RIDE: A LIGHTWEIGHT PLUG-AND-PLAY FRAMEWORK FOR ENHANCING ADVERSARIAL ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) remain vulnerable to adversarial attacks based on deliberately crafted perturbations. Previous arts target specific adversarial attacks (e.g., adversarial training) or impose substantial computational overhead on inference time (e.g., adversarial purification). Moreover, both of them significantly compromise the accuracy on clean examples, which dominate real-world tasks, thereby marginalizing their practical applications. To address these challenges, we propose a lightweight plug-and-play framework, namely RIDE, to protect DNNs in inference time. By combining detection and purification, RIDE precisely identifies and purifies adversarial examples, enhancing model robustness while maintaining comparable accuracy on clean examples. In particular, RIDE leverages information from the low-variance subspace (a subspace that exhibits severe selection bias) to precisely detect adversarial examples, which have been demonstrated to diverge from clean examples in this subspace. Moreover, the samples identified as adversarial are purified through an adversarial filter, which uses a denoising autoencoder to optimize the input toward regions of higher training data coverage, driving adversarial examples back to the clean data manifold. Finally, RIDE applies inference-time data augmentation to reduce noise in the predictions, further enhancing the robustness of this framework. Extensive experiments demonstrate that RIDE increases robust accuracy at most by 50.97% while maintaining comparable accuracy on clean examples. Moreover, RIDE uses only 0.87% of the inference time compared to that of classical inference-time defense methods.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable success in various fields, including recommendation system, intrusion detection, medical analysis, etc. However, despite their extraordinary real-world performance, DNNs remain vulnerable to adversarial examples, which are elaborately crafted by adding imperceptible perturbations to clean examples, causing DNNs to make mistakes in practical tasks Szegedy et al. (2014); Goodfellow et al. (2015); Wang et al. (2025).

To enhance the robustness of DNNs, researchers use adversarial training to find the saddle point, rather than the valley floor, on the training surface Madry et al. (2018); Zhang et al. (2019); Tsipras et al. (2019b); Kireev et al. (2023), which shatters the gradients against specific adversarial attacks, hindering attackers from crafting adversarial examples through gradient ascent Athalye et al. (2018). However, adversarial training techniques aim to defend against specific adversarial attacks and require retraining the target model to enhance its robustness, making them computationally inefficient. To overcome this shortage, inference-time defense methods have become a promising direction for improving adversarial robustness Samangouei et al. (2018); Nie et al. (2022); Xie et al. (2018). In particular, these methods incorporate an additional model (e.g., a diffusion model Song et al. (2024); Lei et al. (2025)) to purify adversarial examples, which aims to eliminate adversarial perturbations from harmful examples. Although these methods can be applied to various models without retraining, they still incur substantial computation overhead during purification Lei et al. (2025), making them less efficient in practical scenarios. More critically, existing methods enhance robustness at the cost of compromising accuracy on clean examples (which dominate real-world tasks), thereby marginalizing their practical applications. In a nutshell, simultaneously achieving accuracy, efficiency, and strong robustness is still an open problem in defending against adversarial examples.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To bridge this gap, we propose RIDE (**R**esidual-aware **I**nference-time **De**fender), a Lightweight Plug-and-Play framework, to accurately detect and efficiently purify adversarial examples, thereby enhancing the robustness of target models while maintaining their accuracy on clean examples. Specifically, RIDE uses three components to protect target models: a detector, an adversarial filter, and a prediction enhancement module. First, as adversarial attackers have been shown to exploit selection bias to launch their attacks Schmidt et al. (2018); Tsipras et al. (2019a); Tan et al. (2026), we demonstrate that selection bias is more severe in the residual subspace (a subspace where examples exhibit low variance), where adversarial examples diverge from clean examples (Thm. 1). Therefore, the detector leverages these distinctions to accurately detect adversarial examples, minimizing the impact on the accuracy of clean examples. Moreover, examples identified as adversarial are purified by an adversarial filter, which uses a denoising autoencoder (DAE) to optimize adversarial examples toward regions of higher training data coverage (Thm. 2), driving them back to the clean data manifold. Finally, RIDE applies data augmentation to clean examples and purified adversarial examples, averaging predictions over randomly processed inputs to reduce prediction noise Xie et al. (2018). *As a result, once these plug-and-play components are integrated into target models, the equipped models become strongly robust without sacrificing accuracy on clean examples.*

Extensive experiments demonstrate that RIDE effectively improves the robustness of target models, increasing robust accuracy at most by 50.97% while maintaining competitive accuracy on clean examples. Moreover, compared to classical inference-time defense method (e.g., DiffPure), RIDE uses only 0.87% of the inference time, highlighting its lightweight design.

In summary, our contributions are three-fold:

- We demonstrate that adversarial examples diverge from clean examples in a residual subspace (a subspace where examples exhibit low variance), which can be used to accurately detect adversarial examples.
- We propose RIDE, a lightweight plug-and-play framework for robustness enhancement. By combining adversarial detection, adversarial purification, and prediction enhancement, RIDE achieves robustness enhancement without sacrificing accuracy on clean examples.
- Extensive experiments demonstrate that RIDE improves robustness against various adversarial attacks while preserving accuracy on clean examples, demonstrating its effectiveness as a practical plug-and-play defense solution.

2 Related Work

Training-time defenses. Adversarial training is one of the prevalent approaches to improve the robustness of DNNs. In particular, PGD-based adversarial training formalizes robustness enhancement as a min-max optimization and increases robust accuracy against specific adversarial attacks Madry et al. (2018); Kireev et al. (2023). To further improve adversarial training techniques, lots of researchers attempt to balance robustness and accuracy, taking into account misclassified examples or weight perturbations Zhang et al. (2019); Wang et al. (2020); Wu & Zhu (2020). However, these methods still target specific adversarial attacks and require retraining the target model, which are computationally ineffective, thereby limiting their application to practical scenarios.

Inference-time defenses. Inference-time defenses aim to improve adversarial robustness without modifying the target classifier, making them suitable for already deployed models. Early research relied mainly on input transformations, statistical detection, or generative reconstruction Meng & Chen (2017); Xu et al. (2018); Samangouei et al. (2018); Song et al. (2018); Xie et al. (2018); Guo et al. (2018). In recent years, researchers have shifted toward adversarial purification at test time. DiffPure Nie et al. (2022) introduced diffusion-based purification by adding noise to adversarial inputs and recovering them through iterative denoising. Follow-up works have further improved the purification process through test-time purification Tang & Zhang (2024) or diffusion-based generation Song et al. (2024). However, generative purification remains computationally expensive, motivating recent efforts toward more efficient purification Lei et al. (2025). Although these methods enhance the robustness of target models, their purification capabilities still rely on pretrained models and specific distillation strategies, making them ineffective in practical applications.

Another concern is the distortion of clean examples during purification. Recent studies show that uniform noise injection compromises clean data Pei et al. (2025), causing the recovered output to

deviate from the original and resulting in the loss of accuracy Park et al. (2025). In this work, we propose RIDE, a lightweight plug-and-play framework. Specifically, RIDE consists of residual-aware detection, DAE based purification, and prediction enhancement. These approaches incur modest inference overhead, introduce no additional noise, and reduce unnecessary modifications to clean samples.

3 Residual Subspace Is Exploited to Launch Adversarial Attacks

The key component in RIDE is the detector, which enables RIDE to accurately distinguish adversarial examples from clean examples, reducing unnecessary processing of clean examples. This observation motivates us to investigate the differences between adversarial and clean examples.

Previous work demonstrates that attackers exploit selection bias (i.e., the absence of certain regions of training data) to launch adversarial attacks, which implies that adversarial examples are perturbed to fall into regions uncovered by training data Schmidt et al. (2018); Tsotsos & Luo (2021); Tan et al. (2026). However, in a high-dimensional space, the coverage of training data is not evenly distributed across dimensions, which motivates us to prove the following theorem.

Theorem 1. *Denote $\{\lambda_i\}_{i=1}^d$ and $\{\boldsymbol{\mu}_i\}_{i=1}^d$ as the eigenvalues and corresponding eigen vectors of Σ, the covariance matrix of $\boldsymbol{X}$. For a subspace $\mathcal{S} = \mathrm{span}\{\boldsymbol{\mu}_k, \cdots, \boldsymbol{\mu}_d\}$, we have*

$$P(\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\| \geq \epsilon) \leq \frac{\sum_{i=k}^{d} \lambda_i}{\epsilon^2}, \tag{1}$$

where $K_{\mathcal{S}} = (\boldsymbol{\mu}_k^{\mathrm{T}}, \cdots, \boldsymbol{\mu}_d^{\mathrm{T}})$ projects vectors to $\mathcal{S}$.

Proof. Due to space constraints, the proof is deferred to Appendix A. □

Thm. 1 implies that the data coverage Tan et al. (2026) is decided by the eigenvalues. In other words, the subspace with small eigenvalues, where the samples are concentrated around the expectations, exhibits more severe selection bias, making it easier for attackers to exploit.

Based on Thm. 1, we propose to detect adversarial examples in the *residual subspace*, where the summation of corresponding eigenvalues are small enough. Denote the basis of this subspace as $K_{\mathcal{S}}$, which is a subset of the eigenvectors of the covariance matrix, we use the projected norm of input samples as the indicator to identify adversarial examples, that is,

$$S_{mean} = \frac{1}{d}\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|_1, \; S_{max} = \|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|_\infty, \tag{2}$$

where d represents the dimension of the residual subspace.

As indicated in Eq. (1), samples diverging from the expectation in residual subspace are small probability events, which motivates us to use hypothesis testing Wasserman (2013), a prevalent method for statistical decision making, to detect adversarial examples. Simply put, large deviations in the residual subspace indicate that the corresponding examples are adversarial with high probability.

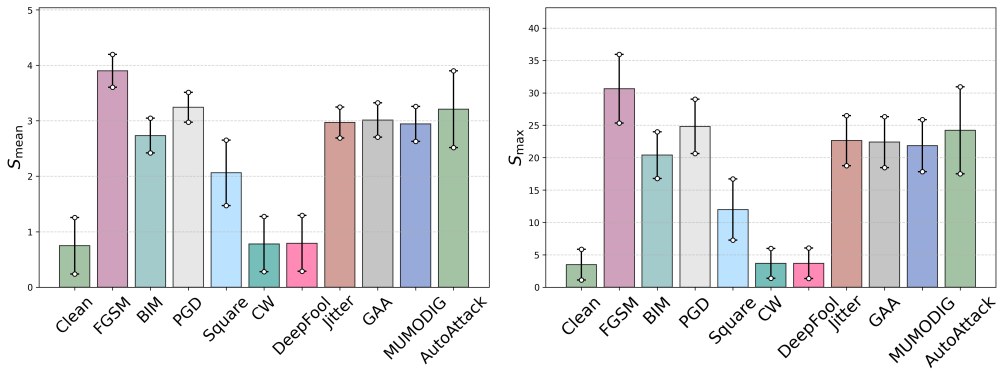

Figure 1: The indicator for clean and adversarial examples in the residual subspace.

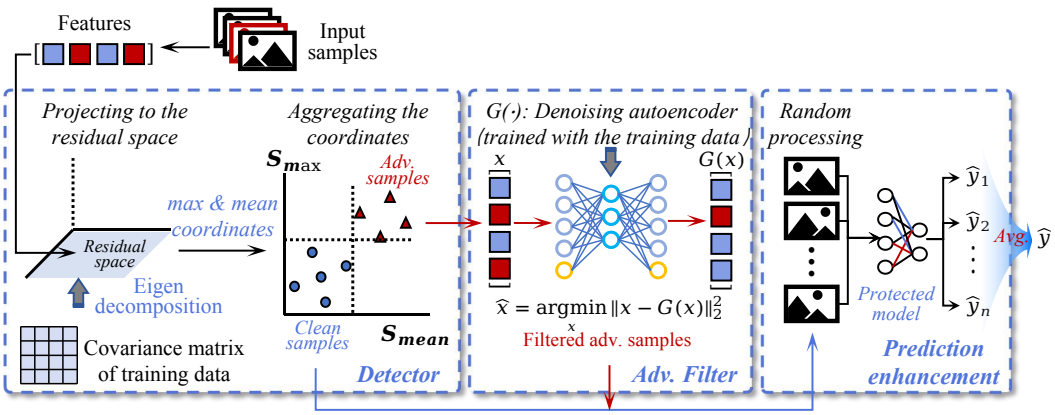

Figure 2: Overview of RIDE: the detector projects inputs samples into a residual subspace, where the training samples exhibit low variance. Samples that are identified as adversarial are filtered by minimizing the reconstruction error of the optimal DAE. The final output is obtained through averaging the predictions over randomly augmented clean samples or filtered adversarial samples, which significantly reduces prediction noise.

To validate this theorem, we display S_{mean} and S_{max} for clean examples and adversarial examples generated by various attacks in Fig. 1. The results show that adversarial examples generated by most attacks exhibit larger values of these indicators in the residual subspace, supporting our conclusion that S_{mean} and S_{max} are good indicators to distinguish adversarial examples from clean ones. Although adversarial examples generated by the C&W and DeepFool attacks exhibit values similar to those of clean examples, which are difficult to identify, motivating us to design a prediction enhancement module to address these attacks Xie et al. (2018).

4 RIDE: Residual-aware Inference-time Defender

Based on analysis in Sec. 3, we propose a lightweight plug-and-play defense framework, namely RIDE, to achieve inference-time defense without retraining the target model. As illustrated in Fig. 2, RIDE consists of three components: a residual-aware detector, a DAE based adversarial filter, and a module of prediction enhancement. By combining detection and adversarial purification, RIDE achieves high accuracy on clean examples, as well as strong robustness against adversarial examples.

4.1 Design of Detector

As demonstrated in Sec. 3, adversarial examples are different from clean examples in the residual subspace, which can be used to accurately detect adversarial examples. Therefore, we use the deviation in the residual subspace as an indicator in the detector.

Specifically, the key step in projecting the input data into the residual space is to determine the projection matrix, i.e., $K_{\mathcal{S}}$, and the expectation $\mathbb{E}[\boldsymbol{X}]$. According to Thm. 1, $K_{\mathcal{S}}$ refers to a submatrix of the eigenvector matrix of Σ, the covariance matrix of the input data. In practice, this matrix is represented by the covariance matrix of the training data. The eigenvector matrix is obtained by applying principal component analysis (PCA) Strang (2022) on Σ. The eigenvectors corresponding to the k smallest eigenvalues are then selected to form $K_{\mathcal{S}}$. Moreover, $\mathbb{E}[\boldsymbol{X}]$ is obtained by taking expectation over the training data.

$K_{\mathcal{S}}$ is used to project samples into the residual subspace. That is, for an input data $\hat{\boldsymbol{x}}$, its residual representation is calculated as

$$\boldsymbol{r} = K_{\mathcal{S}}(\hat{\boldsymbol{x}} - \mathbb{E}[\boldsymbol{X}]).$$

Follow the definition in Sec. 3, $\frac{1}{d}\|\boldsymbol{r}\|_1$ represents the deviation of $\hat{\boldsymbol{x}}$ in the residual subspace, thus can be used as an indicator to identify adversarial examples. A large value of $\frac{1}{d}\|\boldsymbol{r}\|_1$ indicates that $\hat{\boldsymbol{x}}$ deviates from the clean data manifold and is therefore identified as adversarial. Finally, we use the 95% quantile of $\frac{1}{d}\|\boldsymbol{r}\|_1$ over the training data as the threshold to detect adversarial examples. Since

the indicator exceeding this threshold is a low-probability event under clean distributions ($\leq 5\%$), its occurrence reveals an abnormality in the input data, which is the idea of hypothesis testing.

4.2 Adversarial Filter

The detector aims to accurately distinguish adversarial examples from clean examples, thereby minimizing the impact on the accuracy of clean examples. The examples identified as adversarial are then purified by an adversarial filter, which removes the harmful perturbations from these examples.

In particular, the target of adversarial filter is eliminating harmful perturbations to drive adversarial examples back to the clean data manifold. Building on the fact that attackers exploit selection bias to launch adversarial attacks by perturbing input data into the uncovered regions of the training data Schmidt et al. (2018); Tsotsos & Luo (2021); Tan et al. (2026), we formalize adversarial example purification as a problem of optimizing input data toward a nearby region of higher data coverage, where training data are more concentrated (i.e., $p(\boldsymbol{x})$ is larger).

As the analytical formulation of $p(\boldsymbol{x})$ is not available in practice, we cannot directly optimize it to purify adversarial examples. To overcome this difficulty, we demonstrate the following theorem.

Theorem 2. *Denote the optimal DAE trained with $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\xi}$ as $\mathcal{D}(\cdot)$, where $\tilde{\boldsymbol{x}} \sim p(\boldsymbol{x})$ and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathrm{I})$. If $\log p(\boldsymbol{x})$ is L-smooth and $\sigma^2 \leq \frac{2}{L}$, we have:*

$$\boldsymbol{x}_{n+1} = \mathcal{D}(\boldsymbol{x}_n) \Rightarrow p(\boldsymbol{x}_{n+1}) \geq p(\boldsymbol{x}_n). \tag{3}$$

Proof. Due to space constraints, the proof is deferred to Appendix B. □

Thm. 2 demonstrates that, if the training data distribution is smooth (i.e., it avoids extremely irregular distributions), using $\boldsymbol{x}_{n+1} = \mathcal{D}(\boldsymbol{x}_n)$ instead to optimize the input data drives the data toward the region of higher data coverage, which is the target of the adversarial filter.

Therefore, RIDE trains a DAE on the training data to perform purification within the adversarial filter (as shown in Fig. 2). Moreover, the iterated process in Eq. (3) is unstable because it is almost impossible for all the eigenvalues of the Jacobian matrix of the trained DAE (i.e., $\mathcal{D}(\cdot)$) to be less than 1. Hence, we use the following optimization process instead to remove harmful perturbations.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\mathcal{D}(\boldsymbol{x}) - \boldsymbol{x}\|_2^2. \tag{4}$$

Similar to Eq. (3), this optimization process converges to the nearby fixed-point of $\mathcal{D}(\cdot)$, thus can be used to purify adversarial examples.

4.3 Prediction Enhancement Module

Finally, as deliberately crafted perturbations, which are highly aligned with the local geometry of the decision boundary, are demonstrated to be sensitive to small variations Ilyas et al. (2019), RIDE incorporates a prediction enhancement module to further reduce noise in the model prediction.

In a nutshell, RIDE randomly transforms an input data for N times, and then takes expectation over the predictions of these randomly transformed samples. Specifically, given an input data $\boldsymbol{x}$ ($\boldsymbol{x}$ can be a clean example or a purified adversarial example), RIDE first randomly transforms it as

$$\tilde{\boldsymbol{x}}_i = T_i(\boldsymbol{x}), \quad i = 1, \ldots, N, \tag{5}$$

where $T_i(\cdot)$ represents the i-th random transformation. These random transformations involve flipping and cropping with padding, among others.

The final prediction of RIDE is then obtained by taking the expectation over the predictions of randomly transformed data, that is,

$$p(y|\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{F}(T_i(\boldsymbol{x}); \boldsymbol{W}), \tag{6}$$

where $\mathcal{F}(\cdot; \boldsymbol{W})$ denotes the target model to be protected.

This module reduces the influence of noise introduced by undetectable perturbations or the purification process, further enhancing the robustness of the target model.

5 EXPERIMENTS

In this section, we aim to validate the design of RIDE and demonstrate that the proposed framework is lightweight and effective in enhancing the robustness of DNN models. All experiments are performed on a server equipped with two Intel(R) Xeon(R) Gold 6430 CPUs (2 × 32 cores), 1 TB memory, and eight NVIDIA GeForce RTX 4090 24GB GPUs. The server runs Ubuntu 24.04.1 LTS, with Python 3.10.20, PyTorch 2.5.1+cu121, and CUDA 12.1.

5.1 EXPERIMENTAL SETUP

Datasets and Models. We evaluate RIDE on four classical datasets: CIFAR-10, CIFAR-100 Krizhevsky (2009), SVHN Netzer et al. (2011), and PathMNIST Yang et al. (2023). In addition, both standard and adversarially trained ResNet-18 He et al. (2016) are used in our evaluation to demonstrate RIDE can further improve the robustness of target models. Specifically, the standard model are trained for 100 epochs using SGD with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 5×10^{-4}. The learning rate is scaled by a factor of 0.1 at epochs 75 and 90.

Adversarial Attacks. Various adversarial attacks are employed to comprehensively demonstrate the effectiveness of RIDE, including FGSM Goodfellow et al. (2015), DeepFool Moosavi-Dezfooli et al. (2016), BIM Kurakin et al. (2017), C&W Carlini & Wagner (2017), PGD Madry et al. (2018), Square Attack Andriushchenko et al. (2020), AutoAttack (AA) Croce & Hein (2020), Jitter Schwinn et al. (2023), GAA Gan & Wo (2025), and MUMODIG Ren et al. (2025). In particular, for ℓ_∞-bounded attacks, the perturbation budget is set to $\epsilon = 8/255$. DeepFool uses 50 steps with an overshoot of 0.02. Square Attack uses 5000 queries. BIM uses $\alpha = 2/255$, and 10 iterations, while PGD uses the same perturbation budget and step size with random initialization. C&W uses $c = 0.01$, $\kappa = 0$, 1000 optimization steps, and a learning rate of 0.01. AutoAttack adopts its standard configuration under the ℓ_∞ norm with $\epsilon = 8/255$. Moreover, for GAA and MUMODIG, WRN-28-10 is used as the surrogate model to attack ResNet-18 models. GAA is configured with 10 iterations, $N = 20$, and $\rho = 0.8/255$, while MUMODIG uses 10 iterations, $N_{\text{trans}} = 6$, $N_{\text{base}} = 1$, $N_{\text{interpolate}} = 1$, $\text{region}_{\text{num}} = 2$, and $\lambda = 0.65$.

Adversarial Training Baselines. We compare RIDE with PGD-AT Madry et al. (2018), TRADES Zhang et al. (2019), MART Wang et al. (2020), and AWP Wu et al. (2020). Specifically, TRADES uses $\epsilon = 8/255$, $\alpha = 2/255$, 10 attack steps, and $\beta = 3.0$. TRADES+AWP uses $\gamma = 0.005$. MART follows the configuration of its original paper. Unless otherwise specified, adversarially trained models use the same configuration as the standard-trained models.

Inference Defense Baselines and RIDE Configuration. We further compare RIDE with MagNet Meng & Chen (2017) and DiffPure Nie et al. (2022). In particular, DiffPure uses its default diffusion purification configuration with $t = 100$. For RIDE, the dimension of the residual subspace is set to 972. The denoising autoencoder is trained for 150 epochs, using a Gaussian noise with $\sigma = 0.1$. During inference, the optimization process performs 100 steps with a learning rate of 0.01. In the prediction enhancement module, the randomized transformations include random horizontal flipping and random cropping with a padding of 4. The expectation is to computed over 10 randomly transformed data.

5.2 COMPARISONS WITH EXISTING DEFENSES

We compare RIDE with representative adversarial training methods, including PGD-AT Madry et al. (2018), TRADES Zhang et al. (2019), MART Wang et al. (2020), and AWP Wu et al. (2020), as well as inference-time purification methods MagNet Meng & Chen (2017) and DiffPure Nie et al. (2022). To better simulate realistic deployment scenarios, where adversarial examples usually constitute only a small portion of incoming data, we construct mixed evaluation sets by combining 90% clean samples with 10% adversarial samples for each attack type. The main results on CIFAR-10 are presented in Table 1.

As shown in Table 1, RIDE consistently maintains high accuracy under the practical evaluation settings. Compared to adversarial training methods, RIDE achieves consistently higher accuracy in all attack scenarios, increasing the accuracy by 14.30% on average. In particular, RIDE simultane-

Table 1: Comparisons on CIFAR-10 under a 10% adversarial samples ratio.

Method	Clean	FGSM	BIM	PGD	Square	C&W	DeepFool	Jitter	GAA	MUMODIG	AA
Without Defense	**94.13**	86.36	84.66	84.66	84.68	84.83	85.55	84.77	85.15	86.97	84.66
PGD-AT	84.85	82.35	81.44	81.52	81.98	84.53	76.99	82.91	83.13	83.39	81.07
TRADES	86.56	83.23	82.51	82.57	83.07	86.18	78.57	84.18	84.56	84.99	82.23
MART	79.14	77.04	76.74	76.75	76.66	78.80	71.88	77.76	77.60	77.89	76.20
TRADES+AWP	85.19	82.22	81.63	81.68	82.06	84.84	77.14	82.86	83.50	83.84	81.24
MagNet	75.46	73.44	73.70	73.82	75.46	75.36	75.20	74.40	74.30	74.07	74.40
DiffPure	88.84	88.53	88.27	88.59	88.85	89.13	88.80	88.85	88.57	88.69	89.28
RIDE (Ours)	93.85	**92.30**	**91.62**	**91.93**	**92.91**	**93.14**	**93.70**	**92.67**	**92.30**	**91.64**	**92.37**

Table 2: Plugging RIDE into existing defenses on CIFAR-10 (100% adversarial sample ratio).

Method	Clean	FGSM	BIM	PGD	Square	C&W	DeepFool	Jitter	GAA	MUMODIG	AA
Without Defense	**94.13**	16.46	0.00	0.00	0.72	2.09	10.27	0.76	4.85	22.91	0.00
Without Defense+RIDE	93.85	**77.45**	**71.16**	**75.06**	**84.67**	**87.38**	**89.76**	**82.10**	**77.47**	**70.75**	**79.72**
PGD-AT	84.85	57.67	50.19	50.78	54.07	81.13	7.34	64.40	67.20	70.31	46.31
PGD-AT+RIDE	**84.91**	**65.03**	**61.69**	**61.90**	**78.31**	**84.41**	**73.87**	**70.42**	**67.92**	**71.42**	**63.78**
TRADES	86.56	53.84	46.07	46.54	51.61	82.39	7.34	61.94	66.32	70.36	43.62
TRADES+RIDE	**86.57**	**61.32**	**57.18**	**57.91**	**79.11**	**86.23**	**76.18**	**66.89**	**67.10**	**71.28**	**59.75**
MART	79.14	57.98	54.74	54.98	53.82	75.53	6.23	65.59	65.25	66.86	48.53
MART+RIDE	**79.22**	**61.75**	**60.50**	**60.86**	**73.02**	**78.71**	**70.51**	**68.22**	**65.74**	**67.24**	**61.83**
TRADES+AWP	85.19	54.26	48.37	48.83	51.63	80.99	5.53	61.04	67.24	69.54	44.91
TRADES+AWP+RIDE	**85.40**	**58.47**	**54.95**	**55.28**	**76.26**	**84.50**	**70.86**	**64.46**	**67.25**	**69.70**	**57.14**
DiffPure	88.91	85.48	86.01	86.04	86.81	89.20	88.65	87.65	85.21	85.39	86.83
DiffPure+RIDE	**89.41**	**86.17**	**87.05**	**86.87**	**87.35**	**89.38**	**88.84**	**88.17**	**85.67**	**85.40**	**87.47**

ously preserves better accuracy on clean examples. Its performance is superior to diffusion-based purification methods, particularly for attacks that introduce significant deviations in the input data.

We further investigate whether RIDE can improve classical adversarial defenses by plugging it into these defense methods, the results are displayed in Table 2. These results indicate that integrating RIDE with classical defense methods consistently improves their robustness, increasing the robust accuracy by 92.14% on average. Furthermore, the experiments demonstrate that RIDE serves as a plug-and-play solution for existing defense methods, further enhancing their robustness.

We further evaluate RIDE on CIFAR-100, SVHN, and PathMNIST to comprehensively assess its performance. As shown in Table 3, RIDE consistently outperforms traditional methods, increasing the robust accuracy at most by 50.97%. The results of GAA and MUMODIG attacks further demonstrate the effectiveness of RIDE under transferable adversarial attacks.

5.3 Ablation Study

First, we evaluate the impact of the detection module under different adversarial sample ratios. As shown in Fig. 3, the defense with detection consistently achieves higher accuracy than the corresponding defense without detection under all adversarial ratios. At a low adversarial ratio of 1%, the defense with detection achieves a clear advantage because the detector can accurately distinguish adversarial examples from clean examples, avoiding unnecessary purification. In contrast, the defense without detection applies the purification process to all inputs, introducing extra distortion to clean samples. As the adversarial sample ratio increases to 50%, the accuracy of all methods decreases due to the increasing difficulty caused by more adversarial inputs. However, the detection-based defense still outperforms its counterpart without detection, but the gap between them becomes smaller. This phenomenon is reasonable, since the proportion of samples that should not be purified decreases as the ratio of adversarial examples increases. These results also demonstrate that the detection module effectively protects clean samples from unnecessary processing.

Moreover, Fig. 5 further illustrates the performance under different adversarial sample ratios. As the adversarial ratio increases, the model without defense suffers a substantial accuracy degradation, whereas all models with defenses exhibit strong robustness under various attacks. In particular, RIDE framework with detection consistently outperforms other defense methods, especially under strong adversarial attacks (e.g., AutoAttack). These results demonstrate that the detection module is

Table 3: Comparisons across different datasets (%) (10% adversarial sample ratio).

Dataset	Method	Clean	FGSM	BIM	PGD	Square	C&W	DeepFool	Jitter	GAA	MUMODIG	AA
CIFAR-100	Without Defense	74.11	67.36	66.47	66.47	66.51	66.86	67.65	66.80	67.04	67.26	66.47
	PGD-AT	60.44	57.55	57.08	57.07	57.42	59.90	55.09	58.85	58.61	58.81	56.77
	TRADES	57.13	54.05	53.70	53.71	53.93	56.51	51.97	55.58	55.49	55.43	53.42
	MART	53.58	51.26	50.98	51.02	51.03	53.22	48.72	52.22	52.35	52.39	50.48
	TRADES+AWP	55.99	53.20	52.93	52.97	53.10	55.39	51.06	54.36	54.37	54.31	52.51
	RIDE (Ours)	**74.16**	**71.22**	**70.66**	**70.94**	**72.37**	**73.16**	**73.55**	**71.32**	**71.78**	**70.96**	**72.33**
SVHN	Without Defense	96.77	91.17	87.19	87.20	87.35	91.24	87.90	88.43	89.21	92.86	87.12
	PGD-AT	94.34	92.26	91.16	91.21	90.69	93.92	86.15	92.36	92.24	92.58	89.95
	TRADES	95.65	94.13	92.19	92.22	91.82	95.04	87.33	93.45	93.37	93.73	90.96
	MART	95.41	95.10	93.25	93.45	89.72	91.69	88.18	94.15	93.11	93.65	89.28
	TRADES+AWP	94.30	91.50	90.73	90.76	90.68	93.92	85.55	92.16	91.95	92.37	89.59
	RIDE (Ours)	**96.84**	**95.63**	**95.97**	**96.06**	**96.29**	**96.30**	**95.62**	**96.41**	**95.51**	**95.52**	**96.15**
PathMNIST	Without Defense	**90.21**	82.70	81.39	81.39	81.45	81.69	82.08	81.43	83.06	82.51	81.39
	PGD-AT	79.46	77.34	76.78	76.84	76.87	79.05	72.19	77.58	77.38	78.01	76.45
	TRADES	81.39	78.51	77.77	77.81	78.01	80.77	74.03	78.74	79.32	79.93	77.50
	MART	79.58	77.58	76.84	76.88	76.78	79.25	72.34	77.45	77.69	78.19	76.42
	TRADES+AWP	81.98	78.93	78.09	78.06	78.02	81.60	74.53	78.82	80.10	80.70	76.88
	RIDE (Ours)	89.96	**88.89**	**88.79**	**88.79**	**88.08**	**88.45**	**89.57**	**89.40**	**88.73**	**88.29**	**88.73**

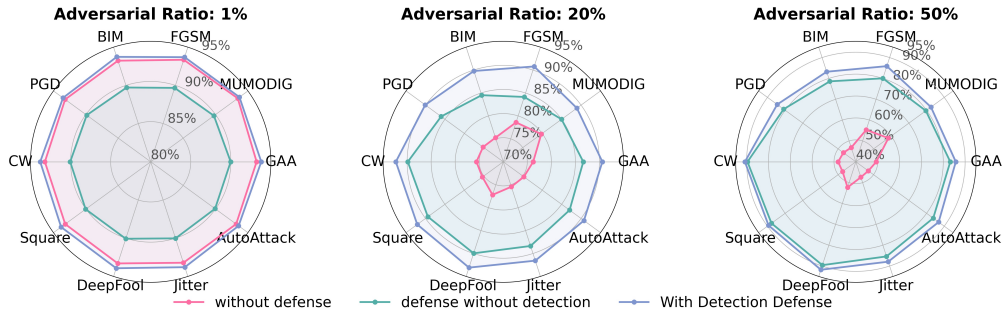

Figure 3: The necessity of the detector under different ratios of adversarial samples.

an essential part in RIDE, which can further improve robust accuracy compared to the counterpart without detection. Furthermore, under high adversarial sample ratios, the methods with and without detection both significantly outperform PGD-AT, indicating that the proposed inference-time defense can effectively enhance the robustness of the target model without retraining.

Finally, we conduct ablation studies on the defense components as well as their processing order. As shown in Fig. 4, the prediction enhancement (PE) module and the adversarial filter (AF) provide complementary benefits under various attacks, with robust accuracy increasing as each is added to the framework. Moreover, the PE module excels at purifying adversarial examples generated by C&W, DeepFool, and Square, while the adversarial filter provides stronger improvement for other attacks. By combining these modules in sequence, RIDE substantially enhances the robustness of target model in the practical scenarios.

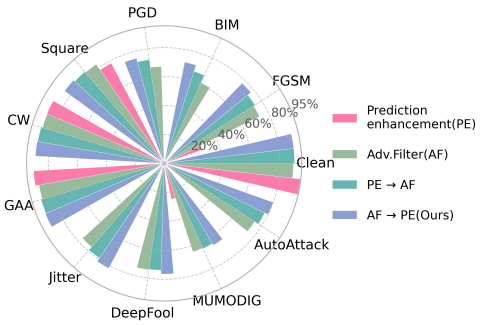

Figure 4: Ablation study of the adversarial filter and the prediction enhancement module.

Moreover, as indicated in Fig. 4, the proposed processing sequence, i.e., AF→PE, outperforms other counterparts across various attacks, which demonstrates the effectiveness of RIDE's design. That is, applying the AF module first removes the bulk of adversarial perturbations, after which PE module further suppresses the residual noise.

Defense Against Adaptive Attack. Following the criteria in Athalye et al. (2018), we use adaptive attacks, where all defense components are exposed as white-box to attackers, to further evaluate the

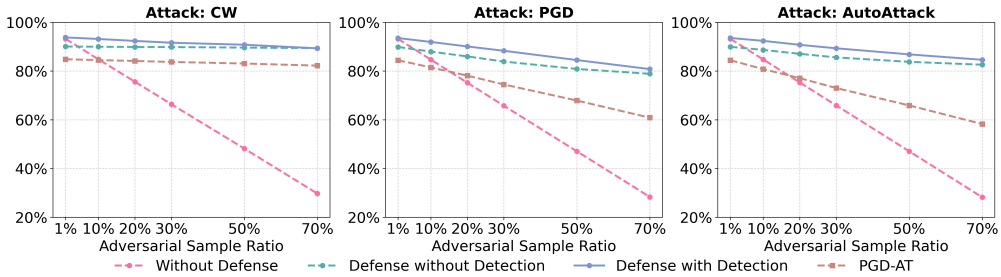

Figure 5: Performance under different ratios of adversarial samples.

Table 4: Comparison under adaptive attack (%).

Method	PGD+EOT	BPDA+EOT	Avg.
Without Defense	0.0	0.0	0.0
DiffPure	46.48	**76.37**	61.43
RIDE	**70.12**	69.92	**70.02**

Table 5: The Efficiency Comparison.

Method	Additional Storage (MB)	Inference Time (ms)
DiffPure	1628.22	168.7433
RIDE	**86.93**	**1.4664**

performance of inference-time defense. In particular, PGD+EOT and BPDA+EOT attacks are implemented using its standard code and corresponding experimental settings from Lee & Kim (2023). As shown in Table 4, the robust accuracy drops when all components are exposed as white-box to attackers. However, even though each defense method is vulnerable to specific adaptive attacks, RIDE achieves higher robust accuracy on average, increasing it by 13.98%, which demonstrates that RIDE is more robust to adaptive attacks in practice.

Inference Efficiency. We further compare the inference efficiency of RIDE with DiffPure on CIFAR-10. As shown in Table 5, RIDE requires 86.93 MB of additional storage, only 5.34% compared to the storage used in DiffPure. Meanwhile, RIDE achieves an average inference time of 1.4664 ms/sample, substantially reducing the inference cost compared to DiffPure, which requires 168.7433 ms/sample. These results demonstrate the lightweight inference property of RIDE.

Validating Residual-Subspace Detection. Finally, for the input data $\boldsymbol{x}$, we use the deviations in the residual subspace as a regularization term (i.e., $loss'(\boldsymbol{x}) = loss(\boldsymbol{x}) - \lambda \cdot \frac{1}{d}\|K_{\mathcal{S}}(\boldsymbol{x} - \mathbb{E}[\boldsymbol{X}])\|_1$, where $loss(\boldsymbol{x})$ represents the original attack loss) in attacks to adapt them for escaping the detection in RIDE. The results in Table 6 show that the accuracy of both RIDE and the undefended vanilla model increases with λ. These results indicate that it becomes difficult for adversaries to find valid adversarial examples that simultaneously fool the model and maintain low deviation within the residual subspace. This phenomenon further validates our theorem that adversarial examples exhibit significantly larger deviations than clean examples in the residual subspace, which RIDE exploits for detection. Moreover, RIDE consistently achieves higher accuracy, demonstrating that the other components are effective in defending against the identified adversarial examples.

Table 6: Detection indicator regularization using BPDA+EOT.

Defense	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 5$
w/o Def.	0.	38.8	52.34	57.42	68.36
RIDE	69.92	72.27	76.56	78.52	83.40

6 Conclusion

In this work, we propose a lightweight plug-and-play defense framework, namely RIDE. By combining residual-aware adversarial example detection, DAE based adversarial purification, and prediction enhancement, RIDE achieves efficient inference-time robustness enhancement while maintaining competitive accuracy on clean example, significantly improving its usability in real-world scenarios. In particular, RIDE can be plugged into various defense frameworks without retraining the target model to further enhance their robustness, which demonstrates that RIDE is a broadly applicable plug-and-play solution for robustness enhancement.

7 LLM USAGE

This paper uses a large language model (LLM) to assist with manuscript writing, language polishing, which improves the clarity and readability of the text.

The LLM was not involved in the formulation of the research problem, the design of RIDE, the experimental design, and the analysis of experimental results. All research ideas, methodological designs, experiments, and scientific conclusions were developed and conducted by the authors. The use of the LLM was limited to the refinement of language and did not involve scientific content generation or data analysis.

REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 484–501. Springer, 2020.

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80, pp. 274–283. PMLR, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of 2017 IEEE Symposium on Security and Privacy (S&P)*, pp. 39–57. IEEE, 2017.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 2206–2216. PMLR, 2020.

Fuquan Gan and Yan Wo. Boosting the transferability of adversarial examples through gradient aggregation. *IEEE Transactions on Information Forensics and Security*, 20:5563–5576, 2025. doi: 10.1109/TIFS.2025.3574989.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

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 (CVPR)*, pp. 770–778, 2016.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Klim Kireev, Bogdan Kulynych, and Carmela Troncoso. Adversarial robustness for tabular data through cost and utility awareness. In *Proceedings of Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *Workshop of International Conference on Learning Representations*, 2017.

Minjong Lee and Dongwoo Kim. Robust evaluation of diffusion-based adversarial purification. *arXiv preprint arXiv:2303.09051*, 2023.

Chun Tong Lei, Hon Ming Yam, Zhongliang Guo, Yifei Qian, and Chun Pong Lau. Instant adversarial purification with adversarial consistency distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24331–24340, 2025.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 135–147. ACM, 2017.

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 (CVPR)*, pp. 2574–2582, 2016.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 16805–16827. PMLR, 2022.

Mincheol Park, Cheonjun Park, Seungseop Lim, Mijin Koo, Hyunwuk Lee, Won Woo Ro, and Suhyun Kim. Adversarial purification via super-resolution and diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4605–4615, 2025.

Gaozheng Pei, Shaojie Lyu, Gong Chen, Ke Ma, Qianqian Xu, Yingfei Sun, and Qingming Huang. Divide and conquer: Heterogeneous noise integration for diffusion-based adversarial purification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29268–29277, 2025.

Yuchen Ren, Zhengyu Zhao, Chenhao Lin, Bo Yang, Lu Zhou, Zhe Liu, and Chao Shen. Improving integrated gradient-based transferable adversarial examples by refining the integration path. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 6731–6739, 2025.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5019–5031, 2018.

Leo Schwinn, René Raab, An Nguyen, Dario Zanca, and Bjoern Eskofier. Exploring misclassifications of robust neural networks to enhance adversarial attacks. *Applied Intelligence*, 53:19843–19859, 2023.

Kaiyu Song, Hanjiang Lai, Yan Pan, and Jian Yin. Mimicdiffusion: Purifying adversarial perturbation via mimicking clean diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24665–24674, 2024.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

Qi Tan, Yi Zhao, Laizhong Cui, Qi Li, Ming Zhu, Xing Fu, Weiqiang Wang, Xiaotong Lin, and Ke Xu. Robust fraud transaction detection: A two-player game approach. In *Proceedings of Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2026.

Linyu Tang and Lei Zhang. Robust overfitting does matter: Test-time adversarial purification with fgsm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24347–24356, 2024.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Proceedings of International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019a.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019b.

John K. Tsotsos and Jun Luo. Probing the effect of selection bias on NN generalization with a thought experiment. *CoRR*, abs/2105.09934, 2021.

Yichen Wang, Hangtao Zhang, Hewen Pan, Ziqi Zhou, Xianlong Wang, Peijin Guo, Lulu Xue, Shengshan Hu, Minghui Li, and Leo Yu Zhang. Advedm: Fine-grained adversarial attack against vlm-based embodied agents. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

Larry Wasserman. All of statistics: a concise course in statistical inference, 2013.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Lei Wu and Zhanxing Zhu. Towards understanding and improving the transferability of adversarial examples in deep neural networks. In *Proceedings of The Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pp. 837–850. PMLR, 2020.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings of Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 7472–7482. PMLR, 2019.

A Proof of Theorem 1

Proof. Let $Z_i = \boldsymbol{\mu}_i^{\mathrm{T}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])$. Since $\{\boldsymbol{\mu}_i\}_{i=1}^d$ are eigen vectors of Σ, i.e., the covariance matrix of $\boldsymbol{X}$, we have

$$\mathbb{E}[Z_i^2] = \boldsymbol{\mu}_i^{\mathrm{T}}\mathbb{E}[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^{\mathrm{T}}]\boldsymbol{\mu}_i = \boldsymbol{\mu}_i^{\mathrm{T}}\Sigma\boldsymbol{\mu}_i = \lambda_i,$$

Therefore,

$$\mathbb{E}\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|^2 = \mathbb{E}\|(Z_k, \cdots, Z_d)\|^2 = \mathbb{E}(\sum\nolimits_{i=k}^{d} Z_i^2) = \sum\nolimits_{i=k}^{d} \lambda_i,$$

where the second equality depends on the fact that $\langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle = 0$ when $i \neq j$.

Finally, as $\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|^2 \geq 0$, according to Markov's inequality, we have

$$P(\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|^2 \geq \epsilon^2) \leq \frac{\mathbb{E}\|K_{\mathcal{S}}(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\|^2}{\epsilon^2} = \frac{\sum_{i=k}^{d} \lambda_i}{\epsilon^2},$$

which immediately concludes the proof. □

B Proof of Theorem 2

Proof. According to Tweedie's formula, if $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{I})$, the optimal DAE trained with $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\xi}$, where $\tilde{\boldsymbol{x}} \sim p(\boldsymbol{x})$, is

$$\mathcal{D}(\boldsymbol{x}) = \boldsymbol{x} + \sigma^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}).$$

Hence, we have $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \sigma^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_n)$.

Additionally, based on the L-smooth property of $\log p(\boldsymbol{x})$, we have

$$\log p(\boldsymbol{x}_{n+1}) \geq \log p(\boldsymbol{x}_n) - \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_n)^{\mathrm{T}} (\boldsymbol{x}_n - \boldsymbol{x}_{n+1}) - \frac{L}{2} \|\boldsymbol{x}_{n+1} - \boldsymbol{x}_n\|^2 \tag{7}$$

$$= \log p(\boldsymbol{x}_n) + \sigma^2 (1 - \frac{L\sigma^2}{2}) \|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_n)\|^2 \tag{8}$$

$$\geq \log p(\boldsymbol{x}_n). \tag{9}$$

The last inequality depends on the fact that $L\sigma^2 \leq 2$, which immediately concludes the proof. □

C Performance of more attacks under different ratios of adversarial samples.

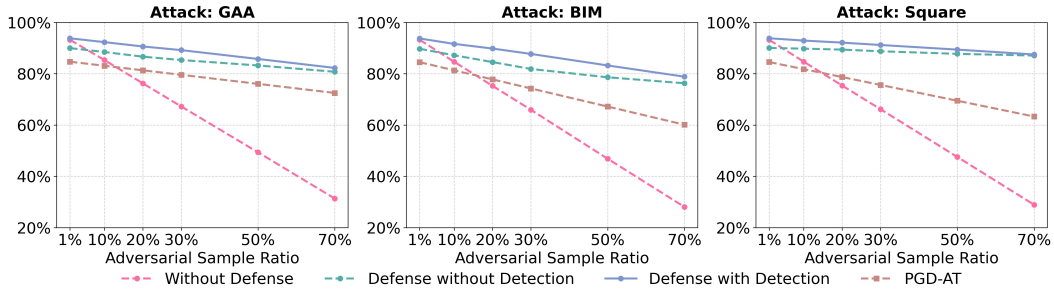

Figure 6: Performance under different ratios of adversarial examples generated by GAA, BIM, and Square.

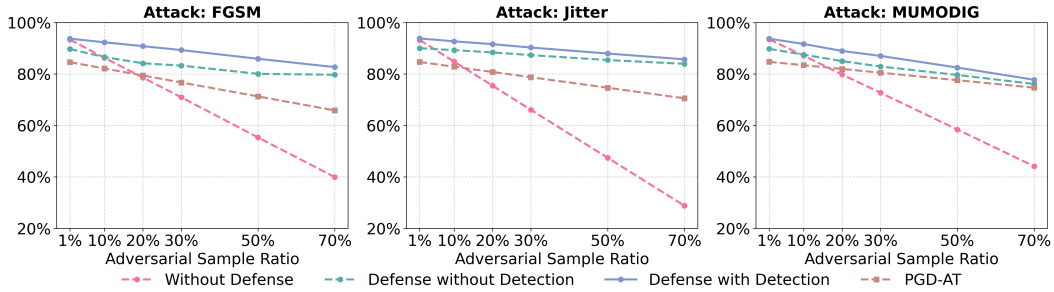

Figure 7: Performance under different ratios of adversarial examples generated by FGSM, Jitter, and MUMODIG.

Additional results under different ratios of adversarial examples are provided in Fig. 6 and Fig. 7. The results further show that the proposed defense maintains stable performance as the adversarial

sample ratio increases, while the version with the detection module generally provides higher accuracy than the counterpart without it. These results are consistent with the observations in the main experiments and further demonstrate the effectiveness of the detection module.

The effectiveness of the detector against various attacks are evaluated on CIFAR-10. As shown in Fig. 8, the detector achieves strong detection performance for almost all attacks. These results demonstrate the effectiveness of the detector in RIDE. Meanwhile, the consistently low FPRs verify that the detector effectively constrains redundant process on clean samples. Moreover, the adversarial examples generated by C&W and DeepFool obtain lower F1 scores, indicating that these examples are closer to clean samples and are therefore more difficult to identify. Combined with the results in Tab. 1, demonstrating that these adversarial examples are effectively purified by RIDE's prediction enhancement module.

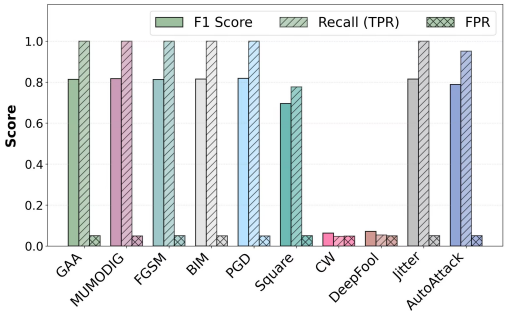

Figure 8: The performance of the detector on CIFAR-10 against different attacks.