A Additional Experiment: Adversarial Examples (in Adversarial Training) Contain Adversarial Non-robust Features

In Section 3, we collect misclassified adversarial examples on an early checkpoint A and evaluate them on a later checkpoint B with their correct labels to examine the memorization of those adversarial non-robust features. This is because adversarial examples used for AT consist of robust features from the original label y and non-robust features from the misclassified label $\hat{y}$, so B must have memorized those features (non-robust on A) to correctly classify them. In this section, we further provide a rigorous discussion on the non-robust features contained in the adversarial examples through an additional experiment.

Experiment Design. Recall that Ilyas et al. [15] leverage targeted PGD attack [22] to craft adversarial examples on a standard (non-robust) classifier and relabel them with their target labels to build a non-robust dataset. Finding standard training on it yields good accuracy on clean test data, they prove that those adversarial examples contain non-robust features corresponding to the target labels (Section 3.2 of their paper).

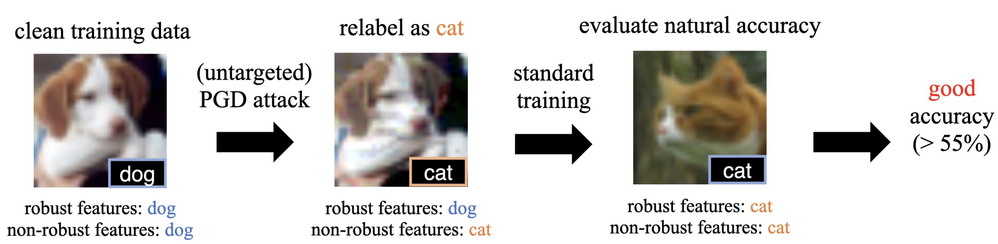


Figure 7: Mining non-robust features from adversarial examples in adversarial training. 1) Craft adversarial examples $\hat{x}_i$ using untargeted PGD attack on a AT checkpoint. 2) Relabel them (only misclassified ones) with misclassified label $\hat{y}_i$ to build a non-robust dataset $\{(\hat{x}_i, \hat{y}_i)\}$. 3) Perform standard training from the checkpoint. 4) Achieve good natural accuracy ($> 55\%$).

However, different from their settings where we know the target labels of the adversarial examples and they are generated on a non-robust classifier, we lay emphasis on mining non-robust features from adversarial examples generated on-the-fly in AT. To this end, we first craft adversarial examples on the AT checkpoint at some epoch using untargeted PGD attack [22] following the real setting of AT, then relabel the *misclassified* ones $\hat{x}_i$ (*e.g.,* a dog) with their misclassified labels $\hat{y}_i \neq y_i$ (*e.g.,* cat) to build a non-robust dataset $\{(\hat{x}_i, \hat{y}_i)\}$. In order to capture non-robust features at exactly the training epoch these adversarial examples are used, we continue to perform standard training directly from the AT checkpoint on the non-robust dataset, and finally evaluate natural accuracy. See Figure 7 for an illustration of our experiment.

In details, we select the AT models at epoch 60 (before LR decay) and epoch 1,000 (after LR decay) to conduct the above experiment. We use untargeted PGD-20 with perturbation norm $\varepsilon_\infty = 16/255$ to craft adversarial examples on those checkpoints from the *training data* of CIFAR-10 [17]. The attack we adopt is a little bit stronger than the attack of PGD-10 with $\varepsilon_\infty = 8/255$ that is commonly used to generate adversarial examples in baseline adversarial training, because the training robust accuracy rises to as high as 94.67% at epoch 1,000 (Figure 1a) and gives less than 2,700 misclassified examples, which is significantly insufficient for further training. Using the stronger attack, we obtain a success rate of 78.38% on the checkpoint at epoch 60 and a success rate of 34.86% on the checkpoint at epoch 1,000. Since the attack success rates are different, we randomly select a same number of misclassified adversarial examples for standard training for a fair comparison. The learning rate is initially set to 0.1 and decays to 0.01 after 20 epochs for another 10 epochs of fine-tuning. Given that the original checkpoints already have non-trivial natural accuracy, we also add two control groups that train with random labels instead of $\hat{y}$ to exclude the influence that may brought by the original accuracy.

Results. As shown in Figure 8, we find that standard training on the non-robust dataset $\{(\hat{x}_i, \hat{y}_i)\}$ successfully converges to fairly good accuracy ($> 55\%$) on natural test images, *i.e.,* predicting cats as cats, no matter from which AT checkpoints (either epoch 60 or epoch 1,000) the adversarial examples are crafted. Also, we can see that the non-trivial natural accuracy has nothing to do with the original accuracy of the AT checkpoints, as the accuracy plummets to around 10% (random guessing) as soon as the standard training starts with random labels. This proves that *adversarial examples in*

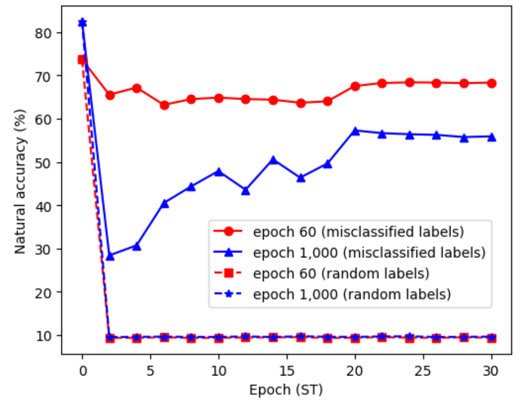


Figure 8: Natural accuracy during standard training. Standard training on the non-robust dataset built from the checkpoint at either epoch 60 or epoch 1,000 converges to fairly good natural accuracy ($> 55\%$). The failure of training with random labels proves that the good accuracy has nothing to do with the original natural accuracy of the AT checkpoint.

adversarial training do contain non-robust features w.r.t. the classes to which they are misclassified, which strongly corroborates the validity of the experiments in Section 3.

Discussion on the Influence Brought by Robust Features During Standard Training. Since untargeted PGD attack cannot be assigned with a target label, we cannot guarantee that the misclassified labels $\hat{y}$ to be uniformly distributed regardless of the original labels (especially for real-world datasets). This implies that the non-robust features are not completely decoupled from robust features, *i.e.*, training on $\{(\hat{x}_i, \hat{y}_i)\}$ may take advantage of $\hat{x}_i$'s robust features from y_i through the correlation between y_i and $\hat{y}_i$. However, we argue that robust features from y_i mingling with non-robust features from $\hat{y}_i$ only increases the difficulty of obtaining a good natural accuracy. This is because learning through the shortcut will only wrongly map the robust features from y_i to $\hat{y}_i$ that never equals to y_i, but during the evaluation, each clean test example x_i will always have robust and non-robust features from y_i, and such wrong mapping will induce x_i to be misclassified to some $\hat{y}_i$ due to the label of its robust features. As a result, despite the negative influence brought by robust features, we still achieve good natural accuracy at last, which further solidifies our conclusion.

B Experiment Details

In this section, we provide more experiment details that are omitted before due to the page limit.

B.1 Baseline Adversarial Training

In this paper, we mainly consider classification task on CIFAR-10 [17]. The dataset contains 60,000 32×32 RGB images from 10 classes. For each class, there are 5,000 images for training and 1,000 images for evaluation. Since we mainly aim to track the training dynamic to verify our understandings in RO instead of formally evaluating an algorithm's performance, we do not hold a validation set in most of our verification experiments and directly train on the full training set.

For baseline adversarial training, We use PreActResNet-18 [12] model as the classifier. We use PGD-10 attack [22] with step size $\alpha = 2/255$ and perturbation norm $\varepsilon_\infty = 8/255$ to craft adversarial examples on-the-fly. Following the settings in Madry et al. [22], we use SGD optimizer with momentum 0.9, weight decay 5×10^{-4} and batch size 128 to train the model for as many as 1,000 epochs. The learning rate (LR) is initially set to be 0.1 and decays to 0.01 at epoch 100 and further decays to 0.001 at epoch 150. For the version without LR decay used for comparison in our paper, we simply keep the LR to be 0.1 during the whole training process.

Each model included in this paper is trained on a single NVIDIA GeForce RTX 3090 GPU. For PGD-AT, it takes about 3d 14h to finish 1,000 epochs of training.

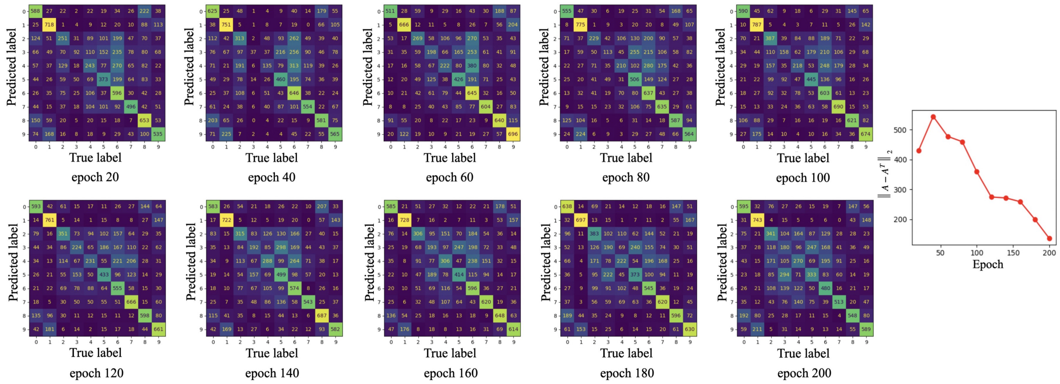
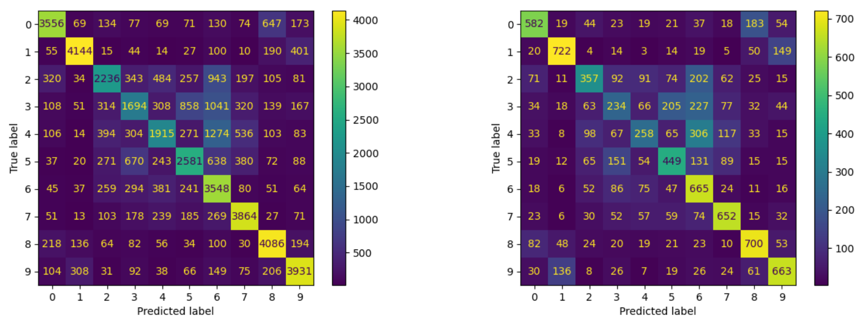


Figure 9: More test-time confusion matrices during the first 200 epochs of the training. After LR decays (the second row), the confusion matrix A immediately becomes symmetric, as the spectral norm $\|A - A^T\|_2$ *w.r.t.* the matrix decreases from ≥ 350 before epoch 100 to ≤ 150 after epoch 200.



(a) train (b) test

Figure 10: Confusion matrices of the training and the test data at an epoch before robust overfitting starts. They show nearly a same pattern of attacking preference among classes.

B.2 Verification Experiments for Our Minimax Game Perspective on Robust Overfitting

Memorization of Adversarial Non-robust Features After LR Decay. We craft adversarial examples on a checkpoint before LR decay (60th epoch) and evaluate the *misclassified* ones on a checkpoint after LR decay ($\geq$ 150th epoch) with their correct labels to evaluate the memorization of non-robust features in the training adversarial examples (see Appendix A for detailed discussions) in Section 3.1 and 3.2. Following Appendix A, we adopt PGD-20 attack with perturbation norm $\varepsilon_\infty = 16/255$ to craft adversarial examples which is stronger than the common attack setting we use in PGD-AT. We note that test adversarial examples crafted by a stronger attack indicates stronger extraction of the non-robust features, so they are more indicative of non-robust feature memorization when still correctly classified.

Verification I: More Non-robust Features, Worse Robustness. At the beginning of Section 3.2, we create synthetic datasets to demonstrate that memorizing the non-robust training features indeed harms test-time model robustness. To instill non-robust features into the training dataset, we minimize the adversarial loss *w.r.t.* the training data in a way that just like PGD attack, with the only difference that we minimize the adversarial loss instead of maximizing it. Since we only use a very small perturbation norm $\varepsilon_\infty \leq 4/255$, the added features are bound to be non-robust. For a fair comparison, we also perturb the training set with random uniform noise of the same perturbation norm to exclude the influence brought by (slight) data distribution shifts. We continue training from the 100-th baseline AT checkpoint (before LR decay) on each synthetic dataset for 10 epochs, and then evaluate model robustness with clean test data.

Verification II: Vanishing Target-class Features in Test Adversarial Examples. This is to say that when RO happens, we expect that test adversarial examples become less informative of the classes to which they are misclassified according to our theory. To verify this, we first craft adversarial examples on a checkpoint T after RO begins, then evaluate the misclassified ones *with their misclassified labels* $\hat{y}$ on the checkpoint saved at epoch 60. As a result, the accuracy reflects how much information (non-robust features) from $\hat{y}$ the adversarial examples have to contain to be misclassified to $\hat{y}$ on T. All adversarial examples evaluated in the experiments in Section 3.2.2 are crafted using PGD-10 attack with perturbation norm $\varepsilon_\infty = 8/255$.

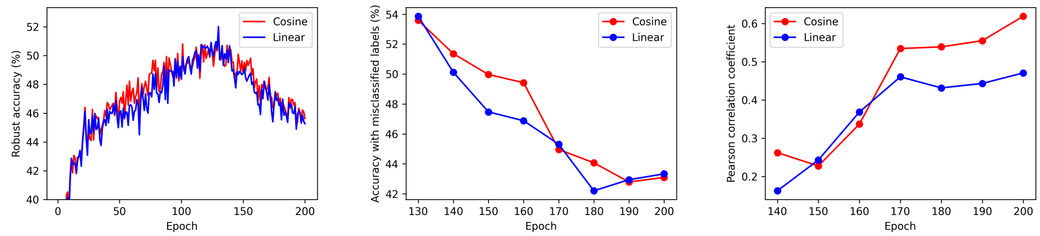
Verification III: Bilateral Class Correlation. To quantitatively analyze the correlation strength of bilateral misclassification described in Section 3.2.2, we first summarize all $y_i \to y_j$ misclassification rates into two confusion matrices $P^{\text{train}}, P^{\text{test}} \in \mathbb{R}^{C\times C}$ for the training and test data, respectively. Because we are mainly interested in the effect of the LR decay, we focus on the relative change on the test confusion matrix before and after LR decay, *i.e.,* $\Delta P^{\text{test}} = P^{\text{test}}_{\text{after}} - P^{\text{test}}_{\text{before}}$. According to our theory, for each class pair (i, j), there should be a strong correlation between the training misclassification of $i \to j$ before LR decay, *i.e.,* $(P^{\text{train}}_{\text{before}})_{ij}$, and the *increase* in test misclassification of $j \to i$, *i.e.,* $\Delta P^{\text{test}}_{ji}$, as $i \to j$ training misclassification (may due to intrinsic class bias, as will be further discussed below in details) induces $i \to j$ false mappings and creates $j \to i$ shortcuts. To examine their relationship, we plot the two variables $((P^{\text{train}}_{\text{before}})_{ij}, \Delta P^{\text{test}}_{ji})$ and compute their Pearson correlation coefficient ρ.

Verification IV: Symmetrized Confusion Matrix. In Section 3.2.2, we mention the growing symmetry of the test-time confusion matrix after LR decay as an evidence of the strengthening $y \to y'$ and $y' \to y$ correlation. Here we present more confusion matrices during the 200 epochs of the training in Figure 9, and it is very clear that the confusion matrices soon become symmetric after LR decay and RO starts. For a deeper comprehension of this phenomenon, we first visualize the confusion matrices of the training and the test data at an epoch before RO starts in Figure 10. They exhibit nearly a same pattern of attacking preference among classes (*e.g.,* $y \to y'$) due to the bias rooted in the dataset, *e.g.,* class 6 is intrinsically vulnerable in this case. For the test data, this intrinsic bias wouldn't be wiped out through learning due to the non-generalizability of the memorization of non-robust features in the training data, as discussed at the beginning of Section 3.2 (*i.e.,* $y \to y'$ bias still holds); and for the training data, this biased feature memorization will open shortcuts for test-time adversarial attack as discussed in Section 3.2 (*i.e.,* $y' \to y$ begins). Combining both $y \to y'$ and $y' \to y$, we arrive at the symmetry of test-time confusion matrix.

Additional Results: Changing LR Decay Schedule. Our discussion above is based on the piecewise LR decay schedule, in which the sudden decay of LR most obviously reflects our understandings. Besides, we also explore other LR decay schedules, including Cosine/Linear LR decay, to check whether different LR decay schedules will affect the observations and claims we made in this paper. For each schedule, we train the model for 200 epochs following the settings in Rice et al. [26]. As demonstrated in Figure 11a, we arrive at the same finding as Rice et al. [26] that with Cosine/Linear LR decay schedule, the training still suffers from severe RO after epoch 130. Then, we rerun the empirical verification experiments in Section 3.2.2 and find that under both the two LR decay schedules 1) the test adversarial examples indeed contain less and less target-class non-robust features as the training goes and RO becomes severer and severer (Figure 11b), 2) the bilateral class correlation becomes increasingly strong (Figure 11c) and 3) the confusion matrix indeed becomes symmetric (Figure 12). The results are almost the same as the results achieved when we adopt piecewise LR decay schedule because even though these LR decay schedules are mild, the LR eventually becomes small and makes the trainer $\mathcal{T}$ overly strong to memorize the harmful non-robust features, indicating that our understandings in the cause of RO is fundamental and regardless of whatever LR decay schedule is used.

Table 3: Training with stronger attack and evaluating model robustness on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

Attack Strength	Natural			PGD-20			AutoAttack		
	best	final	diff	best	final	diff	best	final	diff
$\varepsilon = 8/255$, PGD-10 (baseline)	**83.50**	**84.94**	**-1.44**	55.05	47.60	7.45	49.89	43.83	6.06
$\varepsilon = 10/255$, PGD-12	80.66	82.48	-1.82	56.63	50.63	6.00	50.89	46.13	4.76
$\varepsilon = 12/255$, PGD-15	78.17	80.25	-2.08	**57.09**	53.13	3.96	**50.99**	47.66	3.33
$\varepsilon = 14/255$, PGD-17	73.92	76.70	-2.78	56.42	54.04	2.38	50.28	48.53	1.75
$\varepsilon = 16/255$, PGD-20	69.51	73.11	-3.60	55.09	**54.27**	**0.82**	49.58	**48.53**	**1.05**



(a) test robust accuracy during training (b) target-class non-robust features (c) bilateral class correlation

Figure 11: Empirical verification of our explanation for robust overfitting when Cosine/Linear LR decay schedule is applied. (a) With Cosine/Linear LR decay schedule, the training still suffers from severe RO. (b) After RO begins, non-robust features in the test data become less and less informative of the classes to which they are misclassified. (c) Increasingly strong correlation between training-time $y \rightarrow y'$ misclassification and test-time $y' \rightarrow y$ misclassification increase.

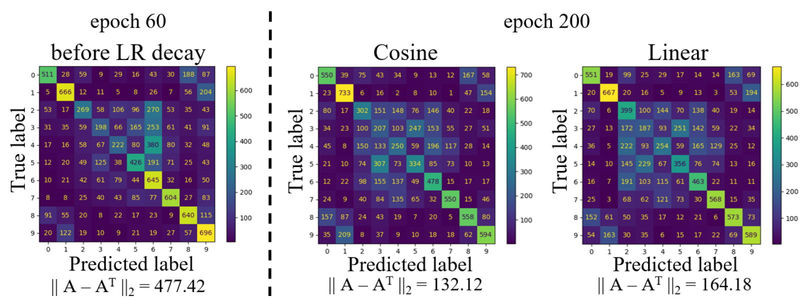


Figure 12: Test-time confusion matrix also becomes symmetric and implies that the bilateral correlation also exists when Cosine/Linear LR decay schedule is applied.

B.3 Experiments on the Effect of Stronger Training Attacker

In Section 4.2, we point out that using a stronger attacker in AT is able to mitigate RO to some extent by neutralizing the trainer $\mathcal{T}$'s fitting power when it is overly strong. To achieve the results reported in Figure 5d, we craft adversarial examples on-the-fly with more PGD iteration steps when ε is larger (see Table 3), and further evaluate the best and last robustness of the WA models against PGD-20 and AA. Although RO is only partially mitigated and natural accuracy decreases when a stronger attacker is applied as summarized in Addepalli et al. [1], it may be surprising to find from Table 3 that an attacker of appropriate strength may significantly boost the best WA robustness. This suggests using a stronger attack could potentially be an interesting new path to stronger adversarial defense, and we leave it for future work

B.4 Detailed Experiment Setup of ReBAT for Mitigating Robust Overfitting

Datasets. Beside CIFAR-10, we also include CIFAR-100 [17] and Tiny-ImageNet [6] for evaluation of the effectiveness of ReBAT. CIFAR-100 shares the same training and test images with CIFAR-10, but it classifies them into 100 categories, *i.e.,* 500 training images and 100 test images for each class. Tiny-ImageNet is a subset of ImageNet [6] which contains labeled 64×64 RGB images from 200 classes. For each class, it includes 500 and 50 images for training and evaluation respectively. Following Rice et al. [26], we hold out 1,000 images from the original CIFAR-10/100 training set, and similarly 2,000 images from the original Tiny-ImageNet training set as validation sets.

Training Strategy. For CIFAR-10 and CIFAR-100, we follow exactly the same training strategy as introduced in Appendix B.1, except that for ReBAT[strong] we adopt PGD-12 with perturbation norm $\varepsilon_\infty = 10/255$ for training after LR decay. For Tiny-ImageNet, we follow the learning schedule of Chen et al. [4], in which the model is trained for a total of 100 epochs and the LR decays twice (by 0.1) at epoch 50 and 80.

Choices of Hyperparameters. For KD+SWA [4], PGD-AT+TE [8], AWP [33] and WA+CutMix [25], we strictly follow their original settings of hyperparameters. For MLCAT$_{\text{WP}}$ [34], we simply report the test results reported in their paper. Following Chen et al. [4], SWA/WA as well as the ReBAT regularization purposed in Section 4.1 start at epoch 105 (5 epochs later than the first LR decay where robust overfitting often begins), and following Rebuffi et al. [25] we choose the EMA decay rate of WA to be $\gamma = 0.999$. Please refer to Table 4 for our choices of the decay factor d and regularization strength λ. We notice that since CutMix improves the difficulty of learning, the model demands a relatively larger decay factor to better fit the augmented data. For Tiny-ImageNet, we also apply a larger λ after the second LR decay to better maintain the flatness of adversarial loss landscape and control robust overfitting. We provide more discussions on the choice of hyperparameters in Appendix C.1.

Table 4: Choices of hyperparameters when training models on different datasets using different network architectures with ReBAT.

Network Architecture	Method	CIFAR-10/CIFAR-100	Tiny-ImageNet
PreActResNet-18	ReBAT	$d = 1.5, \lambda = 1.0$	$d = 4.0, \lambda_1 = 2.0, \lambda_2 = 10.0$
	ReBAT[strong]	$d = 1.7, \lambda = 1.0$	$d = 4.0, \lambda_1 = 2.0, \lambda_2 = 10.0$
	ReBAT+CutMix	$d = 4.0, \lambda = 2.0$	$d = 6.0, \lambda = 1.5$
WideResNet-34-10	ReBAT	$d = 1.3, \lambda = 0.5$	-
	ReBAT[strong]	$d = 1.3, \lambda = 0.5$	-
	ReBAT+CutMix	$d = 4.0, \lambda = 2.0$	-

B.5 Training Robust Accuracy

Figure 13 shows the robust accuracy change on the training data during the training. Compared with vanilla PGD-AT that yields training robust accuracy over 80% at epoch 200, ReBAT manages to suppress it to only 65%. It successfully prevents the trainer $\mathcal{T}$ from learning the non-robust features *w.r.t.* the training data too fast and too well, and therefore significantly reduces the robust generalization gap (from $\sim 35\%$ to $\sim 9\%$) and mitigates RO.

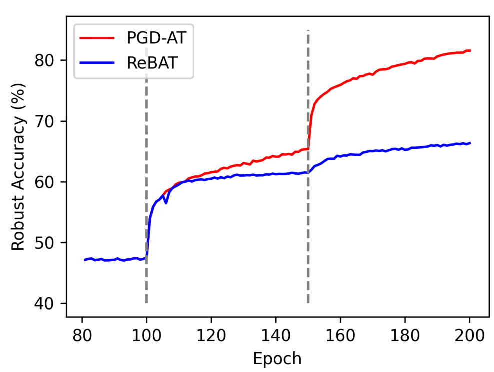


Figure 13: Training robust accuracy of PGD-AT and ReBAT on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

B.6 Training Efficiency

We also test and report the training time (per epoch) of several methods evaluated in this paper. For a fair comparison, all the compared methods are integrated into a universal training framework and each test runs on a single NVIDIA GeForce RTX 3090 GPU.

From Table 5, we can see that ReBAT requires nearly no extra computational cost compared with vanilla PGD-AT (136.2s *v.s.* 131.6s per epoch), implying that it is an efficient training method in practical. We also remark that KD+SWA, one of the most competitive methods that aims to address the RO issue, is not really computationally efficient as it requires to pretrain a robust classifier and a non-robust one as AT teacher and ST teacher respectively.

Table 5: Combining training time per epoch on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

Method	Training Time per Epoch (s)
PGD-AT	131.6
WA	132.1
KD+SWA	131.6+16.5+141.7
AWP	142.8
MLCAT_{wp}	353.3
ReBAT	136.2
WA+CutMix	168.6
ReBAT+CutMix	173.1

C More Experiments on ReBAT

In this section, we conduct extensional experiments on the proposed ReBAT method to further demonstrate its effectiveness, efficiency and flexibility.

C.1 Additional Results on BoAT Loss

In Section B.4, we discuss the detailed configurations for the experiments in Figure 5a, where we show that BoAT can largely mitigate robust overfitting. Here, we further summarize the performance of best and final checkpoints of the original AT+WA method and our BoAT. As shown in Table 6, BoAT not only boosts the best robustness by a large margin (0.67% higher against AA) but also significantly suppresses RO (1.58% *v.s.* 6.06% against AA). To achieve the reported robustness, we first use $\lambda_1 = 10.0$ after the first LR decay and then apply $\lambda_2 = 60.0$ after the second LR decay to better maintain the flatness of adversarial loss landscape and control robust overfitting.

Table 6: Comparing model robustness w/ and w/o BoAT loss on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

Method	Natural			PGD-20			AutoAttack		
	best	final	diff	best	final	diff	best	final	diff
AT+WA	**83.50**	**84.94**	-1.44	55.05	47.60	7.45	49.89	43.83	6.06
ReBAT($d = 10$)	81.54	82.42	**-0.88**	**55.29**	**53.43**	**1.86**	**50.56**	**48.98**	**1.58**

C.2 Additional Results on the Effect of LR decay

Below we show that when a relatively large decay factor d is applied, *i.e.,* the model has overly strong fitting ability that results in robust overfitting, a large regularization coefficient λ should be choosen for better performance. Table 7 reveals this relationship between d and λ. When $d = 1.3$, even a λ as small as 1.0 will harm both the best and last robustness as well as natural accuracy, as $d = 1.3$ is already too small a decay factor that makes the model suffering from underfitting and naturally requiring no more flatness regularization. On the other side, when d is relatively large, even a strong regularization of $\lambda = 4.0$ is not adequate to fully suppress RO. Besides, comparing the situation of $\lambda > 0$ and $\lambda = 0$ for a fixed $d(\geq 1.5)$, we emphasize that the purposed BoAT loss again exhibits its apparent effectiveness in simultaneously boosting the best robustness and mitigating RO.

C.3 Additional Results on Adopting Stronger Training Time Attacker

In Section 4.3, we design two versions of ReBAT, and we name the stronger one that also uses a stronger training attacker as ReBAT[strong]. Here we fix the hyperparameter used in ReBAT above (note that we use a larger LR decay factor $d = 1.7$ in ReBAT[strong] for CIFAR-10, and now we still use $d = 1.5$ as in ReBAT) and adjust the attacker strength to study its influence when combined with ReBAT. According to Figure 14, though a slightly stronger attack (*e.g.,* $\varepsilon = 9/255$) may marginally improves the best and last robust accuracy, it heavily degrades natural accuracy, particularly when much stronger attack is used. We deem that this is because it breaks the balance from the other side

Table 7: Changing decay factor d and regularization strength λ and evaluating model robustness on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

Method	Natural			PGD-20			AutoAttack		
	best	final	diff	best	final	diff	best	final	diff
ReBAT($d = 1.3, \lambda = 0.0$)	**81.17**	**81.27**	-0.10	**56.43**	**56.23**	0.20	**50.80**	**50.75**	0.05
ReBAT($d = 1.3, \lambda = 1.0$)	80.67	80.63	0.04	56.27	56.15	0.12	50.74	50.65	0.09
ReBAT($d = 1.3, \lambda = 4.0$)	78.50	78.36	**0.14**	55.10	55.04	**0.06**	50.31	50.30	**0.01**
ReBAT($d = 1.5, \lambda = 0.0$)	**81.90**	**82.39**	-0.49	56.21	55.95	0.26	50.81	50.58	0.23
ReBAT($d = 1.5, \lambda = 1.0$)	81.86	81.91	**-0.05**	**56.36**	**56.12**	**0.24**	**51.13**	**51.22**	-0.09
ReBAT($d = 1.5, \lambda = 4.0$)	79.68	79.88	-0.20	55.72	55.65	0.07	50.52	50.50	**0.02**
ReBAT($d = 4.0, \lambda = 0.0$)	**83.05**	**85.38**	**-2.33**	55.98	50.87	5.11	50.38	46.95	3.43
ReBAT($d = 4.0, \lambda = 1.0$)	82.46	84.84	-2.38	55.86	52.02	3.84	50.87	47.88	2.99
ReBAT($d = 4.0, \lambda = 4.0$)	80.99	84.07	-3.08	**56.06**	**53.58**	**2.48**	**51.00**	**49.02**	**1.98**

that the overly strong attacker $\mathcal{A}$ dominates the adversarial game and results in an underfitting state that harms both robust and natural accuracy.

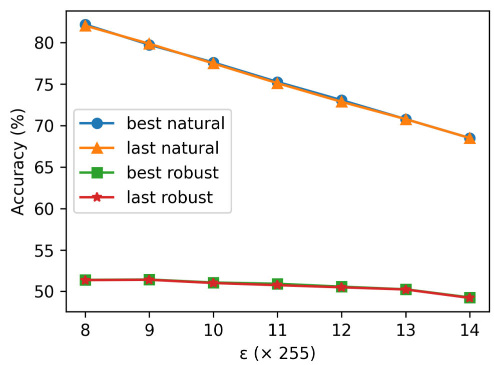


Figure 14: Using different adversarial attack strength in ReBAT.

C.4 Further Improving Natural Accuracy with Knowledge Distillation

Chen et al. [4] propose to adopt knowledge distillation (KD) [13] to mitigate RO and it is worth mentioning that their method achieves relatively good natural accuracy according to Table 1 and 2. Since our ReBAT method is orthogonal to KD, we propose to combine our techniques with KD to further improve natural accuracy. Specifically, we simplify their method by using only a non-robust standard classifier as a teacher (ST teacher) instead of using both a ST teacher and a AT teacher, because i) a large sum of computational cost for training the AT teacher will be saved, ii) our main goal is to improve natural accuracy so the ST teacher matters more and iii) ReBAT already use the WA model as a very good teacher. This gives the final loss function as

$$\ell_{\mathrm{ReBAT+KD}}(x, y; \theta) = (1 - \lambda_{\mathrm{ST}}) \cdot \ell_{\mathrm{BoAT}}(x, y; \theta) + \lambda_{\mathrm{ST}} \cdot \mathrm{KL}\left(f_\theta(x) \| f_{\mathrm{ST}}(x)\right), \tag{4}$$

where f_{ST} indicates the ST teacher and λ_{ST} is a trade-off parameter.

Table 8: Combining our methods with knowledge distillation and evaluating model robustness on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on the PreActResNet-18 architecture.

Method	Natural			PGD-20			AutoAttack		
	best	final	diff	best	final	diff	best	final	diff
ReBAT ($\lambda_{\mathrm{ST}} = 0.0$)	81.86	81.91	**-0.05**	**56.36**	**56.12**	0.24	**51.13**	**51.22**	-0.09
ReBAT+KD ($\lambda_{\mathrm{ST}} = 0.4$)	83.59	83.64	**-0.05**	54.91	54.77	-0.14	50.70	50.99	**-0.29**
ReBAT+KD ($\lambda_{\mathrm{ST}} = 0.5$)	84.12	84.20	-0.08	55.28	55.39	-0.11	50.47	50.72	-0.25
ReBAT+KD ($\lambda_{\mathrm{ST}} = 0.6$)	**84.34**	**84.72**	-0.38	53.83	54.30	**-0.47**	50.15	50.37	-0.22

Table 8 compares the performance of ReBAT+KD when different λ_{ST} is applied, and clearly a large λ_{ST} results in improvement in natural accuracy and decreases robustness (may due to the theoretically principled trade-off between natural accuracy and robustness [37]). However, it is still noteworthy that when $\lambda_{\mathrm{ST}} = 0.4$, a notable increase in natural accuracy ($\sim 1.7\%$) is achieved at the cost of only a small slide of $\sim 0.2\%$ in final robustness against AA. Also when $\lambda_{\mathrm{ST}} = 0.5$, it achieves comparable natural accuracy with KD+SWA [4] but has much higher AA robustness. Moreover, we emphasize that RO is almost completely eliminated regardless of the trade-off, which is the main concern of this paper and demonstrates the superiority of our method against previous ones. An intriguing phenomenon is that nearly all the final results are better than the results on the best checkpoints selected by the validation set, which implies that in this training scheme AT enjoys the same property of "training longer, generalize better" as ST without any need of early stopping.

C.5 The Effect of Different Learning Rate Schedules

In the previous experiments we only investigate the piecewise LR decay schedule. However, a natural idea would be using mild LR decay schedules, *e.g.,* Cosine and Linear decay schedule, instead of suddenly decaying it by a factor of d at some epoch in the piecewise decay schedule. As mentioned in Section 5, previous works have shown that changing LR decay schedule fails to effectively suppress RO whether with [31] or without WA [26] because the LR finally becomes small and endows the trainer with overly strong fitting ability. Therefore, here we continue to experiment with modified Cosine and Linear decay schedules that follow a similar LR scale of the piecewise LR decay schedule, and summarise the results in Table 9. To be specific, the LR still decays to 0.01 at epoch 150 and 0.001 at epoch 200, though the two decay stages (from epoch 100 to 150 and 150 to 200) are designed to be gradual following the Cosine/Linear schedule. We also gradually increase the strength of ReBAT regularization from zero as the LR gradually decreases, following the "larger decay factor goes with stronger ReBAT regularization" principle that we introduced in Appendix C.2.

Table 9: Comparing our method with WA on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on PreActResNet-18 architecture, when Cosine/Linear LR decay schedule are applied.

Method	Cosine						Linear					
	Natural		PGD-20		AutoAttack		Natural		PGD-20		AutoAttack	
	best	final	best	final	best	final	best	final	best	final	best	final
WA(d=10)	**82.32**	**84.99**	55.97	47.84	50.59	44.08	**82.21**	**85.01**	56.21	48.10	50.67	44.63
ReBAT(d=10)	82.06	82.22	**56.12**	**54.44**	**50.98**	**49.81**	81.98	82.13	**56.33**	**54.50**	**51.08**	**49.63**

It can be concluded from Table 9 that simply changing the LR decay schedule indeed improves the best robust accuracy from 49.89% to 50.59% and 50.67% against AA respectively, but it provides no help at all in mitigating RO as the final robust accuracy is still below 45% against AA. We also note that in this situation, the application of BoAT loss not only significantly mitigates RO but also further improves the best model robustness, which also proves its effectiveness.

C.6 Results on Different Network Architectures

In previous experiments we compare methods based on PreActResNet-18 and WideResNet-34-10 architecture, and here we also adopt VGG-16 [28] and MobileNetV2 [27] architecture. The significant improvement against baseline PGD-AT in both best and final robust accuracy and in mitigating RO reported in Table 10 further demonstrates that our method works on a wide range of network architectures.

Table 10: Comparing our method with PGD-AT on CIFAR-10 under the perturbation norm $\varepsilon_\infty = 8/255$ based on VGG-16 and MobileNetV2 architecture.

Method	VGG-16						MobileNetV2					
	Natural		PGD-20		AutoAttack		Natural		PGD-20		AutoAttack	
	best	final	best	final	best	final	best	final	best	final	best	final
PGD-AT	**78.43**	**81.64**	50.56	44.25	44.19	39.66	**79.85**	**80.67**	51.56	50.67	46.01	45.27
ReBAT	78.17	78.37	**53.13**	**53.01**	**47.24**	**47.13**	78.98	80.81	**53.18**	**52.57**	**47.66**	**47.35**