arXiv:2211.00525v1 [cs.CV] 1 Nov 2022

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

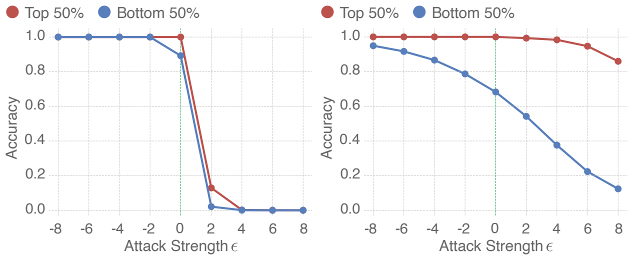
Junhao Dong[1], Seyed-Mohsen Moosavi-Dezfooli[2], Jianhuang Lai[1] and Xiaohua Xie[1]
[1]Sun Yat-Sen University, China, [2]Imperial College London, UK
dongjh8@mail2.sysu.edu.cn, seyed.moosavi@imperial.ac.uk,
{stsljh, xiexiaoh6}@mail.sysu.edu.cn

Abstract

Although current deep learning techniques have yielded superior performance on various computer vision tasks, yet they are still vulnerable to adversarial examples. Adversarial training and its variants have been shown to be the most effective approaches to defend against adversarial examples. These methods usually regularize the difference between output probabilities for an adversarial and its corresponding natural example. However, it may have a negative impact if the model misclassifies a natural example. To circumvent this issue, we propose a novel adversarial training scheme that encourages the model to produce similar outputs for an adversarial example and its "inverse adversarial" counterpart. These samples are generated to maximize the likelihood in the neighborhood of natural examples. Extensive experiments on various vision datasets and architectures demonstrate that our training method achieves state-of-the-art robustness as well as natural accuracy. Furthermore, using a universal version of inverse adversarial examples, we improve the performance of single-step adversarial training techniques at a low computational cost.

1. Introduction

Deep learning has achieved revolutionary progress in numerous computer vision tasks [28, 44, 59] and has emerged as a promising technique for fundamental research in multiple disciplines [35, 39, 56]. However, a well-established study has demonstrated that Deep Neural Networks (DNNs) are extremely vulnerable to adversarial examples [12, 46], which are indistinguishable from natural examples in human vision. In other words, a visually undetectable perturbation to the original example can lead to a significant disruption of the inference result of DNNs. The stealthiness of these tailored examples also makes them easy to bypass manual verification [3, 18], posing a potential security threat to the safety of deep learning-based applications. Consequently, adversarial robustness has been considered a new measurement for deep learning security.



(a) Natural Training (b) Adversarial Training [29]

Figure 1. Average accuracy under different attack strengths for two networks trained on natural and adversarial samples. We rank test examples based on the natural classification accuracy in decreasing order and divide them into two equal halves. Note that the negative ϵ denotes the strength of inverse adversarial perturbation. (a) Naturally trained models are extremely susceptible to perturbations. (b) For adversarially trained models, the adversarial effect is exacerbated on examples that are more possibly to be misclassified. The green line corresponds to natural examples.

Various defense methods have been proposed to improve the adversarial robustness of DNNs [25, 50, 52]. As the primary defense method, adversarial training [12, 29, 46] improves intrinsic network robustness via adaptively augmenting adversarial examples into training examples. Existing adversarial training methods mainly focus on the distribution alignment between legitimate examples and adversarial examples to preserve the consistency of the DNN prediction results [8, 48, 57]. However, there still exists a considerable feature representation gap between natural examples and their adversarial counterparts, resulting in the undesirable decision boundary for the misclassification of natural examples. The misclassification can further be exacerbated when confronted with adversarial examples.

The natural intuition is that: most adversarial examples corresponding to misclassified natural examples are more possibly to be misclassified. Opposite to adversaries that are harmful to DNNs, we introduce inverse adversarial examples[1] that are created via minimizing the objective function as an inverse procedure of adversary generation. Specifically, inverse adversarial examples are affiliative to DNNs, which can be more possibly to be correctly classified. To support this claim, we study the accuracy of trained classification models on two groups of samples (see Figure 1). We present the accuracy of adversarial examples and their inverse counterparts under different attack strengths. For the adversarially trained model, the robust accuracy of examples with high loss suffers from a heavier drop than that of examples with low loss under larger attack strengths. This means that the adversarial counterparts of low-confidence or even misclassified examples are also misclassified. Therefore, the distribution alignment between two misclassified examples might have an unnecessary or even harmful effect on the robustness establishment.

In this paper, beyond the unnecessary or even harmful matching manner between misclassified examples, we propose a novel adversarial training framework based on the inverse version of adversarial examples, dubbed *Inverse Adversarial Training* (IAT), which implicitly bridges the gap between adversarial examples and the high-likelihood region of their belonging classes. Adversarial examples of a certain category can thus be pulled closer to the high-likelihood region instead of their original examples. Specifically, we involve an inverse procedure of the standard adversary generation to obtain the high-likelihood region. In general, inverse adversarial examples can be viewed as the regularization of original examples for reducing prediction errors. Considering the multi-class decision surface and computational cost, we design a class-specific inverse adversary generation paradigm as opposed to the instance-wise pattern. Furthermore, we establish a momentum mechanism for the prediction of inverse adversaries to stabilize the training process. A one-off version of the inverse adversary generation is also proposed for improving time efficiency.

Comprehensive experiments demonstrate the effectiveness and generalizability of our method that can efficiently obtain a better trade-off between natural accuracy and robustness. We also show that our method can also be adapted to larger models with extra generated data for robustness enhancement. Besides, the robustness of single-step adversarial training methods can be further improved at a low cost by incorporating our method.

The main contribution of this paper can be summarized as follows:

- By analyzing the unnecessary, or even harmful, alignment between misclassified examples, we propose a novel adversarial training framework based on the inverse version of adversarial examples, which promotes the aggregation of adversarial examples to the high-likelihood region of their belonging classes.
- Based on the proposed *Inverse Adversarial Training* (IAT) paradigm, we further design a class-specific universal inverse adversary generation strategy to mitigate the class-wise imbalance hiding in the decision surface. We also propose a one-off inverse adversary generation strategy to reduce computational costs with a negligible performance loss.
- Extensive experiments demonstrate the effectiveness of IAT compared with state-of-the-art methods when using large models with extra synthetic data. Furthermore, we achieve a better trade-off between natural accuracy and adversarial robustness efficiently. Our method can also be combined with single-step adversarial training methods as a plug-and-play component for boosting robustness at a low cost.

Related Works. The lethal vulnerabilities of deep neural networks against adversarial examples have been witnessed in [4, 12, 32, 46]. A myriad of attempts have been made to defend against these tailored examples, including adversarial training [21,29,48,57], adversarial detection [17,47], and input transformation-based methods [41, 52, 53]. Among them, adversarial training consistently remains to be the most effective method [2] to improve intrinsic network robustness via augmenting the training data with adversarial examples. In addition, most existing works generally incorporate a regularization term to narrow the distribution difference between natural examples and their adversarial counterparts [8, 48, 57], which has been demonstrated to be beneficial for robustness enhancement. This matching manner seems natural but might be misguided by misclassified natural examples, as we showed in Figure 1. To circumvent this issue, several efforts have been devoted to assigning weights on losses in terms of the intensity of adversarial examples [10, 27, 58]. However, they mainly concentrate on mitigating the imbalance of disturbance effect among adversarial examples, while our primary focus is to prevent the harmful alignment between misclassified examples by incorporating inverse adversarial examples.

Inverse adversarial examples were first formally described in [40], where Salman *et al.* studied them in vision systems to enhance in-distribution performance against new corruptions. In comparison, we investigate the regularization effect of inverse adversarial examples on the distribution alignment during adversarial training for robustness enhancement. A concurrent work [26] also exploits the inverse version of adversarial examples for adversarial robustness incorporating different distance metrics. However, we built on class-specific universal inverse adversaries for adversarial training with more efficiency and robustness. We

[1] The formal definition will be given in the following sections.

also involve the feature-level prior knowledge in the inverse adversary generation for supplementary regularization. Furthermore, we show how our method can be combined with single-step adversarial training techniques to improve both the natural performance and robustness.

2. Background

Consider a DNN classifier $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathbb{R}^C$ with parameters $\boldsymbol{\theta}$ that predicts probabilities of C classes. Specially, we symbolize the output feature representation of the penultimate layer (before logits) $\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{x})$ for a given example $\mathbf{x} \in \mathcal{X}$. Adversarial training can be an effective way to enhance the robustness of DNNs against adversarial perturbations, which adaptively involves adversarial examples in training as strong data augmentation. For a specific dataset $(\mathbf{x}, y) \sim \mathcal{D}$, the standard adversarial training [29] against attacks under ℓ_∞-norm threat model can be formulated as the following min-max optimization problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|_\infty < \epsilon} \mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta}} \left(\mathbf{x} + \boldsymbol{\delta} \right), y \right) \right], \quad (1)$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss and δ is the adversarial perturbation under the ℓ_∞-norm bound ϵ. The outer minimization is to optimize empirical adversarial risk over the network parameters $\boldsymbol{\theta}$. The inner maximization of adversarial training can be viewed as searching for the most harmful adversarial examples $\hat{\mathbf{x}} = \mathbf{x} + \delta$, which can be simplified as an iterative Projected Gradient Descent (PGD) algorithm [29] on the negative loss function.

Besides standard adversarial training, TRADES [57] and MART [48] proposed to utilize Kullback–Leibler (KL) divergence for distribution matching between natural examples and their adversarial counterparts. The objective function of TRADES [57] can be defined as follows:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \Big[& \mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta}} \left(\mathbf{x} \right), y \right) + \\ & \omega \cdot \max_{\|\delta\|_\infty < \epsilon} \mathcal{L}_{\text{KL}} \left(f_{\boldsymbol{\theta}} \left(\mathbf{x} \right) \| f_{\boldsymbol{\theta}} \left(\mathbf{x} + \delta \right) \right) \Big], \end{aligned} \quad (2)$$

where $\mathcal{L}_{\text{KL}}$ denotes KL divergence and ω is the balancing parameter for the trade-off of natural accuracy and adversarial robustness. Generally, KL divergence encourages the predictions of benign examples and adversarial examples to share the same distribution. Nevertheless, this distribution alignment can further undermine the adversarial robustness when benign examples are misclassified, resulting in the wrong guidance during adversarial training. A major drawback in adversarial training is that it costs more considerable computing resources than natural training [1, 42, 51], which hinders robust establishment on larger models. In addition, adversarial training can suffer from more severe overfitting than the natural training paradigm [38]. Later in this paper, we will provide some insights related to the above-mentioned challenges regarding adversarial training.

3. Method

In this section, we first formally define the inverse adversarial example and introduce its class-specific (universal) variant. We then propose a new adversarial training scheme, coined as *Universal Inverse Adversarial Training* (UIAT), via a regularizer that encourages the prediction alignment between adversarial examples and the high-likelihood region of their corresponding classes. Furthermore, the inverse adversary momentum is also proposed for the stabilization of the training process. For boosting time efficiency, we design a one-off version of UIAT by computing inverse adversaries only in one of the epochs without losing much performance.

3.1. Inverse Adversarial Examples

For the image classification task, the adversary generation can be viewed as a process of crossing the decision boundary for misclassification. On the contrary, generating inverse adversarial examples can be regarded as moving away from the decision boundary to the high-likelihood region of a certain class. Specifically, this process can be obtained by iteratively minimizing the classification loss values of inverse adversarial examples. Formally, inverse adversarial examples are inputs to machine learning models, which are tailored to cause the model to obtain more accurate predictions than corresponding natural examples. Similar to adversarial examples, inverse adversaries are obtained by adding visually tiny perturbations to natural ones. We here focus on ℓ_∞-norm bound $\mathbb{B}(\mathbf{x}, \epsilon')$ with radius ϵ' around natural examples on inverse adversaries. One can use PGD to generate inverse adversarial perturbations:

$$\check{\mathbf{x}}^{t+1} = \Pi_{\mathbb{B}(\mathbf{x},\epsilon')} \left(\check{\mathbf{x}}^t - \alpha' \cdot \text{sign} \left(\nabla_{\check{\mathbf{x}}^t} \mathcal{L}_{\text{Inv}} \left(\check{\mathbf{x}}^t, y \right) \right) \right), \quad (3)$$

where α' is the gradient descent step size, $\check{\mathbf{x}}^t$ represents t^{th} iteration update, and $\mathcal{L}_{\text{Inv}}$ denotes the loss function for the inverse adversary generation. Generally, the cross-entropy loss can be a good choice for guiding the inverse adversary generation. Nevertheless, the high-likelihood region of a certain class is far away from any adjacent decision boundaries [22, 49], which means that inverse adversaries are far away from adversarial examples at the feature level. Meanwhile, natural feature embeddings are also desired to lie on the high-likelihood region from the geometric perspective. We thus append a feature-level regularization during the inverse adversary generation for supplementary supervision. Therefore, given a sample $\mathbf{x}$ and its adversarial counterpart $\hat{\mathbf{x}}$, our inverse adversarial loss can be written as follows:

$$\begin{aligned} \mathcal{L}_{\text{Inv}} \left(\check{\mathbf{x}}, y \right) = & \mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta}} \left(\check{\mathbf{x}} \right), y \right) \\ + \beta \cdot [& \mathcal{L}_1 \left(\mathcal{F}_{\boldsymbol{\theta}} \left(\check{\mathbf{x}} \right), \mathcal{F}_{\boldsymbol{\theta}} \left(\mathbf{x} \right) \right) - \mathcal{L}_1 \left(\mathcal{F}_{\boldsymbol{\theta}} \left(\check{\mathbf{x}} \right), \mathcal{F}_{\boldsymbol{\theta}} \left(\hat{\mathbf{x}} \right) \right)], \end{aligned} \quad (4)$$

where β denotes the weighting factor. Similar to insights from adversarial feature space analysis [30], our triplet term on latent representations can further prevent the overfitting of inverse adversaries from extremely high predictions for better guidance. The obtained inverse adversarial examples can then be incorporated into the adversarial training for robustness enhancement.

3.2. Class-specific Inverse Adversaries

We have introduced the instance-wise inverse adversarial example in the previous section, which is effective in approximating the high-confidence region in the decision surface. However, the inverse adversary generation suffers from a high computational cost due to iterative gradient computation. In general, the instance-wise inverse adversary generation can take almost the same time as the original adversary generation. To reduce the computational cost of inverse adversary generation, we further design a *Class-Specific Universal* inverse adversary generation strategy inspired by [31, 43]. The universal strategy allows examples of the same class to share a universal adversarial perturbation. In other words, each class owns a universal inverse adversarial perturbation that can be effective in approaching its high-likelihood region (lower the objective loss). In this way, we can find a shared direction to reach high-likelihood regions, which can also mitigate the individual noise between different examples. The class-specific universal adversarial perturbation $\mathbf{z}_c$ for class c can be defined as:

$$\begin{gathered}\mathcal{L}_{\text{Inv}}\left(\Pi_{\mathbb{B}(\mathbf{x}^c,\epsilon')}\left(\mathbf{x}^c+\mathbf{z}_c\right),y^c\right)<\mathcal{L}_{\text{Inv}}\left(\mathbf{x}^c,y^c\right)\\ \text{for “most” } \mathbf{x}^c\sim\mathcal{D}^c.\end{gathered}\tag{5}$$

We sample natural examples $\mathbf{x}^c$ and corresponding labels y^c from dataset $\mathcal{D}^c$ of category c. The class-specific universal inverse perturbation $\mathbf{z}_c$ is effective in most of the examples from the same class c for reducing the loss. Note that we keep updating class-specific inverse perturbations throughout the whole training stage. For a certain batch of data, we can obtain the updated universal inverse perturbation by solving the following optimization problem:

$$\min_{\|\mathbf{z}_c\|_\infty<\epsilon}\frac{1}{N_c}\sum_{i=1}^{N_c}\mathcal{L}_{\text{Inv}}\left(\mathbf{x}_i^c+\mathbf{z}_c,y_i^c\right),\tag{6}$$

where N_c is the number of training samples belonging to class c of a batch. Specifically, we can further use PGD to solve the above optimization problem to obtain class-specific inverse adversarial perturbation $\mathbf{z}_c$. For time efficiency, we only conduct a single-step PGD to update the universal inverse perturbation for a certain category.

3.3. Universal Inverse Adversarial Training

We here show how the universal inverse adversaries can be used to devise an effective adversarial training algorithm. The universal inverse adversarial example $\check{\mathbf{x}}$ can be obtained by adding the class-specific inverse perturbation to the original example $\mathbf{x}$. The loss function of *Universal Inverse Adversarial Training* (UIAT) can be formulated as below:

$$\mathcal{L}_{\text{UIAT}}=\mathcal{L}_{\text{CE}}\left(f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right),y\right)+\lambda\cdot\mathcal{L}_{\text{KL}}\left(\mathbf{p}^{(t)}\|f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right)\right),\tag{7}$$

where t denotes the current training epoch number. To mitigate the oscillations of noisy predictions throughout the training process, we design a momentum mechanism on the predicted probability of inverse adversaries via incorporating predictions from previous epochs. The momentum mechanism to obtain aggregate predicted probability $\mathbf{p}^{(t)}$ can thus be described as:

$$\mathbf{p}^{(t)}=\begin{cases}f_{\boldsymbol{\theta}}\left(\check{\mathbf{x}}\right), & \text{if } t<T\\ \gamma\cdot\mathbf{p}^{(t-1)}+(1-\gamma)\cdot f_{\boldsymbol{\theta}}\left(\check{\mathbf{x}}\right), & \text{if } t\geq T\end{cases}\tag{8}$$

where γ is the momentum factor. Note that we start to enable the inverse adversary momentum at epoch T to stabilize the training process. The main reason is that the learned representation is unstable during the early training period. Our UIAT method can thus bridge the gap between adversarial examples and the high-likelihood region of their belonging classes for robustness enhancement. The pseudocode of our UIAT is provided in Algorithm 1. We can easily obtain standard IAT by replacing universal inverse adversaries with instance-wise ones. (See Appendix B.1)

To further reduce the computational overhead, we provide a one-off strategy, which only conducts the inverse adversary generation in a certain epoch T' instead of generating inverse adversaries throughout the whole training process. Before epoch T', we replace $\mathbf{p}^{(t)}$ in Equation (7) with $f_{\boldsymbol{\theta}}\left(\mathbf{x}\right)$ for adversarial training, which is similar to [48]. Afterward, we keep replacing $\mathbf{p}^{(t)}$ with the temporary probability $\mathbf{p}^{(T')}$ in following epochs $t>T'$. More details of our one-off strategy are given in Appendix B.2.

4. Experiments

In this section, we conduct extensive experiments to demonstrate the effectiveness and generalizability of our method. We first introduce our experimental settings, including datasets and implementation details. Next, we compare our method with state-of-the-art adversarial training methods in various settings, demonstrating the superiority of our inverse adversarial training. Moreover, we show that our method can be combined with single-step adversarial training methods, which meaningfully increases their performance at only a small additional cost.

4.1. Experimental Setups

Datasets. We conduct experiments on three standard datasets: CIFAR-10, CIFAR-100 [24], and SVHN [34]. Details of datasets are provided in Appendix A.1.

Algorithm 1 Universal Inverse Adversarial Training (UIAT)

```
Input: DNN classifier $f_{\boldsymbol{\theta}}$; dataset $\mathcal{D} = \{(\mathbf{x}, y)\}$ with $C$ classes; batch size m; learning rate $\tau$; radius for adversaries $\epsilon$
    and inverse adversaries $\epsilon'$; step size $\alpha'$ for inverse adversary generation; weighting factors $\lambda$; momentum factor $\gamma$.
Randomly initialize the network parameter $\boldsymbol{\theta}$. Initialize $\mathbf{z}_c \sim 0.001 \cdot \mathcal{N}(0,1)$, for $1 \le c \le C$
while not at end of training do
    for each mini-batch $\{(\mathbf{x}_j, y_j)\}_{j=1}^{m}$ do
        Initialize $l_{Inv}^{c} \leftarrow 0$, for $1 \le c \le C$
        for $j = 1, 2, \ldots, m$ do
            $\hat{\mathbf{x}}_j \leftarrow \mathrm{PGDATTACK}(\mathbf{x}_j, y_j, f_{\boldsymbol{\theta}})$            ▷ Find PGD adversarial example
            $\check{\mathbf{x}}_j \leftarrow \mathbf{x}_j + \mathbf{z}_{y_j}$
            $l_{Inv}^{y_j} \leftarrow l_{Inv}^{y_j} + \mathcal{L}_{\text{Inv}}(\check{\mathbf{x}}_j, y_j)$
        end for
        for $c = 1, \ldots, C$ do
            $\mathbf{z}_c \leftarrow \Pi_{\|\mathbf{z}_c\|_\infty \le \epsilon'} \left(\mathbf{z}_c - \alpha' \cdot \mathrm{sign}\left(\nabla_{\mathbf{z}_c} l_{Inv}^{c}\right)\right)$            ▷ Update class-specific inverse adversaries
        end for
        Obtain $\mathbf{p}_j^{(t)}$, for $1 \le j \le m$, by Eq. (8) according to current epoch number $t$            ▷ Inverse adversary momentum
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \cdot \nabla_{\boldsymbol{\theta}} \left\{ \sum_j \mathcal{L}_{\text{CE}}\left(f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}_j\right), y_j\right) + \lambda \cdot \mathcal{L}_{\text{KL}}\left(\mathbf{p}_j^{(t)} \| f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}_j\right)\right) \right\}$
    end for
end while
return Inverse adversarially trained model $f_{\boldsymbol{\theta}}$.
```

Table 1. Comparison of our methods (UIAT) using ResNet-18 trained on CIFAR-10, CIFAR-100, and SVHN with other adversarial training methods. The ℓ_∞-norm adversarial perturbations are restricted in $\epsilon = 8/255$. We report both natural accuracy (%) and robust accuracy (%). The best result in each column is in **bold**.

Method	CIFAR-10				CIFAR-100				SVHN			
	Natural	PGD	CW	AA	Natural	PGD	CW	AA	Natural	PGD	CW	AA
SAT [29]	83.80	51.40	50.17	47.68	57.39	28.36	26.29	23.18	92.46	50.55	50.40	46.07
TRADES [57]	82.45	52.21	50.29	48.88	54.36	27.49	24.19	23.14	90.63	58.10	55.13	**52.62**
MART [48]	82.20	53.94	50.43	48.04	54.78	28.79	26.15	24.58	89.88	**58.48**	52.48	48.44
HAT [36]	84.86	52.04	50.33	48.85	58.73	27.92	24.60	23.34	92.06	57.35	54.77	52.06
UIAT	**85.01**	54.63	51.10	**49.11**	59.55	**30.81**	**28.05**	**25.73**	**93.28**	58.18	**55.49**	52.45
UIAT (One-off)	84.98	**54.79**	**51.29**	49.05	**60.01**	30.49	27.56	25.45	93.14	58.30	55.45	52.49

Implementation details. Following the setting on RobustBench [6], we use ResNet-18 [14], Pre-activation ResNet-18 (PRN-18) [15], and Wide-ResNet-28-10 (WRN-28-10) [55] as the target networks. For training without extra data, we set the number of epochs to 100 for CIFAR10/100 [24], and 30 for SVHN [34]. We adopt Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum factor 0.9 [33], cyclic learning rate schedule [45] with a maximum learning rate of 0.1, and a weight decay factor of 5×10^{-4}. We adopt PGD method [29] with 10 steps for adversary generation during the training stage. The maximum ℓ_∞-norm of adversarial perturbation is $\epsilon = 8/255$, while the step size α is set as $2/255$ for CIFAR-10/100 and $1/255$ for SVHN following common practices. We set the inverse adversary radius as $\epsilon' = 4/255$. The regularization hyper-parameters β and γ are set to 1.0 and 0.9 in Equation (4) and Equation (8). More details can be found in Appendix A.2.

4.2. Results

Performance of UIAT. We compare our proposed UIAT method with state-of-the-art adversarial training schemes as shown in Table 1. We report the accuracies on natural examples as well as adversarial examples obtained using three strong adversarial attacks: PGD [29] with 20 steps (step size $\alpha = 2/255$), CW [4], and Auto Attack (AA) [7] for a rigorous robustness evaluation. Note that AA is a reliable and powerful ensemble attack that contains three types of white-box attack as well as a strong black-box one. Not only does our method enhance robust accuracy on these three datasets, but it also achieves a better clean accuracy, hence a smaller robustness gap. For CIFAR-100, our method significantly boosts the AA robust accuracy by nearly 2% whilst improving the natural accuracy. Our superior performance on CIFAR-100 also represents the generalizability of UIAT on a more complicated dataset with more classes. In addition, we demonstrate that our UIAT with one-off inverse adver-

Table 2. Time cost comparison of adversarial training methods on CIFAR-10 dataset with different network architectures. We report the average training time (min/epoch) of these methods.

Method	ResNet-18	WRN-28	WRN-34
Natural Training	0.35	0.93	1.22
TRADES [57]	2.57	14.13	16.60
HAT [36]	4.02	16.88	18.95
IAT	2.83	15.37	17.82
UIAT	2.20	11.90	14.77
UIAT (One-off)	1.96	10.74	13.36

Table 3. Adversarial robustness results under different attack configurations using ResNet-18 on CIFAR-10. We present natural accuracy and (Auto-Attack) robust accuracy of different attack radii.

ϵ	Method	Natural	Robust
10/255	TRADES [57]	82.28	38.55
	HAT [36]	81.94	40.12
	UIAT	82.79	40.61
	UIAT (One-off)	82.76	41.16
12/255	TRADES [57]	79.37	31.84
	HAT [36]	79.43	33.28
	UIAT	79.50	34.32
	UIAT (One-off)	79.30	34.61
16/255	TRADES [57]	74.89	18.70
	HAT [36]	74.45	19.42
	UIAT	74.29	21.82
	UIAT (One-off)	74.86	21.96

sary generation can also obtain a similar performance as the standard version of UIAT. In other words, the freezing of well-learned class-specific perturbations can still facilitate the distribution alignment for robustness improvement.

Computational cost comparison. In addition to outperforming state-of-the-art adversarial training methods on natural accuracy and robustness, our UIAT method also has a faster training speed. We compare the average training time (min/epoch) of our methods against other adversarial training methods, as presented in Table 2. For a fair comparison, we conduct all the training experiments on a single NVIDIA Tesla A100 GPU with the same batch size $m = 128$ on the CIFAR-10 dataset using three different network architectures. It can be seen that our one-off UIAT method has an additional 51% time efficiency gain with respect to the state-of-the-art adversarial training method, *i.e.*, HAT [36] with ResNet-18. Note that the major time gap between IAT and UIAT comes from the difference in iteration times. IAT requires an instance-wise iterative inverse adversary generation manner, whilst UIAT only performs a single gradient descent step on each example.

Adversarial training on large ϵ. Besides the frequently-used attack configuration, we also train ResNet-18 with our UIAT method for larger ϵ. Specifically, we report the robustness results of the one-off version of the UIAT method on CIFAR-10 under different ℓ_∞-norm radii: $10/255$; $12/255$; $16/255$. As shown in Table 3, we observe that our UIAT method can achieve better robustness results while preserving a comparable natural accuracy as HAT [36] when facing stronger adversarial attacks.

Table 4. Comparison of adversarial training methods using different networks on CIFAR-10/CIFAR-100 with extra training data. We report natural accuracy and (Auto-Attack) robust accuracy.

Dataset	Architecture	Method	Natural	Robust
CIFAR-10	PRN-18	Rebuffi *et al.* [37]	83.53	56.66
		HAT [36]	86.86	57.09
		UIAT	87.34	58.46
		UIAT (One-off)	87.10	58.15
	WRN-28-10	Rebuffi *et al.* [37]	85.97	60.73
		HAT [36]	88.16	60.97
		UIAT	88.93	61.32
		UIAT (One-off)	88.50	61.40
CIFAR-100	PRN-18	Rebuffi *et al.* [37]	56.87	28.50
		HAT [36]	61.50	28.88
		UIAT	62.20	29.40
		UIAT (One-off)	61.54	28.90
	WRN-28-10	Rebuffi *et al.* [37]	59.18	30.81
		HAT [36]	62.21	31.16
		UIAT	63.26	31.18
		UIAT (One-off)	62.45	31.43

Adversarial training with additional data. Following the experimental settings of [13, 36, 37], we also conduct several experiments to measure the generalizability of our method with extra data. Particularly, we present the robustness results using different model architectures trained on CIFAR-10 and CIFAR-100 with 1M synthetic images produced by the Denoising Diffusion Probabilistic Model (DDPM) [19] as the additional data. We compare our UIAT method and its one-off variant version with state-of-the-art approaches in Table 4. Note that we do not apply the CutMix operation [54] following [36]. As observed, our method obtains better robust accuracy while maintaining the same or even better natural accuracy.

4.3. Single-Step Adversarial Training

The computational cost for multi-step adversarial training is expensive, which has become prohibitive to adversarially train on larger models/datasets. In comparison, single-step methods try to approximate the most harmful adversarial examples with a single gradient ascent step [1, 9, 23, 51] during training. Nevertheless, there still exists a considerable robustness gap between single-step adversarial training methods and multi-step ones.

In this section, we combine the one-off version of our UIAT method with state-of-the-art single-step adversarial training approaches to demonstrate the generalizability and the low time cost of our methods. For time efficiency, we set $\beta = 0$ for Equation (4), which means that we only use cross-entropy loss for inverse adversary generation. More

Table 5. Robustness results of single-step adversarial training methods combined with our one-off UIAT approach on CIFAR-10. We conduct single-step adversarial training with various adversarial radii for comprehensive evaluation. We present the natural accuracy, (Auto-Attack) robust accuracy, and the average time for training an epoch.

Method	$\epsilon = 6/255$		$\epsilon = 8/255$		$\epsilon = 10/255$		Time(s)
	Natural	Robust	Natural	Robust	Natural	Robust	
N-FGSM [9]	84.66	56.36	80.29	48.24	75.59	41.54	48.4
N-FGSM + **UIAT**	85.53	58.21	81.85	49.84	77.85	42.77	57.3
RS-FGSM [51]	86.72	55.28	84.07	46.15	86.32	0.00	32.4
RS-FGSM + **UIAT**	87.60	55.85	85.18	46.31	88.29	0.00	40.7
GradAlign [1]	83.85	55.25	80.17	46.57	76.46	39.85	96.0
GradAlign + **UIAT**	85.52	55.46	82.31	46.74	79.11	39.56	107.8

Table 6. Ablation study using ResNet-18 of three component modules of UIAT for adversarially robust accuracy (%) on CIFAR-10.

	UAG	FR	IAM	Natural	PGD-20	AA
1				83.97	53.98	48.33
2	✓			85.19	53.56	47.63
3	✓	✓		85.11	54.13	48.47
4	✓		✓	84.85	54.29	48.83
5	✓	✓	✓	85.01	54.63	49.11

UAG: Universal Adversary Generation.

FR: Feature-level Regularization.

IAM: Inverse Adversary Momentum.

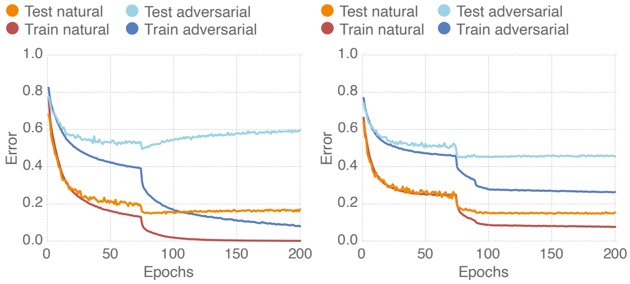
details about how to combine our UIAT method with single-step adversarial training can be found in Appendix B.3. As shown in Table 5, we can observe that UIAT can serve as a plug-and-play component for boosting both natural and robust accuracy. Moreover, we show that our method can effectively adapt to various adversarial training radii for better performance. The additional computational cost for the UIAT method is also acceptable. For instance, in the case of N-FGSM [9], our method can further improve nearly 1.5% for both natural accuracy and adversarially robust accuracy ($\epsilon = 8/255$) with only about an additional 9 seconds time cost for each training epoch.

5. Analysis

5.1. Ablation Study

In this section, we thoroughly investigate the contributions of three components in our UIAT method: 1) Universal Adversary Generation (UAG) in Equation (6), 2) Feature-level Regularization (FR) in Equation (4), and 3) Inverse Adversary Momentum (IAM) in Equation (8). We report both natural accuracy and robust accuracy on CIFAR-10 using ResNet-18 during the ablation study in Table 6.

Our baseline method (The first row in Table 6) is the instance-wise Inverse Adversarial Training (IAT), which has already achieved a competitive robustness performance compared to other methods. It can be seen that the universal inverse adversary generation can effectively improve the performance on natural accuracy, while the robust accuracy slightly drops. Both the feature-level regularization and the inverse adversarial momentum contribute to enhancing the adversarially robust accuracy. We can obtain our UIAT method by integrating these three components, which can effectively improve natural accuracy and robustness.



(a) PGD-AT

(b) **UIAT** (one-off)

Figure 2. The learning curves show the natural and robust accuracy (under PGD-20) on the training/test set of CIFAR-10. Note that (a) represents PGD-AT [29], while (b) is our one-off UIAT method.

5.2. Robust Overfitting

Recent research has demonstrated that adversarial training methods primarily suffer from the robust overfitting issue [38], resulting in the robustness plunge. The robust overfitting induces an irreversible robustness drop (on the test set) after adversarial training for several epochs, especially after the learning rate decay operation. We illustrate the learning curves of standard adversarial training and our one-off version of UIAT in Fig. 2.

For better visualization, we increase the number of training epochs to 200. We can easily observe that the PGD-based Adversarial Training (PGD-AT) [29] severely suffers from the robust overfitting issue. In comparison, our one-off UIAT method can largely mitigate the robust overfitting issue, that is our method does not suffer from a sharp robustness reduction during adversarial training. It can potentially be explained by the observation made in [11], which demonstrates that the robust overfitting comes from the large-loss data during adversarial training. However, our UIAT method implicitly regularizes the large-loss data, *a.k.a.*, misclassified examples to obtain the high-likelihood

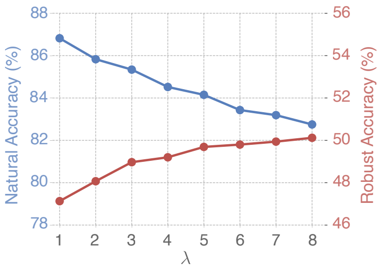
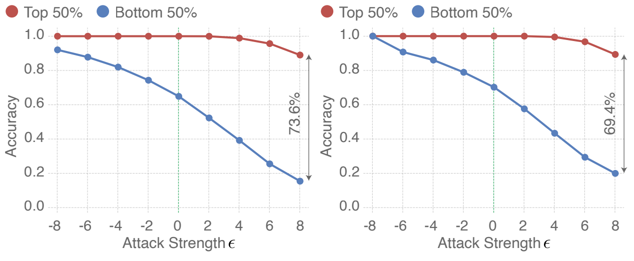


Figure 3. Parameter sensitivity of our one-off UIAT method by tuning the hyper-parameter λ. We report both natural accuracy and (Auto-Attack) robust accuracy.



(a) TRADES [57]

(b) **UIAT** (One-off)

Figure 4. Average accuracy under different attack strengths (performed by PGD-20). Note that the experimental settings are the same as Figure 1. We further annotate the robust accuracy gap between two groups under the attack strength $\epsilon = 8/255$.

region of their true classes, which can thus mitigate the robust overfitting problem. Furthermore, our inverse adversary momentum can also stabilize the training stage by mitigating the oscillations of noisy predictions.

5.3. Trade-off

The trade-off between natural accuracy and robust accuracy during adversarial training has been widely explored [36, 57]. We study the effect of hyper-parameter λ, which can further induce a trade-off between robustness and natural accuracy, as shown in Figure 3. We can observe that the robust accuracy improves when λ increases, while the natural accuracy decreases. Oppositely, the natural accuracy improves when we lower the λ value. Note that our trade-off is different from [57] that balances the importance of cross-entropy of natural examples and KL divergence, whilst our UIAT method optimizes the cross-entropy of adversarial examples and the regularized distribution matching. We also provide an analysis of the effect of other hyper-parameters in Appendix D.

5.4. Why our method is effective?

In this section, we mainly discuss the underlying reason why our method is effective. In other words, we would like to explore what we have gained from inverse adversarial training. Similar to the setting in Figure 1, we also provide the average accuracy under different attack strengths of our UIAT method compared with TRADES [57], as shown in Figure 4. It can be seen that our method can bridge the robust accuracy gap more effectively compared with TRADES [57]. Precisely, our UIAT can effectively enhance the robust accuracy of the bottom 50% group. In addition, we observe that the inverse adversarial examples of UIAT are prone to be classified correctly, which means that our robust model is easily affected by inverse adversaries. On the contrary, our robust model is less susceptible to adversarial examples compared with TRADES [57].

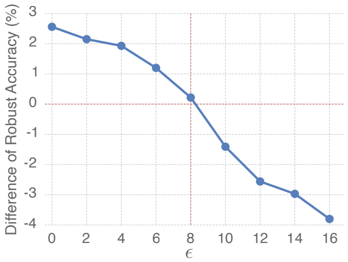


Figure 5. Difference of Auto-Attack (AA) robust accuracy under different attack strengths between our UIAT (One-off) method and TRADES [57]. The red lines are used for reference.

Furthermore, we present the comparison of Auto-Attack (AA) robust accuracy under different attack strengths between our UIAT (One-off) method and TRADES [57], as shown in Figure 5. It can be easily observed that our method outperforms TRADES [57] at weak attack strengths ($\epsilon \leq 8/255$). However, TRADES [57] obtains better robustness than our method when confronted with strong adversarial perturbations ($\epsilon > 8/255$). In other words, our method sacrifices the adversarial robustness against larger visual perturbations to enhance the robustness against smaller ones. This is also in line with the definition and intuition that adversarial perturbations are visually undetectable. Recall that we can also obtain better robustness against larger perturbations when training with larger ϵ as discussed in Section 4.2.

6. Conclusion

In this paper, we explore the unnecessary alignment between misclassified examples and propose a new adversarial training paradigm incorporating the inverse adversarial examples. Furthermore, we design a universal inverse adversary generation strategy to mitigate the class-wise imbalance hiding in the decision surface and accelerate our methods. Extensive experiments demonstrate that our method can efficiently obtain better robustness results without compromising natural accuracy in diverse settings on larger datasets. Moreover, we can obtain a trade-off between natural accuracy and robust accuracy to adapt to different scenarios. Our UIAT method can also be combined with state-of-the-art single-step adversarial training methods for robustness enhancement at a low cost. Finally, we analyze the reason why our method is effective and verify that our UIAT method can potentially bridge the accuracy gap between high-accuracy examples and low-accuracy examples, thus benefiting the robustness.

References

[1] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020. 3, 6, 7, 12

[2] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4312–4321, 2021. 2

[3] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017. 1

[4] 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. 2, 5

[5] 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. 11

[6] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. 5

[7] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020. 5

[8] Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15721–15730, 2021. 1, 2

[9] 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. 6, 7, 12

[10] Junhao Dong, Yuan Wang, Jian-Huang Lai, and Xiaohua Xie. Improving adversarially robust few-shot image classification with generalizable representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9025–9034, 2022. 2

[11] Yinpeng Dong, Ke Xu, Xiao Yang, Tianyu Pang, Zhijie Deng, Hang Su, and Jun Zhu. Exploring memorization in adversarial training. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 7

[12] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 1, 2

[13] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[15] 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. 5, 11

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 11

[17] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *5th International Conference on Learning Representations, ICLR*, 2017. 2

[18] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 1

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 6, 11

[20] 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. 11

[21] Xiaojun Jia, Yong Zhang, Baoyuan Wu, Ke Ma, Jue Wang, and Xiaochun Cao. Las-at: Adversarial training with learnable attack strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13398–13408, 2022. 2

[22] Mostafa Kahla, Si Chen, Hoang Anh Just, and Ruoxi Jia. Label-only model inversion attacks via boundary repulsion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15045–15053, 2022. 3

[23] Hoki Kim, Woojin Lee, and Jaewook Lee. Understanding catastrophic overfitting in single-step adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8119–8127, 2021. 6

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4, 5, 11

[25] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018. 1

[26] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. Collaborative adversarial training. *arXiv preprint arXiv:2205.11156*, 2022. 2

[27] Feng Liu, Bo Han, Tongliang Liu, Chen Gong, Gang Niu, Mingyuan Zhou, Masashi Sugiyama, et al. Probabilistic margins for instance reweighting in adversarial training. *Advances in Neural Information Processing Systems*, 34:23258–23269, 2021. 2

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[29] 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, 3, 5, 7, 11, 12

[30] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. 4

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2

[33] 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. 5, 11

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 4, 5, 11

[35] Luis S Piloto, Ari Weinstein, Peter Battaglia, and Matthew Botvinick. Intuitive physics learning in a deep-learning model inspired by developmental psychology. *Nature human behaviour*, pages 1–11, 2022. 1

[36] 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. 5, 6, 8, 11

[37] 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. 6, 11

[38] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020. 3, 7

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[40] Hadi Salman, Andrew Ilyas, Logan Engstrom, Sai Vemprala, Aleksander Madry, and Ashish Kapoor. Unadversarial examples: Designing objects for robust vision. *Advances in Neural Information Processing Systems*, 34:15270–15284, 2021. 2

[41] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *6th International Conference on Learning Representations, ICLR*, 2018. 2

[42] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019. 3

[43] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020. 4

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR*, 2015. 1

[45] 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. 5, 11

[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR*, 2014. 1, 2

[47] Shixin Tian, Guolei Yang, and Ying Cai. Detecting adversarial examples through image transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[48] 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. 1, 2, 3, 4, 5, 12

[49] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016. 3

[50] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018. 1

[51] 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. 3, 6, 7, 12

[52] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating adversarial effects through randomization. In *6th International Conference on Learning Representations, ICLR*, 2018. 1, 2

[53] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *International Conference on Machine Learning*, pages 12062–12072. PMLR, 2021. 2

[54] 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*, pages 6023–6032, 2019. 6

[55] 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. 5, 11

[56] Varvara E Zemskova, Tai-Long He, Zirui Wan, and Nicolas Grisouard. A deep-learning estimate of the decadal trends in

the southern ocean carbon storage. *Nature communications*, 13(1):1–11, 2022. 1

[57] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019. 1, 2, 3, 5, 6, 8

[58] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan S. Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *9th International Conference on Learning Representations, ICLR*, 2021. 2

[59] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019. 1

Appendix

A. Experimental Settings

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

A.1. Datasets

We conduct all our experiments on CIFAR-10/100 [24] and SVHN [34]. 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) [19] for CIFAR-10/100 following the setting of [36, 37].

A.2. Implementation Details

Following the hyper-parameters setting from [5, 36], we use Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum factor 0.9 [33] cyclic learning rate schedule [45] 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 [37] 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 [36, 37], we use SiLU activation function [16] with Pre-activation ResNet-18 (PRN-18) [15] and Wide-ResNet-28-10 (WRN-28-10) [55]. We further use model weight averaging [20] with a decay factor of 0.995.

For computing adversarial examples during training, we adopt the iterative Projected Gradient Descent (PGD) algorithm [29] 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 radius $\epsilon = 8/255$. We set the inverse perturbation radius as $\epsilon' = 4/255$. The iteration steps for instance-wise inverse perturbation is 5 times with the 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-parameters 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 training epochs and $T' = 320$ for 400 training epochs.

B. Details of Inverse Adversarial Training

B.1. Instance-wise Inverse Adversarial Training

We have introduced how to generate instance-wise inverse adversaries in the main body of this 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 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 2.

B.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 exam-

Algorithm 2 **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 $\tau$; radius for adversaries $\epsilon$ and inverse adversaries $\epsilon'$; iteration times $n$ and step size $\alpha'$ for inverse adversary generation; weighting factors $\lambda$, $\beta$.
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 $\mathbf{z}_j \sim 0.001 \cdot \mathcal{N}(0, 1)$
            $\hat{\mathbf{x}}_j \leftarrow$ PGDATTACK$(\mathbf{x}_j, y_j, f_{\boldsymbol{\theta}})$                ▷ Find PGD adversarial example
            $\check{\mathbf{x}}_j \leftarrow \mathbf{x}_j + \mathbf{z}_j$
            for $t = 1, 2, \ldots, n$ do
                $\check{\mathbf{x}}_j = \Pi_{\mathbb{B}(\mathbf{x}, \epsilon')}\left(\check{\mathbf{x}}_j - \alpha' \cdot \operatorname{sign}\left(\nabla_{\check{\mathbf{x}}_j} \mathcal{L}_{Inv}\left(\check{\mathbf{x}}_j, y\right)\right)\right)$     ▷ 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 $f_{\boldsymbol{\theta}}$.
```

ples. 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:

$$\mathcal{L}_{\mathbf{IAT}}^{OO} = \mathcal{L}_{CE}\left(f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right), y\right) + \lambda \cdot \mathcal{L}_{KL}\left(\mathbf{p}_{OO}^{(t)} \| f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right)\right), \quad (9)$$

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:

$$\mathbf{p}_{OO}^{(t)} = \begin{cases} f_{\boldsymbol{\theta}}\left(\mathbf{x}\right), & \text{if } t < T' \\ f_{\boldsymbol{\theta}}\left(\check{\mathbf{x}}\right), & \text{if } t = T' \\ \mathbf{p}_{OO}^{(T)}, & \text{if } t > T' \end{cases} \quad (10)$$

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 [48]. 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 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.

B.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, 9, 51]. When using ℓ_∞-norm threat model, we can formalize the adversarial training [29] as the following min-max optimization problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|_\infty < \epsilon} \mathcal{L}_{CE}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x} + \boldsymbol{\delta}^{SGL}\right), y\right)\right], \quad (11)$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss and δ is the adversarial perturbation under the ℓ_∞-norm bound ϵ. The inner maximization of adversarial training can be viewed as searching for the most harmful adversarial examples $\hat{\mathbf{x}}^{SGL} = \mathbf{x} + \boldsymbol{\delta}^{SGL}$. Particularly, most single-step adversarial training methods approximate the worst-case perturbation by solving the inner maximization in Equation (11) with the following form:

$$\boldsymbol{\delta}^{SGL} = \psi\Big(\boldsymbol{\eta} + \alpha \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}_{CE}(f_\theta(\mathbf{x} + \boldsymbol{\eta}), y)\right)\Big), \quad (12)$$

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 $\boldsymbol{\delta}^{SGL}$. We primarily focus on outer minimization, where we add an additional KL divergence term between universal inverse adversaries $\check{\mathbf{x}}$ and adversarial examples $\hat{\mathbf{x}}^{SGL}$. Hence, a general form of the loss function for single-step adversarial training (outer minimization) can be defined as below:

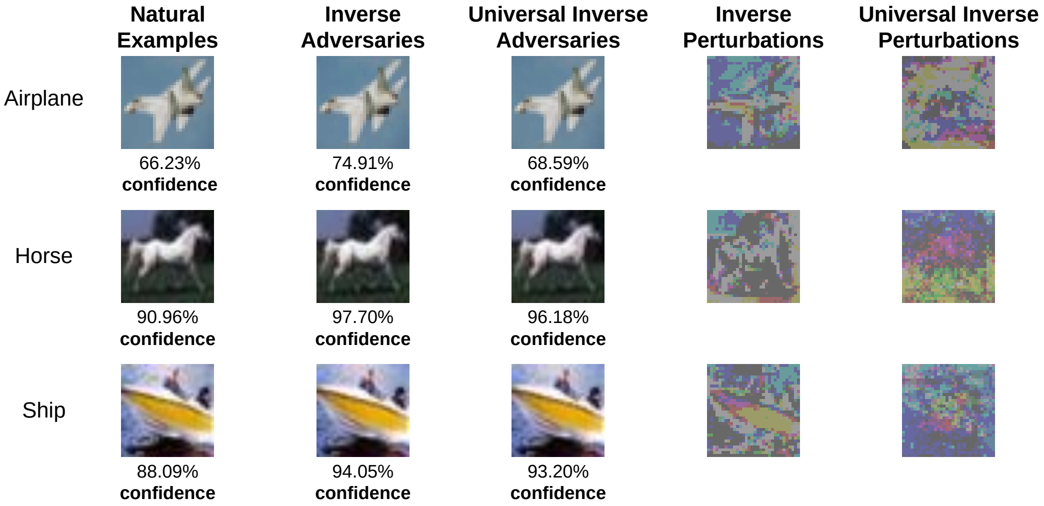


Figure 6. Visualization of both inverse adversaries and class-specific universal inverse adversaries. Their corresponding inverse adversarial perturbations are also presented.

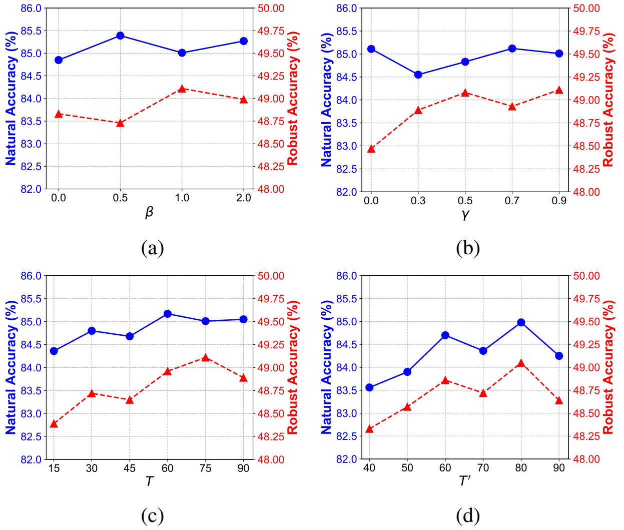


Figure 7. Hyper-parameter sensitivity of our UIAT method on natural accuracy and (Auto-Attack) robust accuracy using ResNet-18 on CIFAR-10. We report the hyper-parameters adjustment of β in (a), γ in (b). The tuning for the starting epoch of momentum T is in (c), and the one-off epoch T' is in (d)

$$\mathcal{L}_{\mathbf{IAT}}^{SGL} = \mathcal{L}_{CE}\left(f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right), y\right) + \lambda \cdot \mathcal{L}_{KL}\left(f_{\boldsymbol{\theta}}\left(\check{\mathbf{x}}\right) \| f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}^{SGL}\right)\right), \tag{13}$$

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 cross-entropy loss for inverse adversary generation. In general, we efficiently combine our method with single-step adversarial training by paying only three additional forward propagation times and one backward propagation time.

C. Visualization

We visualize both the (universal) inverse adversarial examples and their inverse perturbations in Figure 6. It can be seen that class-specific universal inverse adversaries can obtain a similar inverse effect to the original inverse adversaries. These inverse examples are also visually indistinguishable from natural examples.

D. 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 7. 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 start epoch for momentum, it is beneficial to adopt the output probability of inverse adversaries during the second half of training.