Towards Evaluating Transfer-based Attacks Systematically, Practically, and Fairly (Supplementary Material)

Anonymous Author(s)
Affiliation
Address
email

1 ℓ_2 Results

Table 3: Comparing the obtained AA and AAA of some "gradient computation" and "substitute model training" methods. Smaller values indicate more powerful attacks. The adversarial examples were generated under an ℓ_2 constraint with $\epsilon = 5$.

	ResNet -50	VGG -19	Inception v3	EffNetV2 -M	ConvNeXt -B	ViT -B	DeiT -B	BEiT -B	Swin -B	Mixer -B	AAA
I-FGSM Back-end											
- Baseline											
I-FGSM	87.93%	91.82%	94.76%	97.24%	88.96%	91.01%	90.64%	90.18%	95.46%	95.10%	92.31%
- Gradient Computation											
TAP (2018) [18]	88.91%	94.44%	95.29%	98.30%	94.47%	94.94%	95.56%	94.91%	96.89%	96.49%	95.02%
NRDM (2018) [8]	91.41%	92.36%	96.00%	98.94%	95.28%	97.00%	97.14%	97.63%	97.26%	95.43%	95.85%
FDA (2019) [1]	92.24%	96.48%	96.02%	99.17%	96.74%	97.58%	96.78%	96.73%	98.01%	98.38%	96.81%
ILA (2019) [4]	83.49%	84.09%	92.43%	96.31%	91.08%	89.07%	88.32%	88.28%	94.16%	93.30%	90.05%
SGM (2020) [15]	78.82%	-	-	94.68%	82.52%	89.46%	89.79%	89.51%	93.80%	94.84%	-
ILA++ (2020) [5]	80.73%	81.72%	91.46%	95.66%	90.61%	87.87%	88.74%	87.12%	93.63%	92.10%	88.96%
LinBP (2020) [3]	84.18%	90.46%	97.60%	98.74%	90.91%	92.53%	92.40%	93.10%	96.26%	97.94%	93.41%
ConBP (2021) [16]	82.06%	89.37%	96.79%	-	-	-	-	-	-	-	-
SE (2021) [9]	-	-	-	-	-	93.50%	90.74%	92.69%	-	95.70%	-
FIA (2021) [13]	**74.03%**	**76.36%**	89.87%	95.01%	85.26%	82.46%	85.44%	86.59%	92.26%	84.99%	85.23%
PNA (2022) [14]	-	-	-	-	-	90.04%	89.41%	89.39%	94.86%	-	-
NAA (2022) [17]	78.82%	85.62%	**87.47%**	**94.54%**	**71.63%**	**74.86%**	**76.91%**	**74.44%**	**85.79%**	**83.81%**	**81.39%**
- Substitute Model Training											
RFA (2021) [10]	67.07%	-	-	-	-	-	-	-	-	-	-
LGV (2022) [2]	74.50%	-	-	-	-	-	-	-	-	-	-
DRA (2022) [19]	**64.08%**	-	-	-	-	-	-	-	-	-	-
MoreBayesian (2023) [6]	70.27%	-	-	-	-	-	-	-	-	-	-
New Optimization Back-end											
- Baseline											
UN-DP-DI^2-TI-PI-FGSM	43.01%	**55.46%**	72.46%	74.63%	45.17%	44.74%	51.34%	44.53%	64.21%	60.51%	55.61%
- Gradient Computation											
TAP (2018) [18]	77.46%	65.52%	81.72%	92.11%	52.28%	70.49%	77.01%	54.50%	82.71%	74.16%	72.80%
NRDM (2018) [8]	71.29%	78.71%	86.06%	82.66%	65.64%	82.00%	85.22%	67.49%	93.12%	80.89%	79.31%
FDA (2019) [1]	58.47%	65.81%	77.84%	96.22%	79.57%	97.96%	95.63%	83.42%	95.38%	96.48%	84.68%
ILA (2019) [4]	47.83%	57.26%	72.79%	**73.97%**	49.47%	49.48%	64.42%	41.71%	75.91%	65.47%	59.83%
SGM (2020) [15]	**38.66%**	-	-	74.44%	32.59%	**39.81%**	36.00%	**34.64%**	**33.82%**	55.08%	-
ILA++ (2020) [5]	47.60%	55.86%	72.30%	74.29%	49.28%	49.54%	65.07%	41.73%	84.91%	65.50%	60.61%
LinBP (2020) [3]	48.76%	56.29%	89.04%	97.71%	**31.03%**	54.87%	50.77%	55.33%	81.93%	88.73%	65.45%
ConBP (2021) [16]	46.70%	56.23%	82.96%	-	-	-	-	-	-	-	-
SE (2021) [9]	-	-	-	-	-	54.36%	32.67%	38.32%	-	**53.08%**	-
FIA (2021) [13]	44.81%	59.26%	**71.82%**	88.47%	60.23%	52.48%	55.20%	64.83%	75.44%	69.44%	64.20%
PNA (2022) [14]	-	-	-	-	-	43.22%	**29.81%**	38.91%	51.68%	-	-
NAA (2022) [17]	47.03%	60.04%	72.02%	75.26%	41.44%	42.30%	46.82%	47.64%	65.23%	55.23%	**55.30%**
- Substitute Model Training											
RFA (2021) [10]	57.58%	-	-	-	-	-	-	-	-	-	-
LGV (2022) [2]	41.31%	-	-	-	-	-	-	-	-	-	-
DRA (2022) [19]	64.18%	-	-	-	-	-	-	-	-	-	-
MoreBayesian (2023) [6]	**39.01%**	-	-	-	-	-	-	-	-	-	-

Some ℓ_2 results are provided in this section. When I-FGSM is applied as the optimization back-end, same as the ℓ_∞ results in Table 1 in our main paper, NAA achieves the lowest AAA (*i.e.*, 81.39%)

compared with the other "gradient computation" methods, while FIA beats it when ResNet-50 or VGG-19 is chosen as the substitute model. See Table 3. However, unlike in the ℓ_∞ setting, SE shows consistently inferior performance when compared with the I-FGSM baseline in the ℓ_2 setting, and DRA instead of RFA achieves the best performance among "substitute model training" methods.

When UN-DP-DI^2-TI-PI-FGSM is applied as the new optimization back-end, same as in the ℓ_∞ setting, SGM, PNA, and SE provide favorable attack performance, while PNA on the DeiT-B substitute model turns out to be the best (in the sense of achieving lower BAA) and the generated adversarial examples fools victim models to show an accuracy of only 29.81%. The lowest WAA (which is 43.22%) is obtained by PNA. For the "substitute model training" methods, the MoreBayesian method still outperforms the other methods by a large margin.

2 Transfer between Convolution Networks and Vision Transformers

Table 4: The accuracy of victim models in predicting adversarial examples crafted via SGM using ResNet-50 and ViT-B as the substitute model, respectively. Smaller values indicate more powerful attacks. The optimization back-end is UN-DP-DI^2-TI-PI-FGSM, and the adversarial examples were generated under an ℓ_∞ constraint with $\epsilon = 8/255$.

Substitute model	ResNet -50	VGG -19	Inception v3	EffNetV2 -M	ConvNeXt -B	ViT -B	DeiT -B	BEiT -B	Swin -B	Mixer -B	AA
ResNet-50	-	2.72%	7.92%	29.42%	28.52%	48.32%	47.64%	36.82%	47.66%	38.70%	31.97%
ViT-B	30.00%	28.32%	36.40%	37.24%	33.66%	-	28.76%	15.60%	23.26%	25.92%	28.80%

To compare the transfer performance from vision transformers to convolutional networks and from the opposite direction, we report the accuracy of victim models in predicting SGM adversarial examples generated on ResNet-50/ViT-B as the substitute model. The results are shown in Table 4. It can be seen that transferring from vision transformers to convolutional networks is easier. When utilizing ViT-B as the substitute model, the accuracy of convolutional networks shows a range in $[28.32\%, 37.24\%]$, while, with ResNet-50, the accuracy of vision transformers lies in $[36.82\%, 48.32\%]$. Overall, using ViT-B as the substitute model leads to lower average accuracy (28.80% vs 31.97%) and the worst accuracy (37.24% vs 48.32%) on victim models, which means better average and worst-case attack performance, respectively.

3 Detailed Results of Augmentations and Optimizers

Table 5: Detailed results of different combinations of augmentations and optimizers. Smaller values indicate more powerful attacks. The adversarial examples were generated under an ℓ_∞ constraint with $\epsilon = 8/255$.

	ResNet -50	VGG -19	Inception v3	EffNetV2 -M	ConvNeXt -B	ViT -B	DeiT -B	BEiT -B	Swin -B	Mixer -B	AAA
PGD	88.36%	91.63%	93.72%	95.74%	88.50%	90.83%	90.71%	89.89%	94.57%	94.46%	91.84%
I-FGSM	87.79%	91.21%	93.71%	95.46%	88.32%	90.28%	90.28%	89.56%	94.81%	94.37%	91.58%
UN-PGD	86.07%	88.03%	93.02%	94.12%	83.11%	89.74%	89.19%	88.56%	92.37%	94.12%	89.83%
UN-I-FGSM	85.01%	86.88%	93.03%	94.04%	82.78%	89.12%	89.20%	87.76%	91.78%	93.62%	89.32%
SI-PGD	86.51%	86.22%	91.97%	89.31%	83.90%	88.96%	85.54%	87.67%	92.52%	92.96%	88.56%
SI-FGSM	86.21%	85.79%	91.74%	89.63%	83.87%	88.79%	84.78%	87.18%	91.87%	92.79%	88.26%
NI-FGSM	82.91%	87.23%	90.63%	92.09%	82.99%	87.14%	85.22%	86.10%	91.66%	91.97%	87.79%
PI-FGSM	82.46%	87.04%	90.24%	91.97%	82.79%	87.06%	85.36%	85.98%	91.32%	92.16%	87.64%
MI-FGSM	82.42%	86.94%	90.44%	91.91%	82.99%	87.14%	85.27%	85.86%	91.36%	92.04%	87.64%
MI-PGD	83.20%	87.59%	90.97%	91.47%	80.93%	87.07%	84.40%	85.62%	90.87%	91.71%	87.38%
......						......					
UN-DP-SI-DI^2-TI-PI-PGD	42.88%	50.34%	60.68%	44.19%	32.34%	37.28%	39.33%	35.56%	46.66%	44.47%	43.37%
UN-DP-SI-DI^2-TI-NI-FGSM	42.78%	50.40%	60.59%	44.10%	**32.33%**	36.93%	39.42%	35.83%	46.37%	**44.22%**	43.30%
UN-DP-SI-DI^2-TI-MI-FGSM	42.85%	50.34%	60.42%	44.03%	32.49%	36.73%	39.30%	35.91%	46.52%	44.31%	43.29%
UN-DP-SI-DI^2-TI-PI-FGSM	42.92%	50.12%	60.55%	**44.00%**	32.47%	36.74%	39.57%	35.94%	46.16%	44.30%	43.28%
UN-DP-DI^2-TI-PI-PGD	35.68%	49.07%	59.48%	52.40%	33.56%	33.53%	**35.58%**	34.85%	45.92%	46.30%	42.64%
UN-DP-DI^2-TI-MI-PGD	35.57%	48.70%	59.34%	52.34%	33.66%	33.69%	35.75%	34.84%	45.78%	46.45%	42.61%
UN-DP-DI^2-TI-NI-PGD	**35.34%**	48.55%	59.19%	52.20%	33.39%	33.39%	35.72%	34.83%	45.71%	46.42%	42.47%
UN-DP-DI^2-TI-MI-FGSM	35.80%	48.86%	59.15%	52.67%	33.22%	33.19%	35.90%	34.14%	45.28%	46.34%	42.46%
UN-DP-DI^2-TI-NI-FGSM	35.74%	48.77%	59.06%	52.70%	33.16%	33.26%	35.68%	34.24%	45.46%	46.40%	42.45%
UN-DP-DI^2-TI-PI-FGSM	35.70%	**48.33%**	**58.62%**	52.98%	33.64%	**32.74%**	36.58%	**33.72%**	**45.24%**	46.60%	**42.42%**

We show the detailed results of different combinations of augmentations and optimizers in Table 5. It can be seen that UN-DP-DI^2-TI-PI-FGSM achieves the best performance on average, despite the optimal solution on different substitute models are different.

4 Implementation Details

Augmentations and Optimizer. For PGD, DI^2-FGSM, MI-FGSM, NI-FGSM, and PI-FGSM, we use the default hyperparameters. For TI-FGSM, we randomly translate the input with a range of [-3, +3] since its performance is better than the approximation using a 7×7 Gaussian kernel in many implementations [7, 11, 12, 6]. For SI-FGSM and Admix, both of them average the gradients obtained by feeding different augmented inputs into the substitute model, which may lead to unfair comparisons. Therefore, we randomly select one input from the augmented copies, and the hyperparameters remain the same as in their original papers. For UN, the noise added to the input follows $\mathcal{U}(-\epsilon, \epsilon)$ and $\mathcal{U}(-\frac{\epsilon}{\sqrt{HW}}, \frac{\epsilon}{\sqrt{HW}})$ (the dimension of inputs is $3 \times H \times W$) for attacks under ℓ_∞ and ℓ_2 constraints, respectively. For DP, we divide the perturbation into 16×16 patches and randomly drop 50% of the patches at each iteration.

Gradient Computation. For TAIG, VT, IR, TAP, FDA, SE, and PNA, we set the same hyper-parameters as in their original papers. For NRDM, ILA, ILA++, LinBP, ConBP, FIA, and NAA, the main hyper-parameter which significantly impacts the performance is the choice of the middle layer. The scaling factor of SGM is also related to the selection of the substitute model. We tune these hyper-parameters by evaluating on a validation set consisting of 500 samples that do not overlap with the samples in the test set.

Substitute Model Training. In this category of methods, ResNet-50 is commonly chosen as the substitute model, and we collect the models from the GitHub repositories of these methods. For LGV and MoreBayesian, we only sample once at each iteration.

Generative Modeling. In this category of methods, all the generators are collected from the GitHub repositories of these methods.

References

[1] Aditya Ganeshan, Vivek BS, and R Venkatesh Babu. Fda: Feature disruptive attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8069–8079, 2019.

[2] Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. Lgv: Boosting adversarial example transferability from large geometric vicinity. *arXiv preprint arXiv:2207.13129*, 2022.

[3] Yiwen Guo, Qizhang Li, and Hao Chen. Backpropagating linearly improves transferability of adversarial examples. In *NeurIPS*, 2020.

[4] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *ICCV*, 2019.

[5] Qizhang Li, Yiwen Guo, and Hao Chen. Yet another intermediate-leve attack. In *ECCV*, 2020.

[6] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. Making substitute models more bayesian can enhance transferability of adversarial examples. In *International Conference on Learning Representations*, 2023.

[7] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*, 2019.

[8] Muzammal Naseer, Salman H Khan, Shafin Rahman, and Fatih Porikli. Task-generalizable adversarial attack based on perceptual metric. *arXiv preprint arXiv:1811.09020*, 2018.

[9] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli. On improving adversarial transferability of vision transformers. *arXiv preprint arXiv:2106.04169*, 2021.

[10] Jacob Springer, Melanie Mitchell, and Garrett Kenyon. A little robustness goes a long way: Leveraging robust features for targeted transfer attacks. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. Admix: Enhancing the transferability of adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16158–16167, 2021.

[12] Xiaosen Wang, Jiadong Lin, Han Hu, Jingdong Wang, and Kun He. Boosting adversarial transferability through enhanced momentum. *arXiv preprint arXiv:2103.10609*, 2021.

[13] Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature importance-aware transferable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7639–7648, 2021.

[14] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. Towards transferable adversarial attacks on vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[15] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Rethinking the security of skip connections in resnet-like neural networks. In *ICLR*, 2020.

[16] Chaoning Zhang, Philipp Benz, Gyusang Cho, Adil Karjauv, Soomin Ham, Chan-Hyun Youn, and In So Kweon. Backpropagating smoothly improves transferability of adversarial examples. In *CVPR 2021 Workshop Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV)*, volume 2, 2021.

[17] Jianping Zhang, Weibin Wu, Jen-tse Huang, Yizhan Huang, Wenxuan Wang, Yuxin Su, and Michael R Lyu. Improving adversarial transferability via neuron attribution-based attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14993–15002, 2022.

[18] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *ECCV*, 2018.

[19] Yao Zhu, Yuefeng Chen, Xiaodan Li, Kejiang Chen, Yuan He, Xiang Tian, Bolun Zheng, Yaowu Chen, and Qingming Huang. Toward understanding and boosting adversarial transferability from a distribution perspective. *IEEE Transactions on Image Processing*, 31:6487–6501, 2022.