Supplementary Material for "The Enemy of My Enemy is My Friend: Exploring Inverse Adversaries for Improving Adversarial Training"

Junhao Dong[1], Seyed-Mohsen Moosavi-Dezfooli[2], Jianhuang Lai[1,3,4] and Xiaohua Xie[1,3,4]
[1]School of Computer Science and Engineering, Sun Yat-Sen University, China
[2]Imperial College London, UK
[3]Guangdong Province Key Laboratory of Information Security Technology, China
[4]Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China
dongjh8@mail2.sysu.edu.cn, seyed.moosavi@imperial.ac.uk,
{stsljh, xiexiaoh6}@mail.sysu.edu.cn

In this supplementary material, we first present our experimental setup (see Section 1). Moreover, we give more details about our inverse adversarial training (see Section 2), including instance-wise inverse adversarial training, one-off strategy, and how our method can be combined with single-step adversarial training methods. We also provide visualization results (see Section 3) and the analysis of hyper-parameters (see Section 4).

1. Experimental Settings

In this section, we provide detailed experimental settings of used databases and our method.

1.1. Datasets

We conduct all our experiments on CIFAR-10/100 [8] and SVHN [11]. The CIFAR-10 dataset contains 60,000 color images with the size of 32×32 in 10 classes. The CIFAR-100 dataset shares the same setting as CIFAR-10, except it owns 100 classes consisting of 600 images each. In CIFAR-10/CIFAR-100 dataset, 50,000 images are for training, and 10,000 images are for testing the performance. SVHN is a dataset of street view house numbers, which includes 73,257 examples for training and 26,032 examples for evaluation. For training with additional data, we also include 1M synthetic images generated by the Denoising Diffusion Probabilistic Model (DDPM) [6] for CIFAR-10/100 following the setting of [12, 13].

1.2. Implementation Details

Following the hyper-parameter settings from [2, 12], we use Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum factor of 0.9 [10] and cyclic learning rate schedule [14] with the batch size of 128, the maximum learning rate of 0.1, and a weight decay factor of 5×10^{-4}. For training without extra data, our model is trained for 100 epochs for CIFAR-10/100 and 30 epochs for SVHN.

For training with synthetic DDPM-generated data [13] on CIFAR-10/100, we train models for 400 CIFAR-10-equivalent epochs (the same amount of training examples as standard CIFAR-10 in an epoch) with the batch size of 512. The original-to-generated ratio (*e.g.*, a ratio of 0.3 means that we include 7 synthetic images for every 3 original images) is 0.3 for CIFAR-10 and 0.4 for CIFAR-100. We also adopt the cyclic learning rate strategy with a maximum learning rate of 0.2. Following the training setup from [12, 13], we use SiLU activation function [5] with Pre-activation ResNet-18 (PRN-18) [4] and Wide-ResNet-28-10 (WRN-28-10) [17]. We further use model weight averaging [7] with a decay factor of 0.995.

For generating adversarial examples during training, we adopt the iterative Projected Gradient Descent (PGD) algorithm [9] on the cross-entropy loss function for 10 steps with the step size $\alpha = 2/255$ for CIAFR-10/100 and $\alpha = 1/255$ for SVHN. We mainly consider the ℓ_∞-norm threat model with the maximum adversarial perturbation $\epsilon = 8/255$. We set the inverse perturbation radius as $\epsilon' = 4/255$. The number of iterations for instance-wise inverse perturbation is 5 with the iteration step size of $\alpha' = 2/255$, whilst we conduct single-step gradient descent on universal inverse perturbation with the step size of $\alpha' = 4/255$. We choose the trade-off factor $\lambda = 3.5$ for CIFAR10/100 and $\lambda = 3.0$ for SVHN. The regularization hyper-parameter β is set to 1.0. We pick the inverse momentum factor $\gamma = 0.9$ for our standard inverse adversarial training method except for the one-off setting. We do not involve the inverse momentum when adopting the one-off strategy. The momentum mechanism starts at epoch $T = 75$ when training for 100 epochs and starts at epoch $T = 350$ when training for 400 epochs. The one-off epoch choice is $T' = 80$ for 100

Algorithm 1 **I**nverse **A**dversarial **T**raining (**IAT**)

Input: DNN classifier $f_{\boldsymbol{\theta}}$; dataset $\mathcal{D} = \{(\mathbf{x}, y)\}$ with C classes; batch size m; learning rate τ; radius for adversaries ϵ and inverse adversaries ϵ'; iteration times n and step size α' for inverse adversary generation; weighting factors λ, β.

Randomly initialize the network parameter $\boldsymbol{\theta}$
while not at end of training **do**
  for each mini-batch $(\mathbf{x}, y) = \{(\mathbf{x}_j, y_j)\}_{j=1}^{m}$ **do**
    for $j = 1, 2, \ldots, m$ **do**
      Initialize Inverse adversarial perturbation $\mathbf{z}_j \sim 0.001 \cdot \mathcal{N}(0, 1)$
      $\hat{\mathbf{x}}_j \leftarrow \text{PGDATTACK}(\mathbf{x}_j, y_j, f_{\boldsymbol{\theta}}, \mathcal{L}_{\text{CE}})$ ▷ Find PGD adversarial example
      $\check{\mathbf{x}}_j \leftarrow \mathbf{x}_j + \mathbf{z}_j$
      for $t = 1, 2, \ldots, n$ **do**
        $\check{\mathbf{x}}_j = \Pi_{\mathbb{B}(\mathbf{x}, \epsilon')} \left(\check{\mathbf{x}}_j - \alpha' \cdot \text{sign} \left(\nabla_{\check{\mathbf{x}}_j} \mathcal{L}_{\text{Inv}} \left(\check{\mathbf{x}}_j, y \right) \right) \right)$ ▷ Update instance-wise inverse adversaries
      end for
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \cdot \nabla_{\boldsymbol{\theta}} \left\{ \sum_j \mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}}_j \right), y_j \right) + \lambda \cdot \mathcal{L}_{\text{KL}} \left(f_{\boldsymbol{\theta}} \left(\check{\mathbf{x}}_j \right) \| f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}}_j \right) \right) \right\}$
  end for
end while
return Inverse adversarially trained model $f_{\boldsymbol{\theta}}$.

training epochs and $T' = 320$ for 400 training epochs.

2. Details of Inverse Adversarial Training

2.1. Instance-wise Inverse Adversarial Training

We have introduced how to generate instance-wise inverse adversaries in the main paper. In this section, we give more details about combining inverse adversarial examples with adversarial training. In general, we generate inverse adversarial perturbation for each natural example via the PGD method [9] optimized on the inverse adversarial loss. The instance-wise Inverse Adversarial Training (**IAT**) is quite similar to Universal Inverse Adversarial Training (**UIAT**) we have introduced in detail. We can easily obtain IAT by replacing universal inverse adversaries with instance-wise inverse adversaries. We provide the pseudo-code of IAT in Algorithm 1.

2.2. One-off Strategy

In this section, we provide more details about the one-off strategy and how it can be combined with our method. The one-off strategy means generating inverse adversarial examples for only one certain epoch T' instead of throughout the whole training stage. During the standard inverse adversarial training, we mainly optimize cross-entropy loss of adversarial examples and Kullback–Leibler (KL) divergence between inverse adversaries and adversarial examples. However, the one-off strategy mainly focuses on the substitution of the inverse adversaries throughout the adversarial training, which can reduce the computational overhead effectively. The loss function for the One-Off version of inverse adversarial training can be formulated as below:

$$\mathcal{L}_{\text{IAT}}^{\text{OO}} = \mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}} \right), y \right) + \lambda \cdot \mathcal{L}_{\text{KL}} \left(\mathbf{p}_{\text{OO}}^{(t)} \| f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}} \right) \right), \quad (1)$$

where $\hat{\mathbf{x}}$ is the adversarial example. $\mathbf{p}_{OO}^{(t)}$ denotes the one-off output probability that mainly depends on the current training epoch t, which can be obtained by:

$$\mathbf{p}_{\text{OO}}^{(t)} = \begin{cases} f_{\boldsymbol{\theta}} \left(\mathbf{x} \right), & \text{if } t < T' \\ f_{\boldsymbol{\theta}} \left(\check{\mathbf{x}} \right), & \text{if } t = T' \\ \mathbf{p}_{\text{OO}}^{(T')}, & \text{if } t > T' \end{cases} \quad (2)$$

where $\check{\mathbf{x}}$ denotes the inverse adversarial example, and T' is the only epoch for generating inverse adversarial examples. Before epoch T', we replace inverse adversaries with natural examples during adversarial training, which is similar to [15]. Particularly, we generate inverse adversarial examples and use them for distribution alignment during epoch T'. After epoch T', we use the output probability of inverse adversaries at epoch T' instead of recomputing inverse adversarial examples. The motivation is that the feature representation tends to be stable at a later stage of training. Thus we can consistently obtain the high-likelihood region with the same inverse adversarial examples. Therefore, it is reasonable to continue to use the previously computed inverse adversaries to represent the high-likelihood region during the current training epoch.

2.3. Single-step Adversarial Training

In this section, we give more details about how our method can be combined with single-step adversarial training methods [1, 3, 16]. When using ℓ_∞-norm threat model, we can formalize the adversarial training [9] as the following min-max optimization problem:

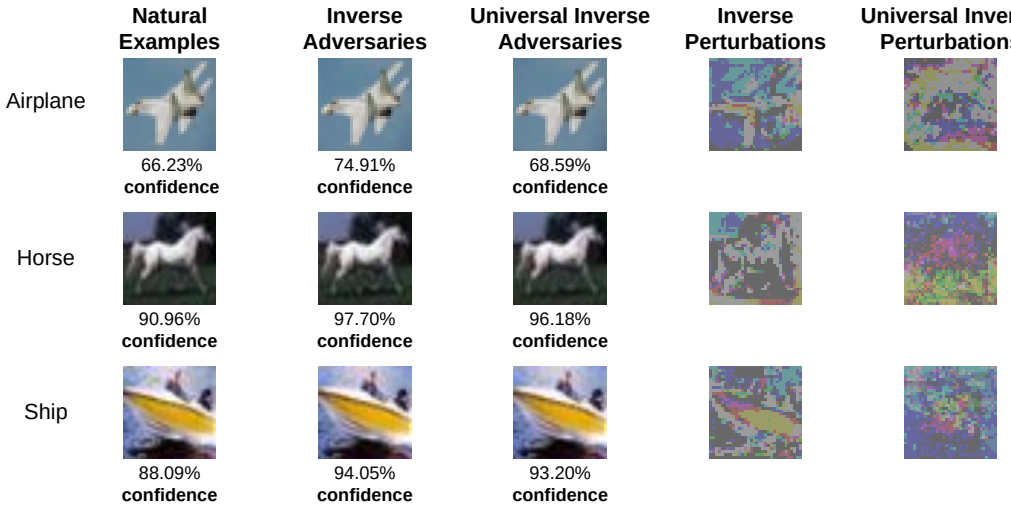

Figure 1. Visualization of both inverse adversaries and class-specific universal inverse adversaries. Their corresponding inverse adversarial perturbations are also presented. In addition, we present the prediction confidence of the ground-truth class.

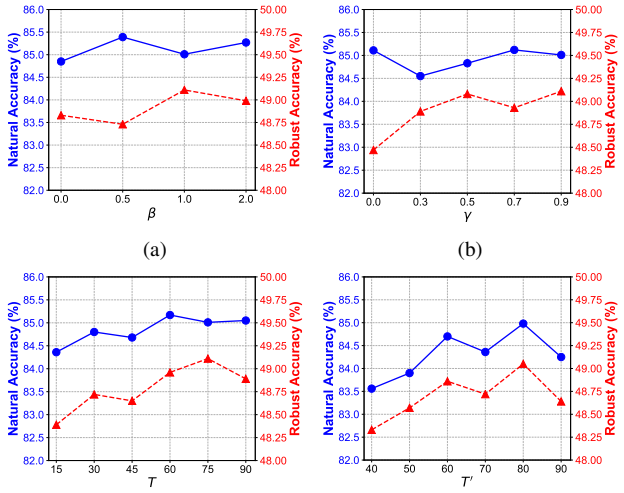

Figure 2. Hyper-parameter sensitivity of our UIAT method on natural accuracy and (Auto-Attack) robust accuracy using ResNet-18 on CIFAR-10. We report the hyper-parameters adjustment of β in (a) and γ in (b). The tuning for the starting epoch of momentum T is in (c), and the one-off epoch T' is in (d)

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|_\infty<\epsilon} \mathcal{L}_{\mathrm{CE}} \left(f_{\boldsymbol{\theta}} \left(\mathbf{x} + \boldsymbol{\delta}^{\mathrm{SGL}} \right), y \right) \right], \quad (3)$$

where $\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss, and δ is the adversarial perturbation under the ℓ_∞-norm bound ϵ. The inner maximization problem of adversarial training can be viewed as searching for the most harmful adversarial examples $\hat{\mathbf{x}}^{\mathrm{SGL}} = \mathbf{x} + \boldsymbol{\delta}^{\mathrm{SGL}}$. However, the iterative solution for the inner maximization problem, *i.e.*, the PGD method, suffers from a high computational cost. Notably, most single-step adversarial training methods approximate the worst-case perturbation by solving the inner maximization in Equation (3) with the following form:

$$\boldsymbol{\delta}^{\mathrm{SGL}} = \psi\Big(\boldsymbol{\eta} + \alpha \cdot \mathrm{sign} \left(\nabla_{\mathbf{x}} \mathcal{L}_{\mathrm{CE}}(f_\theta(\mathbf{x}+\boldsymbol{\eta}), y) \right) \Big), \quad (4)$$

where ψ is a projection operator onto the ℓ_∞-norm ball and $\boldsymbol{\eta}$ is drawn from a certain distribution $\boldsymbol{\Omega}$ that can be typically a uniform distribution between $[-\epsilon, \epsilon]$. When combining our UIAT method with these single-step adversarial training methods, we do not modify the inner maximization to obtain adversarial perturbations $\boldsymbol{\delta}^{\mathrm{SGL}}$. We primarily focus on outer minimization, where we add an additional KL divergence term between universal inverse adversaries $\check{\mathbf{x}}$ and adversarial examples $\hat{\mathbf{x}}^{\mathrm{SGL}}$. Hence, a general form of the loss function for single-step adversarial training (outer minimization) can be defined as below:

$$\mathcal{L}_{\mathrm{IAT}}^{\mathrm{SGL}} = \mathcal{L}_{\mathrm{CE}} \left(f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}} \right), y \right) + \lambda \cdot \mathcal{L}_{\mathrm{KL}} \left(f_{\boldsymbol{\theta}} \left(\check{\mathbf{x}} \right) \| f_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}}^{\mathrm{SGL}} \right) \right), \quad (5)$$

where $\check{\mathbf{x}}$ denotes the universal inverse adversarial example that is also obtained by single-step gradient descent on the inverse adversarial loss. Note that we do not apply the feature-level regularization during inverse adversary generation for efficiency, which means we only use the cross-entropy loss for inverse adversary generation. In general, we can efficiently combine our method with single-step adversarial training by paying only three additional forward propagation times and one backward propagation time for each batch of data.

3. Visualization

We visualize both the (universal) inverse adversarial examples and their inverse perturbations in Figure 1. It can be seen that class-specific universal inverse adversaries can

obtain a similar inverse effect (improving predicting confidence on the correct category) to the original inverse adversaries. Note that the standard inverse adversarial perturbation is specific to a certain example, while the universal inverse perturbation can apply to examples from a given category. These inverse examples are also visually indistinguishable from natural examples.

4. Hyper-parameter Analysis

To comprehensively analyze the contribution of each component, we report natural accuracy and robust accuracy when tuning component weights, as shown in Figure 2. It can be seen that enlarging the momentum factor γ can further improve the adversarially robust accuracy. In addition, choosing the start epoch T for enabling inverse adversarial momentum during the second half of training can benefit both natural accuracy and adversarial robustness. In particular, we can observe that the choice for the one-off epoch T' is essential, and there exists a huge performance variance when tuning this hyper-parameter. Similar to the choice for the start epoch for momentum, it is beneficial to adopt the one-off output probability of inverse adversaries during the second half of training.

References

[1] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020. 2

[2] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *NIPS ML Systems Workshop*, 2017. 1

[3] Pau de Jorge, Adel Bibi, Riccardo Volpi, Amartya Sanyal, Philip HS Torr, Grégory Rogez, and Puneet K Dokania. Make some noise: Reliable and efficient single-step adversarial training. *arXiv preprint arXiv:2202.01181*, 2022. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 1

[5] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 1

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1

[7] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI*, pages 876–885. AUAI Press, 2018. 1

[8] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018*, 2018. 1, 2

[10] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983. 1

[11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 1

[12] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 1

[13] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021. 1

[14] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, page 1100612. SPIE, 2019. 1

[15] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019. 2

[16] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR*, 2020. 2

[17] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC*, 2016. 1