	91.32	75.28	85.48	92.92	71.28	84.80	85.72	85.92	92.72
	TRADES	93.72	92.92	**89.76**	52.36	92.00	77.76	4.24	0.40	89.08	84.44
	VAT	96.52	87.96	80.92	84.88	48.28	0.20	2.24	0.36	9.36	81.68
	TTA	**97.24**	89.32	75.04	86.08	91.76	63.92	83.44	86.08	84.52	93.00
	ARF	97.16	89.68	76.72	87.40	92.48	67.44	84.28	86.56	86.36	92.72
	TRADES + ARF	93.72	93.24	89.52	84.88	92.76	83.92	74.60	48.72	90.36	89.32
	VAT + ARF	96.72	**95.28**	87.96	**94.24**	**95.68**	**87.96**	**96.28**	**95.44**	**95.08**	**94.76**
Tiny ImageNet	Plain	64.16	25.68	11.68	32.92	30.60	0.08	9.52	17.28	41.60	28.84
	Ensemble	69.08	56.48	28.12	52.44	65.68	46.92	68.24	51.40	61.84	58.60
	TRADES	45.60	45.12	33.92	37.60	44.68	41.12	10.48	6.36	43.84	42.08
	VAT	**63.24**	53.64	40.96	**56.64**	23.92	0.52	9.76	18.12	54.28	41.12
	TTA	51.88	36.88	19.08	40.16	43.36	30.20	45.92	33.08	42.80	38.44
	ARF	54.00	41.00	22.64	43.44	47.84	35.56	49.40	37.36	45.56	43.96
	TRADES + ARF	37.52	36.32	31.08	35.48	36.36	33.60	33.52	25.64	37.80	36.80
	VAT + ARF	56.00	**54.04**	**48.16**	54.44	**56.04**	**52.84**	**56.64**	**48.88**	**56.76**	**50.08**

Table 2: Comparison of accuracies (%) for various classifiers on CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet trained on Resnet-101. All attacks are detailed in Section 4 in the main paper. We boldface the best results. Ensemble is presented just as a reference as it has an unfair advantage.

Dataset	Method	Normal	FGSM^1	FGSM^2	JSMA	PGD^1	PGD^2	Deepfool	CW$_{L2}$	CW$_{L\infty}$	Square
CIFAR-10	Plain	94.96	58.08	44.12	77.92	9.04	0.00	3.60	1.32	24.24	56.32
	Ensemble	96.24	78.76	55.56	87.48	77.52	14.36	93.52	83.08	81.16	87.60
	TRADES	85.04	82.76	72.80	70.20	82.88	68.08	8.56	0.16	75.08	77.56
	VAT	**94.36**	80.24	64.40	83.84	76.76	10.96	3.60	2.12	36.28	82.68
	TTA	91.52	77.92	58.96	84.80	83.04	55.24	86.36	81.60	80.60	83.16
	ARF	93.64	81.64	63.44	84.88	88.00	60.48	**91.20**	83.96	86.48	86.36
	TRADES + ARF	82.12	81.00	75.24	77.24	81.16	74.04	66.00	67.12	78.88	79.00
	VAT + ARF	93.40	**88.92**	**76.52**	**90.60**	**89.76**	**76.36**	88.88	**85.48**	**87.84**	**89.00**
CIFAR-100	Plain	75.56	33.20	17.56	53.44	16.72	0.12	9.40	13.52	46.92	27.64
	Ensemble	79.56	60.48	29.84	59.52	69.64	42.60	78.12	54.96	72.00	63.72
	TRADES	55.52	54.68	44.28	46.64	54.60	49.88	10.36	4.52	53.08	47.32
	VAT	**74.28**	55.08	31.32	59.72	64.80	10.56	9.44	8.84	41.12	52.12
	TTA	68.48	51.16	32.60	59.04	62.12	48.16	65.16	48.96	62.16	56.48
	ARF	70.60	55.40	35.92	61.20	66.16	53.16	**67.64**	53.80	**66.48**	59.52
	TRADES + ARF	50.88	49.76	44.72	46.32	49.40	45.88	43.68	35.00	49.68	46.96
	VAT + ARF	70.52	**65.16**	**48.72**	**66.12**	**67.48**	**60.36**	66.56	**56.84**	62.92	**64.52**
SVHN	Plain	97.48	80.12	62.12	57.72	53.00	1.52	2.32	4.88	33.36	71.32
	Ensemble	98.08	90.04	72.04	87.80	92.72	64.80	84.12	83.12	88.12	93.24
	TRADES	93.52	93.12	89.32	38.88	92.40	74.12	4.60	0.68	87.92	83.60
	VAT	94.64	89.60	86.24	75.72	71.20	12.64	3.32	12.64	56.32	78.72
	TTA	97.16	87.92	71.00	86.56	90.36	54.92	80.60	81.92	85.36	93.36
	ARF	**97.24**	88.80	71.72	87.60	91.08	61.56	82.68	82.32	86.28	**93.68**
	TRADES + ARF	94.04	93.32	**89.88**	79.52	93.28	**82.56**	72.68	54.36	90.48	88.64
	VAT + ARF	95.52	**93.80**	89.04	**93.32**	**94.32**	82.28	**95.72**	**91.08**	**92.24**	92.04
Tiny ImageNet	Plain	66.56	25.88	11.76	34.48	30.44	0.24	9.08	14.56	40.60	32.00
	Ensemble	70.20	55.24	27.92	53.16	65.96	46.12	68.80	53.16	61.88	61.84
	TRADES	45.28	43.88	35.72	37.80	44.36	40.92	10.96	5.04	44.20	41.64
	VAT	64.40	43.96	23.84	**53.24**	48.92	4.68	10.16	26.80	51.84	46.44
	TTA	54.84	39.08	19.52	43.48	46.00	29.96	49.96	34.84	44.76	42.92
	ARF	**57.12**	41.80	22.16	45.80	50.72	35.04	51.28	37.76	47.68	46.52
	TRADES + ARF	35.64	34.04	30.48	34.56	33.88	31.88	30.56	20.88	35.76	32.88
	VAT + ARF	56.12	**50.40**	**36.72**	53.00	**53.92**	**49.68**	**52.88**	**46.40**	**53.32**	**50.12**

3 Test-time Augmentations

Here we detail all the transforms we used to generate our Test-Time Augmentations (TTAs) for our robust classification methods, as described in Section 3.2 in the main paper. We used different parameters for *soft* transforms and *hard* transforms in the ablation study in Section 5.2. Both sets of parameters are listed below. The main result in the paper, outside the aforementioned ablation study, were calculated only with the *hard* set, which proved to achieve better performance in the ablation study. We denote the original and transformed images as x and x_t, respectively.

1. Rotation: Angle rotation of the image was randomized to $U(-8°, 8°)$ for *soft* and $U(-15°, 15°)$ for *hard*.
2. Translation: The image was allowed to shift horizontally and vertically up to 2 pixels in every direction for CIFAR-10, CIFAR-100, and SVHN, and up to 4 pixels for Tiny ImageNet. This transform behaves similarly for both *soft* and *hard*.
3. Scale: We randomly selected a zoom in ($s > 1$) or a zoom out ($s < 1$). The image was scaled with $s \sim U(0.95, 1.05)$ for *soft* and $s \sim U(0.9, 1.1)$ for *hard*.
4. Mirror: The image was horizontally flipped with a probability of 0.5. This transform was omitted for SVHN dataset, and was the same for *soft* and *hard*.
5. Brightness: Randomly increase/decrease brightness. Let b denote the brightness factor; the transforms is defined as $x_t = b \cdot x$. We randomized $b \sim U(0.8, 1.2)$ for *soft* and $b \sim U(0.6, 1.4)$ for *hard*.
6. Contrast: The contrast factor c was distributed as $c \sim U(0.85, 1.15)$ for *soft* and as $c \sim U(0.7, 1.3)$ for *hard*. The transformed image after contrast is: $x_t = c \cdot x + (1 - c) \cdot \mathbb{E}(x_G) \cdot \mathbb{1}_{nxnx3}$, where $\mathbb{E}(x_G)$ is the mean pixel value on the gray-scale equivalent image and $\mathbb{1}_{nxnx3}$ is a matrix as the size of the original image, filled with ones. The gray-scale image is defined as: $x_G = 0.2989 \cdot R + 0.587 \cdot G + 0.114 \cdot B$ where (R, G, B) are the red, green, and blue channels of x, respectively.
7. Saturation: The saturation factor sat was distributed as $sat \sim U(0.75, 1.25)$ for *soft* and as $sat \sim U(0.5, 1.5)$ for *hard*. It is defined as: $x_t = sat \cdot x + (1 - sat) \cdot x_G$.
8. Hue: The hue factor h was distributed as $h \sim U(0.03, 0.03)$ for *soft* and as $h \sim U(0.06, 0.06)$ for *hard*. The transform updates the hue in the Hue Saturation Value (HSV) representation by h.
9. Gamma: Applying gamma transform on the image. Each channel (r,g,b) on x is transformed to $x_t[r, g, b] = x[r, g, b]^\gamma$, where $\gamma \sim U(0.85, 1.15)$ for *soft* and $\gamma \sim U(0.7, 1.3)$ for *hard*.
10. Blur: The blur transform convolutes the image with a 2D Gaussian kernel: $x_t = G_{2D}(u, v; \sigma_b) * x$, where $G_{2D}(u, v; \sigma_b) = \frac{1}{2\pi\sigma_b^2} \exp \frac{-(u^2+v^2)}{2\sigma_b^2}$, where σ_b is uniformly distributed between 0.001 and a positive constant value σ_{bmax}: $\sigma_b \sim U(0.001, \sigma_{bmax})$.

    We set $\sigma_{bmax} = 0.25$ for *soft* and $\sigma_{bmax} = 0.5$ for *hard*.
11. Noise: The Noise transform adds a white Gaussian noise to the image, $x_t = x + n$, where n is sampled from $n \sim N(0, \sigma)$. The standard deviation of the normal distribution is randomized in our algorithm to be $\sigma \sim U(0, \sigma_{max})$. We set $\sigma_{max} = 0.005$ in all our experiments (see Section 5.2 in the main paper).

It is important to point out that for all the color transforms, geometric transforms (except Mirror) and Noise, the mean value of the transform change is zero, thus our generated TTAs are unbiased.

The transforms were carried out in the following order:

A) Applying all the color transforms ([5]-[9]). The order of the color transforms was randomized.

B) Padding the image with the last value at the edge of the image. CIFAR-10, CIFAR-100, and SVHN were padded to 64x64x3 and Tiny ImageNet was padded to 128x128x3.

C) Applying the random affine transform (transforms [1]-[3]).

D) Blurring the image ([10]).

E) Croppong the center of the image.

F) Applying random horizontal flip (not for SVHN) ([4]).

G) Adding noise ([11]).

Some of our transforms were implemented using the TorchVision package of PyTorch [Paszke et al., 2019].

4 Hardware Setup

We trained our Deep Neural Networks (DNNs), Resnet-34, Resnet-50, and Resnet-101, with a GPU of type NVIDIA GeForce RTX 2080 Ti. This GPU has 11 GB of VRAM. We used multi workers setup and utilized 4 threads of Intel Xeon Silver 4114 CPU.

All the adversarial training with TRADES [Zhang et al., 2019] required more memory, therefore these DNNs were trained on a different server using NVIDIA RTX A6000 GPU that has 48 GB of VRAM. For training with TRADES we used 4 threads of Intel Xeon Gold 5220R CPU.

All the attacks listed in Section 4 in the paper, including the adapted attacks, were carried out on a single NVIDIA GeForce RTX 2080 Ti GPU. All the DNNs training, adversarial attacks, and evaluations were done using a single GPU.

The TTAs were generated on the CPU alone. After generated them, we fed them to the DNNs with a single forward pass (of 256 TTAs).

The random forest classifier was trained using 20 threads of Intel Xeon Silver 4114 CPU.

5 Adversarial Training

We trained some TRADES models for fewer train epochs since we observed this yields more robust classifiers. The normal and adversarial accuracies on CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet trained on Resnet-34 using TRADES , is shown in Table 3. The attacks listed in the table are PGD($L_\infty, \epsilon = 0.01$), PGD($L_\infty, \epsilon = 0.031$), and CW($L_\infty, \epsilon = 0.031$), which are defined in Section 4 in the main paper, abbreviated to PGD^1, PGD^2, and CW$_{L_\infty}$, respectively. Based on these results we trained all the adversarial robust TRADES DNN with 100, 100, 100, and 300 epochs for CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet, respectively. The only exception was training Tiny ImageNet on Resnet-101 with TRADES which was very time consuming, therefore we trained it only for 100 epochs instead of 300 epochs.

Table 3: Normal and adversarial accuracies (%) for adversarially robust DNNs trained with TRADES on Resnet-34, for various number of epochs.

Dataset	Epochs	Normal	PGD^1	PGD^2	CW$_{L_\infty}$
CIFAR-10	100	86.68	**85.12**	**71.88**	**78.28**
	200	**87.08**	84.00	67.96	74.36
	300	86.92	84.28	68.92	75.12
CIFAR-100	100	**53.36**	**52.88**	46.44	**51.04**
	200	53.00	52.00	**47.28**	50.20
	300	53.00	52.32	47.16	50.04
SVHN	100	**92.48**	**90.28**	**70.88**	**82.36**
	200	91.64	84.64	41.04	56.16
Tiny ImageNet	100	41.68	41.04	37.48	40.20
	200	43.88	43.08	37.96	**42.60**
	300	**44.44**	**43.72**	**38.44**	41.88

We trained the VAT models with the same number of epochs as the vanilla Resnets: 300, 300, 200, and 300 epochs for CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet, respectively. Unlike TRADES, the VAT robustness did not degrade in the late epochs, as shown in Table 4.

To optimize the VAT training, we set $\alpha = 1$ as suggested in [Miyato et al., 2016], and experimented with different values of ϵ, as listed in Table 5. Based on these results, we chose to optimize VAT for max robustness on PGD^2 and thus selected ϵ=1,

Table 4: Normal and adversarial accuracies (%) for adversarially robust DNNs trained with VAT on Resnet-34, for various number of epochs.

Dataset	Epochs	Normal	PGD^1	PGD^2	CW$_{L_\infty}$
CIFAR-10	100	93.04	81.12	15.00	34.32
	200	**94.00**	81.60	19.44	49.68
	300	**94.00**	**82.12**	**20.08**	**49.80**
CIFAR-100	100	66.88	58.48	12.12	34.24
	200	70.84	62.12	14.96	42.44
	300	**70.92**	**63.00**	**15.20**	**44.36**
SVHN	100	81.92	63.77	19.23	45.36
	200	**94.90**	**85.00**	**42.35**	**64.68**
Tiny ImageNet	100	50.69	49.03	29.90	41.84
	200	54.06	51.86	32.28	45.38
	300	**54.67**	**52.20**	**32.48**	**45.94**

1, 3, 1 for CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet, respectively.

Table 5: Normal and adversarial accuracies (%) for adversarially robust DNNs trained with VAT on Resnet-34, for various number of ϵ values, as defined in [Miyato et al., 2016].

Dataset	ϵ	Normal	PGD^1	PGD^2	CW$_{L_\infty}$
CIFAR-10	0.5	**95.16**	66.44	1.92	27.52
	1	94.00	**82.12**	**20.08**	**49.80**
	2	94.76	26.28	0.16	22.72
	8	89.80	72.08	11.28	34.28
CIFAR-100	0.5	**74.48**	54.80	3.32	43.08
	1	70.92	**63.00**	**15.20**	**44.36**
	2	70.48	36.68	1.48	35.64
	8	62.28	55.68	15.16	31.20
SVHN	0.5	**95.65**	82.70	18.35	44.87
	1	95.19	53.17	5.95	**77.18**
	3	94.90	**85.00**	**42.35**	64.68
Tiny ImageNet	0.5	57.82	53.51	17.71	42.32
	1	54.67	**52.20**	**32.48**	**45.94**
	2	**58.11**	38.50	1.40	42.39
	8	53.67	47.58	7.30	43.01

6 Alternative Classifiers

Our ARF algorithm trains the random forest classifier using the TTA output logits (see Section 3.2 in the main paper). Instead of training a random forest, we explored four alternative models: A simple (parametric-free) voting on the TTA logits predictions, Logistic regression, linear SVM, and SVM with an RBF kernel. Since our datasets are multi class, we set the classification strategy to be one-vs-rest for all the proposed parametric models. Table 6 shows results of normal and adversarial accuracies on CIFAR-10, trained on Resnet-34, and attacked by all the non adaptive attacks described in Section 4 in the main paper.

We observe that the random forest classifier achieves much better performance than voting and the linear classifiers, and overall it is slightly better than SVM with RBF kernel. In addition, the ARF achieves the highest normal accuracy among all the classifiers we tested. Since SVM with RBF has approximately the same computation run time as random forest, there is no reason to favor it over random forest.

Table 6: Normal and adversarial accuracies (%) on CIFAR-10 when training logistic regression or SVM compared to our proposed random forest classifier.

Classifier	Normal	FGSM^1	FGSM^2	JSMA	PGD^1	PGD^2	Deepfool	CW$_{L_2}$	CW$_{L_\infty}$	Square	Boundary
Voting	91.76	82.48	68.60	84.88	87.08	73.32	83.40	82.32	81.76	85.64	88.40
Logistic regression	92.68	82.40	68.72	82.44	86.24	72.32	84.68	81.20	80.36	83.88	89.60
SVM (linear)	89.40	79.76	67.48	79.84	81.31	65.16	79.12	76.92	75.52	80.24	87.60
SVM (RBF)	93.52	83.52	69.76	84.68	89.76	**78.16**	**87.36**	**84.80**	85.28	87.08	**91.60**
Random forest	**93.76**	**83.72**	**70.20**	**85.28**	**90.32**	77.88	**87.36**	84.36	**85.64**	**87.84**	91.20

7 Transferability

We show that our ARF defense is characterized with excellent transferability, being able to generalize to new (unseen) attacks. Table 7 compares between two different setups of fitting and testing ARF. The top row shows the accuracies we presented in Table 1 in the main paper, where we train ARF on all the attacks (FGSM^1, FGSM^2, JSMA, PGD^1, PGD^2, Deepfool, CW$_{L_2}$, CW$_{L_\infty}$, Square, and Boundary) with a global random forest model, obtained by fitting it on all the aforementioned attacks. The second row shows ARF accuracy using the Leave-One-Out Cross-Validation (LOOCV) procedure, where we fit the random forest on all the attacks except the attack we wish to test it on. For example, we calculate the adversarial accuracies on images generated by FGSM^1 and FGSM^2 after fitting the random forest on images generated by JSMA, PGD^1, PGD^2, Deepfool, CW$_{L_2}$, CW$_{L_\infty}$, Square, and Boundary. Table 8 lists explicitly which attacks were used to fit the random forest for each tested attack. Since this cross-validation method fits seven random forest models, the displayed normal accuracy is their calculated mean and standard deviation.

Table 7: Normal and adversarial accuracies (%) on CIFAR-10 using different setups for fitting the random forest. The top row is the case, where the random forest is fitted and tested on all the non adaptive attacks. The bottom row shows results for the Leave-One-Out Cross-Validation (LOOCV) method, where the tested attack is excluded from the random forest fitting.

Random forest fitting	Normal	FGSM^1	FGSM^2	JSMA	PGD^1	PGD^2	Deepfool	CW$_{L_2}$	CW$_{L_\infty}$	Square	Boundary
Global	93.76	83.72	70.20	85.28	90.32	77.88	87.36	84.36	85.64	87.84	91.20
LOOCV	93.71 ± 0.07	83.48	69.48	84.92	90.16	78.16	87.32	84.92	85.24	87.92	91.60

Table 8: Each row displays which attacks were employed to fit the random forest on the tested attack using the LOOCV procedure.

Tested attack	FGSM^1	FGSM^2	JSMA	PGD^1	PGD^2	Deepfool	CW$_{L_2}$	CW$_{L_\infty}$	Square	Boundary
FGSM^1			✓	✓	✓	✓	✓	✓	✓	✓
FGSM^2			✓	✓	✓	✓	✓	✓	✓	✓
JSMA	✓	✓		✓	✓	✓	✓	✓	✓	✓
PGD^1	✓	✓	✓			✓	✓	✓	✓	✓
PGD^2	✓	✓	✓			✓	✓	✓	✓	✓
Deepfool	✓	✓	✓	✓	✓		✓	✓	✓	✓
CW$_{L_2}$	✓	✓	✓	✓	✓	✓			✓	✓
CW$_{L_\infty}$	✓	✓	✓	✓	✓	✓			✓	✓
Square	✓	✓	✓	✓	✓	✓	✓	✓		✓
Boundary	✓	✓	✓	✓	✓	✓	✓	✓	✓	

8 BPDA Attack

In the white-box settings, we utilized the Backward Pass Differentiable Approximation (BPDA) attack in [Athalye et al., 2018a]. More specifically, we employed the generalized BPDA and replaced the non-differential random forest model with a substitute model that can derive gradients. To that end we used knowledge distillation [Hinton et al., 2015] to train a substitute MLP to mimic the random forest functionality. We used a six layer MLP with the following dimensions:

- For CIFAR-10 and SVHN: $N \cdot C \rightarrow N \cdot C \rightarrow \frac{N \cdot C}{2} \rightarrow \frac{N \cdot C}{4} \rightarrow \frac{N \cdot C}{8} \rightarrow \frac{N \cdot C}{16} \rightarrow C$.
- For CIFAR-100 and Tiny ImageNet: $N \cdot C \rightarrow \frac{N \cdot C}{10} \rightarrow \frac{N \cdot C}{20} \rightarrow \frac{N \cdot C}{40} \rightarrow \frac{N \cdot C}{80} \rightarrow \frac{N \cdot C}{160} \rightarrow C$.

N and C are the TTA size and the number of classes in the dataset, respectively. Since CIFAR-100 and Tiny-ImageNet have higher number of logits, we shrink their layer size in the MLP faster to limit the number of parameters.

After each linear layer we used a batch normalization layer and Relu activation (in this order), except for the last layer. We trained this MLP with TTA logits obtained solely from the *test* set (used for fitting the random forest), and kept the *test-val* hidden. We train this MLP using the KL divergence loss; we did not add the cross entropy loss (with the ground truth label) to the training since our goal is to mimic the random forest gradients with the highest fidelity, and not to improve classification accuracy.

After the MLP was trained, we used the BPDA attack on the hybrid model encapsulating the original DNN and the substituted MLP, connected in tandem. All our robustness results, including on the BPDA attack, show accuracy calculated for adversarial imaged generated from the *test-val* set. It should be emphasized that the above hybrid model (DNN+MLP) was only used to generate the adversarial images. For evaluating robustness we pass these images to the ARF model (DNN with random forest).

The EoT attack [Athalye et al., 2018b] was not used in our experiments because this attack requires a white-box threat model with differential loss. Thus, we could not differentiate the expected value of a loss at the output of the random forest, over our TTA transform distribution. In any rate, we showed in the paper that the transformations alone do not provide protection against adaptive white-box attacks, since our A-PGD attack greatly attenuates the ARF robustness (see Section 5.3 in the main paper).

9 TTA Size Ablation

Here we repeat the TTA size ablation test in Section 5.2 in the main paper for CIFAR-100 and SVHN datasets. We plot the adversarial accuracies for PGD[1], Deepfool, and CW$_{L_2}$ in a logarithmic scale (Figure 1), and show that $N = 256$ TTAs are sufficient also for these datasets.

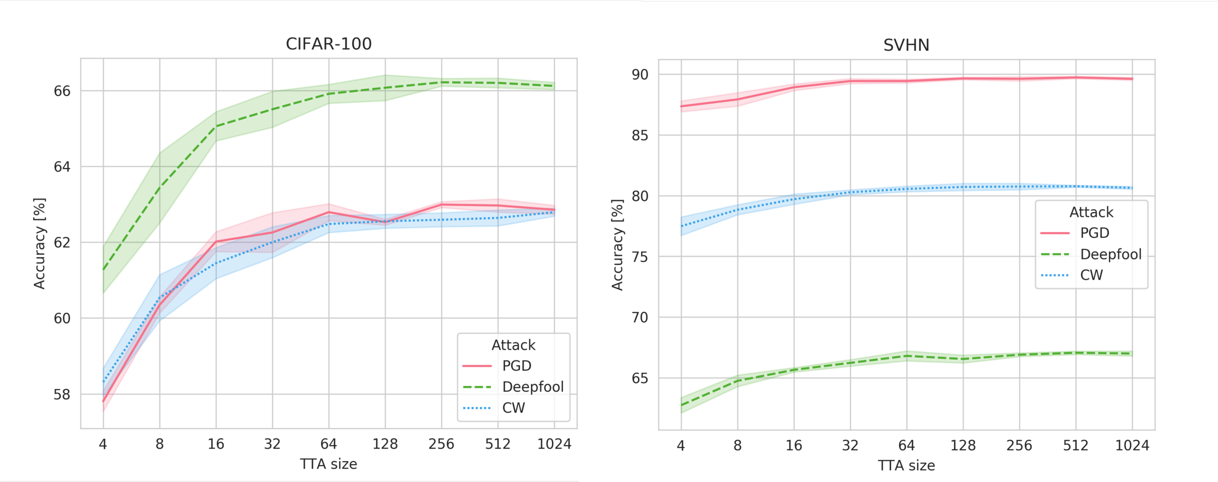


Figure 1: Ablation study on the number of generated TTAs (N). We calculate the adversarial accuracies on CIFAR-100 and SVHN for three attacks as a function of N (logarithmic scale).

10 L_2 Distortion

Table 9 reports the mean L_2 distortion of the adversarial images, generated by Deepfool (gray-box), CW$_{L_2}$ (gray-box), A-PGD (adaptive gray-box), Boundary (black-box), and BPDA (white-box) attacks. The L_2 distortion of an adversarial image x' from the natural image x is defined by $d_{L_2}(x, x') = ||x - x'||_2$. In our average distortion calculation we consider only adversarial images that fooled the defense, meaning, the DNN classified x correctly but the defense method misclassified x'.

This measure is interesting because high L_2 distortion value correlates to perceptible noises on the images, thus invalidates the attack since humans can easily notice it. The above is relevant especially for the unbounded attacks, Deepfool, CW$_{L_2}$, and Boundary (in our experiments), which are not constrained by the L_2 norm. We observe that in the majority of cases, the highest mean L_2 distortion was obtained by using TRADES, either alone or combined with ARF.

Table 9: Mean L_2 distortion values for adversarial images generated by selected attacks, on a vanilla Resnet34 (Plain), adversarially trained Resnet34 (TRADES/VAT), our ARF defense, and a combination of TRADES/VAT with our ARF.

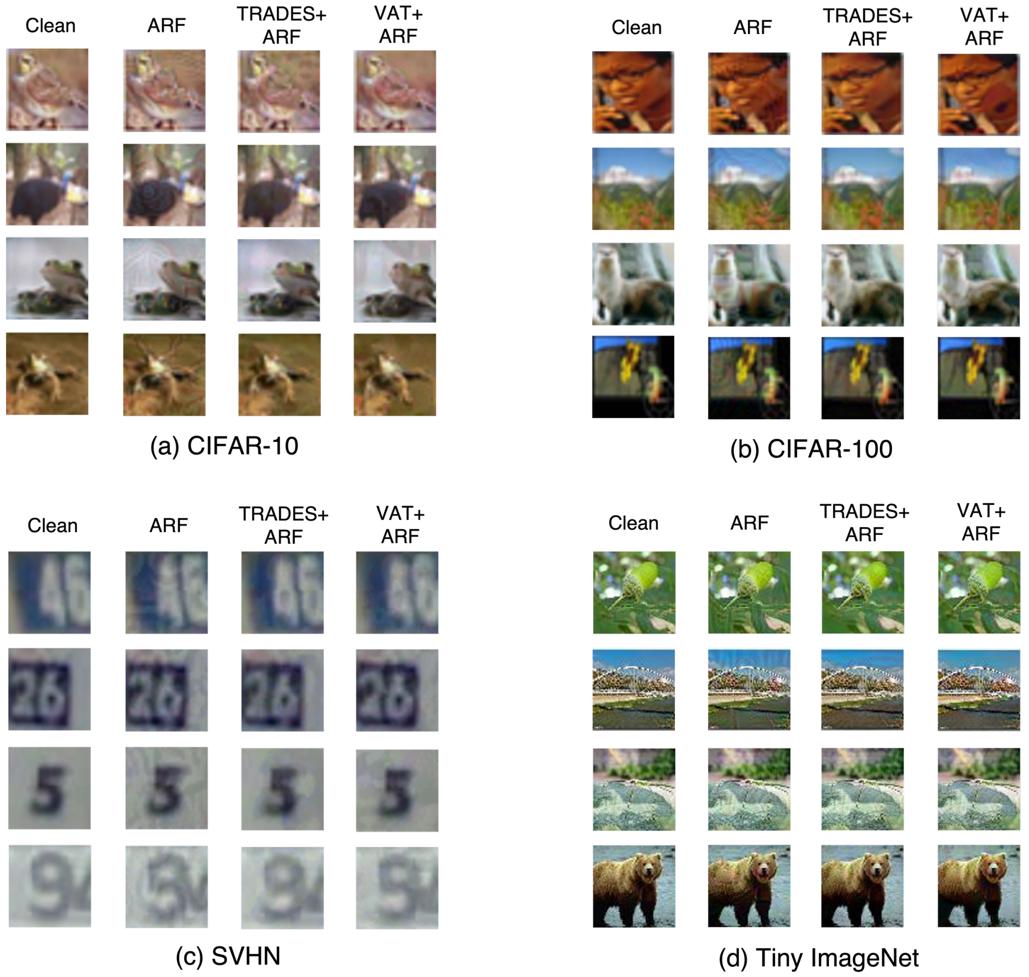
Dataset	Attack	Plain	TRADES	VAT	ARF	TRADES+ ARF	VAT+ ARF
CIFAR-10	Deepfool	0.58	**1.48**	1.46	0.30	1.08	0.56
	CW$_{L_2}$	0.80	1.99	1.67	2.31	1.98	**2.79**
	Boundary	0.25	**1.47**	0.89	0.11	1.37	0.68
	A-PGD	1.36	1.60	1.52	1.35	**1.61**	1.52
	BPDA	1.27	1.36	1.36	1.27	**1.38**	1.36
CIFAR-100	Deepfool	0.20	**0.85**	0.54	0.07	0.50	0.20
	CW$_{L_2}$	1.67	2.92	1.89	2.51	**3.01**	2.31
	Boundary	0.41	**2.29**	1.07	0.25	1.99	0.77
	A-PGD	1.40	**1.61**	1.54	1.40	**1.61**	1.54
	BPDA	1.24	**1.41**	1.36	1.24	1.39	1.34
SVHN	Deepfool	**1.29**	1.07	0.94	1.18	1.09	0.51
	CW$_{L_2}$	1.10	1.37	1.52	**2.50**	1.32	2.01
	Boundary	0.43	0.97	**1.18**	0.16	0.61	0.63
	A-PGD	1.35	**1.57**	1.46	1.34	1.54	1.47
	BPDA	1.22	**1.36**	1.31	1.21	**1.36**	1.32
Tiny ImageNet	Deepfool	0.40	**1.39**	1.22	0.20	0.81	0.73
	CW$_{L_2}$	2.19	**5.22**	4.80	2.39	4.23	4.75
	Boundary	1.49	**7.22**	5.34	1.03	2.65	3.07
	A-PGD	2.86	3.19	3.18	2.87	**3.20**	3.17
	BPDA	2.44	2.80	**2.86**	2.44	2.82	2.83

11 Visual Perceptibility

In Section 5.3 in the main paper, we have shown that our ARF defense is susceptible to the BPDA attack, an adaptive white-box attack that was customly tailored to circumvent our specific random forest classifier. We turn now to show that this attack fails to generate imperceptible images. We display some images generated using BPDA against ARF and demonstrate that a human observer can easily detect an unusual distortion in them.

Figure 4 in the main paper (also presented here as Figure 2 for convinience) exhibits clean images and adversarial images generated by BPDA for CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet. ”Clean” column corresponds to natural (undistorted) images; ”ARF” column denotes images that fool our ARF defense; ”TRADES+ARF” and ”VAT+ARF” columns display images that fool our ARF defense when combined with TRADES and VAT adversarially trained DNNs, respectively. For a fair comparison, we show only images that successfully fool all the three defenses, meaning, the DNN classified the clean image successfully but the adversarial image was able to flip the label despite our random forest classifier.

We note that the most visible noises correspond to attacks on the vanilla ARF method, without incorporating TRADES/VAT into it. This observation is counterintuitive to the reported accuracies in Section 5.3 in the main paper that show better robustness of ARF when combined with TRADES/VAT. In addition, Section 10 exhibits lower L_2 distortion for the vanilla ARF defense on the BPDA compared to TRADES+ARF and VAT+ARF. Nonetheless, these visible distortions decrease the efficacy of BPDA towards our defense. It also shows that although ARF when used alone gets lower quantitative results, qualitatively it has an advantage over the other methods as the examples that overcome it can be easily spotted by a naked eye.



(a) CIFAR-10

(b) CIFAR-100

(c) SVHN

(d) Tiny ImageNet

Figure 2: Adversarial images generated by BPDA circumventing our ARF defense. TRADES+ARF and VAT+ARF correspond to our ARF defense when applied on top of an adversarially trained DNN, TRADES/VAT, respectively. Adversarial images that fool our ARF defense can be easily spotted by the naked eye.

References

[Athalye et al., 2018a] Athalye, A., Carlini, N., and Wagner, D. A. (2018a). Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*.

[Athalye et al., 2018b] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018b). Synthesizing Robust Adversarial Examples. *ArXiv*, abs/1707.0.

[Hinton et al., 2015] Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *ArXiv*, abs/1503.0.

[Miyato et al., 2016] Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. (2016). Distributional Smoothing with Virtual Adversarial Training. In *ICLR*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Zhang et al., 2019] Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L., and Jordan, M. I. (2019). Theoretically Principled Trade-off between Robustness and Accuracy. In *ICML*.