Improving Accuracy-robustness Trade-off via Pixel Reweighted Adversarial Training

Jiacheng Zhang [1] **Feng Liu** [1] **Dawei Zhou** [2] **Jingfeng Zhang** [3] **Tongliang Liu** [4]

Abstract

Adversarial training (AT) trains models using *adversarial examples* (AEs), which are natural images modified with specific perturbations to mislead the model. These perturbations are constrained by a predefined perturbation budget ϵ and are equally applied to each pixel within an image. However, in this paper, we discover that *not all pixels contribute equally* to the accuracy on AEs (i.e., robustness) and accuracy on natural images (i.e., accuracy). Motivated by this finding, we propose ***P**ixel-reweighted **A**dve**R**sarial **T**raining (PART)*, a new framework that *partially* reduces ϵ for less influential pixels, guiding the model to focus more on key regions that affect its outputs. Specifically, we first use *class activation mapping* (CAM) methods to identify important pixel regions, then we keep the perturbation budget for these regions while lowering it for the remaining regions when generating AEs. In the end, we use these pixel-reweighted AEs to train a model. PART achieves a notable improvement in accuracy without compromising robustness on CIFAR-10, SVHN and TinyImagenet-200, justifying the necessity to *allocate distinct weights to different pixel regions* in robust classification.

1. Introduction

Since the discovery of *adversarial examples* (AEs) by Szegedy et al. (2014), the security of deep learning models has become an area of growing concern, especially in critical applications such as autonomous driving. For instance, Kumar et al. (2020) show that by adding imperceptible adversarial noise, a well-trained model misclassifies a 'Stop' traffic sign as a 'Yield' traffic sign. To make sure the trained model is robust to AEs, *adversarial training* (AT) stands out as a representative defensive framework (Goodfellow et al., 2015; Madry et al., 2018), which trains a model with generated AEs. Normally, AEs are crafted by intentionally adding perturbations to the natural images, aiming to mislead the model into making erroneous outputs.

In existing AT methods, e.g., AT (Madry et al., 2018), TRADES (Zhang et al., 2019) and MART (Wang et al., 2020), the magnitude of perturbations (for generating AEs) is usually constrained by a predefined perturbation budget, denoted as ϵ, and *keeps the same* on each pixel within an image by assuming a ℓ_∞-norm constraint. Based on a ℓ_∞-norm constraint, one AE can be generated by solving the following constraint optimization problem:

$$\max_{\boldsymbol{\Delta}} \ell(f(\mathbf{x}+\boldsymbol{\Delta}), y), \text{ subject to } \|\boldsymbol{\Delta}\|_\infty \leq \epsilon, \quad (1)$$

where ℓ is a loss function, f is a model, $\mathbf{x} \in \mathbb{R}^d$ is a natural image, y is the true label of $\mathbf{x}$, $\boldsymbol{\Delta} \in [-\epsilon, \epsilon]^d$ is the adversarial perturbation added to $\mathbf{x}$, $\|\cdot\|_\infty$ is the ℓ_∞-norm, d is the data dimension, and ϵ is the maximum allowed perturbation budget. Let $\boldsymbol{\Delta}^*$ be the solution of the above optimization problem, then $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\Delta}^*$ is the generated AE. Given that $\|\boldsymbol{\Delta}\|_\infty \leq \epsilon$, there is an implicit assumption in this AE generation process: all pixels have the *same* perturbation budget ϵ. We argue that this assumption may *overlook* the fact that different pixel regions influence the model's outputs differently (Geirhos et al., 2019; Brendel & Bethge, 2019; Hermann & Lampinen, 2020).

To the best of our knowledge, how the discrepancies of pixels would affect image classification in AT (i.e., robust classification) has not been well-investigated. Therefore, it is natural to raise the following question: *Are all pixels equally important in robust classification?*

In this paper, we mainly focus on ℓ_∞-norm constraint (we provide analysis on ℓ_2-norm constraint in Appendix A). By conducting a proof-of-concept experiment, we find that *not all pixels contribute equally* to the accuracy on AEs (i.e., robustness) and accuracy on natural images (i.e., accuracy). In

[1]School of Computing and Information Systems, The University of Melbourne [2]State Key Laboratory of Integrated Services Networks, Xidian University [3]School of Computer Science, The University of Auckland / RIKEN AIP [4]Sydney AI Centre, The University of Sydney. Correspondence to: Feng Liu <fengliu.ml@gmail.com>, Tongliang Liu <tliang.liu@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

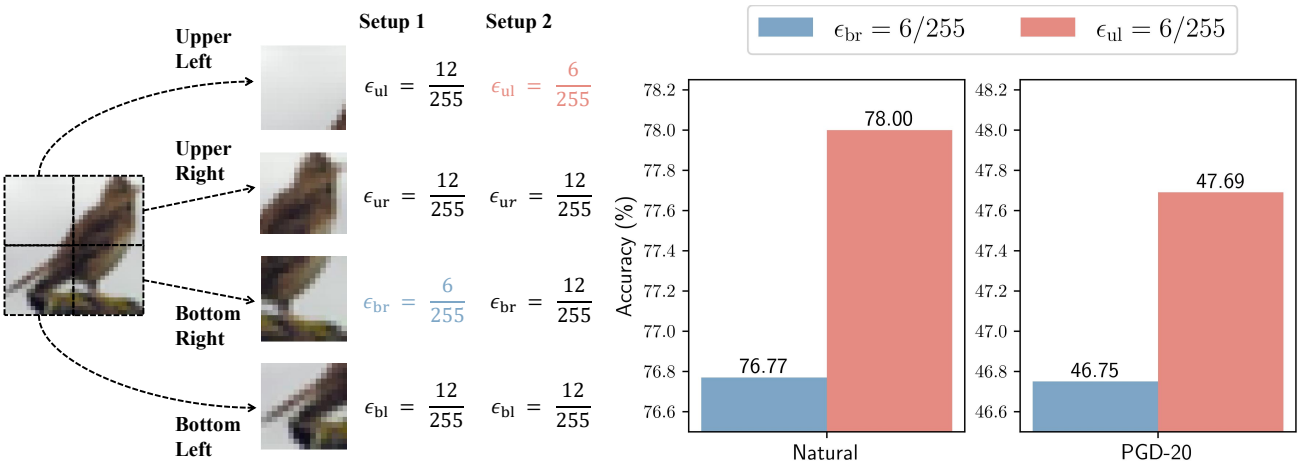


Figure 1. The proof-of-concept experiment. We find that fundamental discrepancies exist among different pixel regions. Specifically, we segment each image into four equal-sized regions (i.e., ul, short for upper left; ur, short for upper right; br, short for bottom right; bl, short for bottom left) and adversarially train two ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) using AT (Madry et al., 2018) with the same experiment settings except for the allocation of ϵ. The robustness is evaluated by ℓ_∞-norm PGD-20 (Madry et al., 2018). With the same overall perturbation budgets (i.e., allocate one of the regions to $6/255$ and others to $12/255$), we find that both natural accuracy and adversarial robustness change significantly if the regional allocation on ϵ is different. For example, by changing $\epsilon_{\text{br}} = 6/255$ to $\epsilon_{\text{ul}} = 6/255$, accuracy gains a 1.23% improvement and robustness gains a 0.94% improvement.

our experiment (see Figure 1), we segment images into four equal regions and train two models with identical settings except for how ϵ is allocated across these regions. To clearly show the difference, we set $\epsilon = \{6/255, 12/255\}$. The variation in ϵ, while maintaining the same overall perturbation budget, results in a notable increase in natural accuracy (from 76.77% to 78%) and adversarial robustness (from 46.75% to 47.69%). This means changing the perturbation budgets for different parts of an image has the potential to boost accuracy and robustness *at the same time*.

Motivated by this finding, we propose a new framework called ***P**ixel-reweighted **A**dve**R**sarial **T**raining (PART)*, to *partially* lower ϵ for pixels that rarely influence the model's outputs, which guides the model to focus more on regions where pixels are important for model's outputs.

To implement PART, we need to understand how pixels influence the model's output first. There are several well-known techniques to achieve this purpose, such as classifier-agnostic methods (e.g., LIME (Ribeiro et al., 2016)) and classifier-dependent methods (e.g., CAM, short for *class activation mapping* (Selvaraju et al., 2017; Fu et al., 2020; Jiang et al., 2021)). Given that classic AE generation processes are fundamentally classifier-dependent (Goodfellow et al., 2015; Madry et al., 2018), we use CAM methods to identify the importance of pixels in terms of the influence on the model's outputs in PART. Then, we propose a ***Pixel**-reweighted **AE G**eneration* (Pixel-AG) method. Pixel-AG can keep the perturbation budget ϵ for important pixel regions while lowering the perturbation budget from ϵ to ϵ^{low} for the remaining regions when generating AEs. In the end, we can train a model with Pixel-AG-generated AEs by using existing AT methods (e.g., AT (Madry et al., 2018), TRADES (Zhang et al., 2019), and MART (Wang et al., 2020)). To further understand PART, we theoretically analyze how perturbation budgets affect AE generation given that features have unequal importance (see Section 3.3).

Through extensive evaluations on benchmark image datasets such as CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011) and TinyImagenet-200 (Wu, 2017), we demonstrate the effectiveness of PART in Section 4.1. Specifically, combined with different AT methods (Madry et al., 2018; Zhang et al., 2019; Wang et al., 2020), PART can boost natural accuracy by a notable margin with little to no degradation on adversarial robustness, and thus improve accuracy-robustness trade-off. Rade & Moosavi-Dezfooli (2022) emphasize that besides proposing defense methods robust to adversarial attacks, the negative impact on accuracy from AT also warrants attention. Differing from most AT methods, our method can effectively mitigate the negative impact on accuracy. Besides, PART is designed as a general framework that can be effortlessly incorporated with a variety of AT strategies (Madry et al., 2018; Zhang et al., 2019; Wang et al., 2020), CAM methods (Selvaraju et al., 2017; Fu et al., 2020; Jiang et al., 2021), and AE generation methods (Madry et al., 2018; Gao et al., 2022).

To deeply understand the performance of PART, we take a close look at the robust feature representations (see Figure 2). By emphasizing the important pixel regions during training,

we find that PART-based classifiers could indeed be guided *more* towards leveraging semantic information in images to make classification decisions. We treat this as an extra advantage of PART, and this might be one of the key reasons why our method can improve the accuracy-robustness trade-off. We provide more qualitative results in Appendix B. We summarize the main contributions of our work as follows:

- We find that different pixel regions contribute differently to robustness and accuracy in robust classification. With the same total perturbation budget, allocating varying budgets to different pixel regions can improve robustness and accuracy at the same time.
- We propose a new framework of AT, namely ***P**ixel-reweighted **A**dve**R**sarial **T**raining (PART)* to guide the model focusing more on regions where pixels are important for model's output, leading to a better alignment with semantic information.
- We empirically show that, compared to the existing defenses, PART achieves a notable improvement in accuracy-robustness trade-off on CIFAR-10, SVHN and TinyImagenet-200 against multiple adversarial attacks, including adaptive attacks.

2. Preliminaries

Adversarial training. The basic idea behind AT (Madry et al., 2018) is to train a model f with AEs generated from the original training data. The objective function of AT is defined as follows:

$$\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i + \mathbf{\Delta}_i^*), y_i),$$

where $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{\Delta}_i^*$ is the most adversarial variant of $\mathbf{x}_i$ within the ϵ-ball centered at $\mathbf{x}_i$, $\mathbf{\Delta}_i^* \in [-\epsilon, \epsilon]^d$ is the optimized adversarial perturbation added to $\mathbf{x}_i$, y_i is the true label of $\mathbf{x}_i$, ℓ is a loss function, and F is the set of all possible neural network models.

The ϵ-ball is defined as $B_\epsilon[\mathbf{x}] = \{\mathbf{x}' | \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \epsilon\}$, where $\|\cdot\|_\infty$ is the ℓ_∞ norm. The most adversarial variant of $\mathbf{x}_i$ within the ϵ-ball is commonly obtained by solving the constrained optimization problem in Eq. (1) using PGD (Madry et al., 2018):

$$\tilde{\mathbf{x}}_i^{(t+1)} = \tilde{\mathbf{x}}_i^{(t)} + \text{clip}(\tilde{\mathbf{x}}_i^{(t)} + \alpha \cdot \text{sign}(\nabla_{\tilde{\mathbf{x}}_i^{(t)}} \ell(f(\tilde{\mathbf{x}}_i^{(t)}), y_i)) - \mathbf{x}_i, -\epsilon, \epsilon), \quad (2)$$

where $\tilde{\mathbf{x}}_i^{(t)}$ is the AE at iteration t, α is the step size, $\text{sign}(\cdot)$ is the sign function, and $\text{clip}(\cdot, -\epsilon, \epsilon)$ is the clip function that projects the adversarial perturbation back into the ϵ-ball, i.e., $\mathbf{\Delta}_i^* \in [-\epsilon, \epsilon]^d$.

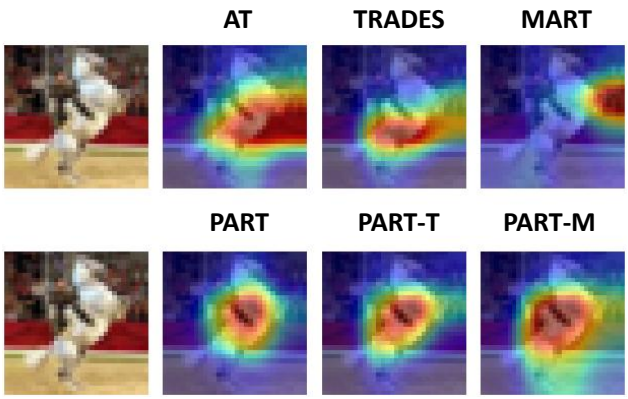


Figure 2. AT-based classifiers (the first row) vs. PART-based classifiers (the second row). The heatmaps are visualized by GradCAM (Selvaraju et al., 2017). In these heatmaps, a shift towards deeper red signifies a greater contribution to classification. This gradation in hue visually emphasizes the most influential pixel regions to the classification results. We find that PART-based methods could indeed be guided *more* towards leveraging semantic information in images (e.g., the horse) to make classification decisions.

Class activation mapping. Vanilla CAM (Zhou et al., 2016) is designed for producing visual explanations of decisions made by *Convolutional Neural Networks* (CNNs) by computing a coarse localization map highlighting important regions in an image for predicting a concept. GradCAM (Selvaraju et al., 2017) improves upon CAM by using the gradient information flowing into the last convolutional layer of the CNN to assign importance values to each neuron. Specifically, let $A_k \in R^{u \times v}$ of width u and height v for any class c be the feature map obtained from the last convolutional layer of the CNN, and let Y_c be the score for class c. GradCAM computes the gradient of Y_c with respect to the feature map A_k, which can be defined as follows:

$$\alpha_{c,k} = \frac{1}{Z} \sum_i \sum_j \frac{\partial Y_c}{\partial A_{k,ij}},$$

where Z is a normalization constant. GradCAM then produces the class activation map L_c for class c by computing the weighted combination of feature maps:

$$L_c = \text{ReLU}(\sum_k \alpha_{c,k} A_k).$$

In this paper, we mainly use GradCAM to identify the importance of the pixel regions since we find that the performance of PART with different CAM methods barely changes.

3. Pixel-reweighted Adversarial Training

In this paper, we find that *not all pixels contribute equally* to the robustness and accuracy by conducting a proof-of-concept experiment. According the Figure 1, given the same

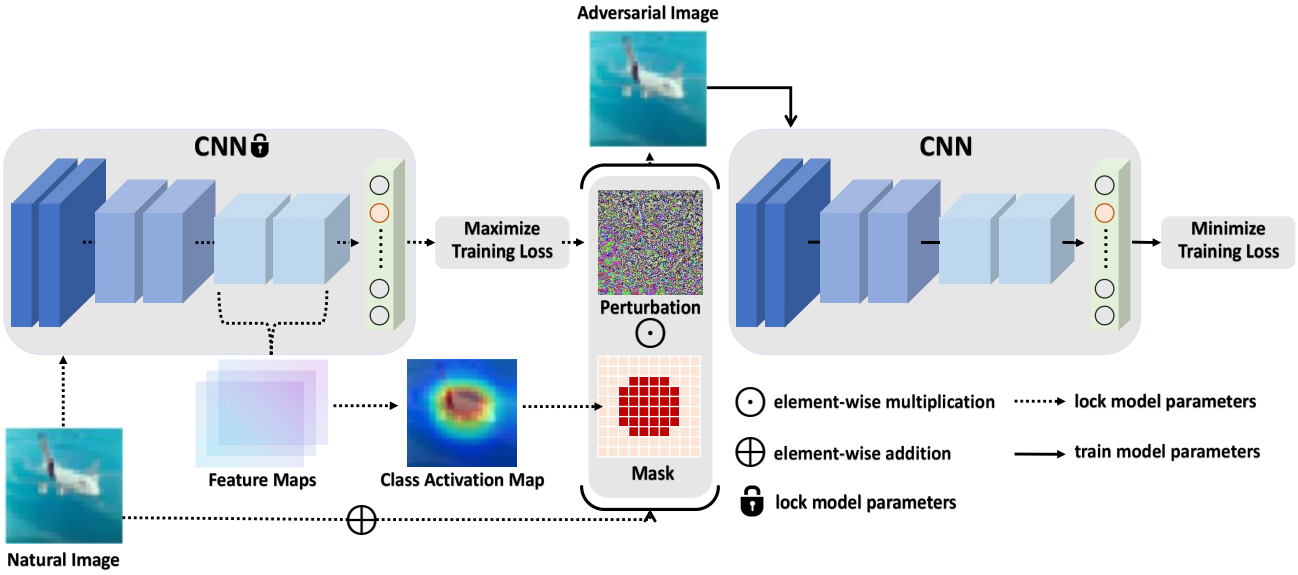


Figure 3. An overview of the training procedure for PART. Compared to AT, PART leverages the power of CAM methods to identify important pixel regions. Based on the class activation map, we element-wisely multiply a mask to the perturbation to keep the perturbation budget ϵ for important pixel regions while shrinking it to ϵ^{low} for their counterparts during the generation process of AEs.

overall perturbation budgets (i.e., allocate one of the regions to $6/255$ and others to $12/255$), we find that both natural accuracy and adversarial robustness change significantly if the regional allocation on ϵ is different. For example, by changing $\epsilon_{\text{br}} = 6/255$ to $\epsilon_{\text{ul}} = 6/255$, accuracy gains a 1.23% improvement and robustness gains a 0.94% improvement. This means changing the perturbation budgets for different parts of an image has the potential to boost robustness and accuracy *at the same time*. Motivated by this finding, we propose a new framework, ***P**ixel-reweighted **A**dve**R**sarial **T**raining* (PART), to *partially* reduce ϵ for less influential pixels, guiding the model to focus more on key regions that affect its outputs. In this section, we begin by introducing the learning objective of PART and its empirical implementation. This will be followed by a theoretical analysis and a comparative discussion with related work.

3.1. Learning Objective of PART

Compared to the existing AT framework, PART will focus on generating AEs whose perturbation budget of each pixel may be different. Thus, we will first introduce the generation process of AEs within PART, and then conclude the learning objective of PART. For convenience, we provide a detailed description of notations in Appendix C.

AE generation process. Compared to Eq. (1), the constraint optimization problem (for generating AEs in PART) will be:

$$\max_{\boldsymbol{\Delta}} \ell(f(\mathbf{x}+\boldsymbol{\Delta}), y), \text{ subject to}$$
$$\|\boldsymbol{v}(\boldsymbol{\Delta}, \mathcal{I}^{\text{high}})\|_\infty \le \epsilon, \ \|\boldsymbol{v}(\boldsymbol{\Delta}, \mathcal{I}^{\text{low}})\|_\infty \le \epsilon^{\text{low}}, \quad (3)$$

where $\epsilon^{\text{low}} < \epsilon$, $\boldsymbol{\Delta} = [\delta_1, \ldots, \delta_d]$, $\mathcal{I}^{\text{high}}$ collect indexes of important pixels, $\mathcal{I}^{\text{low}} = [d]/\mathcal{I}^{\text{high}}$, and $\boldsymbol{v}$ is a function to transform a set (e.g., a set consisting of important pixels in $\boldsymbol{\Delta}$: $\{\delta_i\}_{i\in\mathcal{I}^{\text{high}}}$) to a vector. Then, $\boldsymbol{\Delta}^{\text{high}}$ consists of $\{\delta_i\}_{i\in\mathcal{I}^{\text{high}}}$, and $\boldsymbol{\Delta}^{\text{low}}$ consists of $\{\delta_i\}_{i\in\mathcal{I}^{\text{low}}}$. $\boldsymbol{\Delta}^{\text{high}} \in [-\epsilon, \epsilon]^{d^{\text{high}}}$ is the adversarial perturbation added to important pixel regions with dimension d^{high}, $\boldsymbol{\Delta}^{\text{low}} \in [-\epsilon^{\text{low}}, \epsilon^{\text{low}}]^{d^{\text{low}}}$ is the adversarial perturbation added to the remaining regions with dimension d^{low}, where $d^{\text{high}} = |\mathcal{I}^{\text{high}}|$ and $d^{\text{low}} = |\mathcal{I}^{\text{low}}|$. A higher value of d^{high} means that more pixels are regarded as important ones.

Learning objective. Given a training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$, a loss function ℓ, a function space F, and the largest perturbation budget ϵ, the PART-based algorithms should have the same learning objective as AT-based algorithms:

$$\min_{f\in F} \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i + \boldsymbol{\Delta}_i^*), y_i), \quad (4)$$

where $\boldsymbol{\Delta}_i^* = \arg\max_{\boldsymbol{\Delta}} \ell(f(\mathbf{x}_i + \boldsymbol{\Delta}), y_i)$, subject to $\|\boldsymbol{v}(\boldsymbol{\Delta}, \mathcal{I}^{\text{high}})\|_\infty \le \epsilon$, $\|\boldsymbol{v}(\boldsymbol{\Delta}, \mathcal{I}^{\text{low}})\|_\infty \le \epsilon^{\text{low}}$.

Despite the same objective function, the constraint of PART is clearly different from AT-based methods. In the following subsection, we will introduce how to achieve the above learning objective via an empirical method.

3.2. Realization of PART

We provide a visual illustration of the training procedure for PART using CAM methods in Figure 3 and detailed algorithmic descriptions in Appendix D.

Pixel-reweighted AE generation (Pixel-AG). The constraint optimization problem Eq. (3) implies that the overall perturbation $\mathbf{\Delta}$ consists of two parts: perturbation added to important pixel regions, i.e., $\mathbf{\Delta}^{\text{high}}$ and perturbation added to their counterparts, i.e., $\mathbf{\Delta}^{\text{low}}$.

To generate AEs with appropriate $\mathbf{\Delta}^{\text{high}}$ and $\mathbf{\Delta}^{\text{low}}$, we propose a method called ***Pixel**-reweighted **A**E **G**eneration* (Pixel-AG). Pixel-AG employs CAM methods to differentiate between important pixel regions and their counterparts. Take GradCAM as an example: once we compute the class activation map L_c from Eq. (3), Pixel-AG first resizes L_c to L'_c to match the dimensions d of a natural image $\mathbf{x} = [x_1, ..., x_d]$, i.e., $L'_c \in \mathbb{R}^d$. Then it scales L'_c to $\tilde{L}_c$ to make sure the pixel regions highlighted by GradCAM have a weight value $\omega > 1$. Let $\tilde{L}_c = [\omega_1, ..., \omega_d]$ and $\mathbf{\Delta} = [\delta_1, ..., \delta_d]$ consists of $\mathbf{\Delta}^{\text{high}}$ and $\mathbf{\Delta}^{\text{low}}$. Then, for any $i \in [d]$, we define $\delta_i \in \mathbf{\Delta}^{\text{high}}$ if $\omega_i > 1$ and $\delta_i \in \mathbf{\Delta}^{\text{low}}$ otherwise, subject to $\|\boldsymbol{v}(\mathbf{\Delta}, \mathcal{I}^{\text{high}})\|_\infty \leq \epsilon$ and $\|\boldsymbol{v}(\mathbf{\Delta}, \mathcal{I}^{\text{low}})\|_\infty \leq \epsilon^{\text{low}}$.

Technically, this is equivalent to element-wisely multiply a mask $\boldsymbol{m} = [m_1, ..., m_d]$ to a $\mathbf{\Delta}$ constraint by $\|\mathbf{\Delta}\|_\infty \leq \epsilon$, where each element of $\boldsymbol{m}$ is defined as:

$$m_i = \begin{cases} 1 & \text{if } \omega_i > 1 \\ \epsilon^{\text{low}}/\epsilon & \text{otherwise} \end{cases}.$$

Let $\mathbf{\Delta}^*$ be the optimal solution of $\mathbf{\Delta}$, then $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{\Delta}^*$ is the AE generated by Pixel-AG, which can be obtained by solving Eq. (4) using an adapted version of Eq. (2):

$$\tilde{\mathbf{x}}_i^{(t+1)} = \tilde{\mathbf{x}}_i^{(t)} + \boldsymbol{m} \odot \text{clip}(\tilde{\mathbf{x}}_i^{(t)} + \alpha \cdot \text{sign}(\nabla_{\tilde{\mathbf{x}}_i^{(t)}} \ell(f(\tilde{\mathbf{x}}_i^{(t)}), y_i)) - \mathbf{x}_i, -\epsilon, \epsilon),$$

where $\odot$ is the Hadamard product. By doing so, we element-wisely multiply a mask $\boldsymbol{m}$ to the perturbation to keep the perturbation budget ϵ for important pixel regions while shrinking it to ϵ^{low} for their counterparts.

How to select ϵ^{low}. Given that the value of ϵ^{low} is designed to be a small number (e.g., $6/255$) and the computational cost of AT is expensive, we do not apply any algorithms to search for an optimal ϵ^{low} to avoid introducing extra training time to our framework. Instead, we directly set $\epsilon^{\text{low}} = \epsilon - 1/255$ by default. Without losing generality, we thoroughly investigate the impact of different values of ϵ^{low} on the robustness and accuracy of our method (see Section 4.1). Designing an efficient searching algorithm for ϵ remains an open question, and we leave it as future work.

Burn-in period. To improve the effectiveness of PART, we integrate a *burn-in period* into our training process. Specifically, we use AT as a warm-up at the early stage of training. Then, we incorporate Pixel-AG into PART for further training. This is because the classifier is not properly learned initially, and thus may badly identify pixel regions that are important to the model's output. By default, we set the *burn-in-period* of PART to be the initial 20 epochs.

Integration with other methods. The innovation on the AE generation allows PART to be orthogonal to many AT methods (e.g., AT (Madry et al., 2018), TRADES (Zhang et al., 2019) and MART (Wang et al., 2020)), and thus PART can be easily integrated into these methods. Moreover, the constraint optimization problem in Eq. (4) is general and can be addressed using various existing algorithms, such as PGD (Madry et al., 2018) and MMA (Gao et al., 2022). Besides, many CAM methods can be used as alternatives to GradCAM, such as XGradCAM (Fu et al., 2020) and LayerCAM (Jiang et al., 2021). Therefore, the compatibility of PART allows itself to serve as a general framework.

3.3. How ϵ Affect the Generation of AEs

In this subsection, we study a toy setting to shed some light on how pixels with different levels of importance would affect the generated AEs. The proof of Lemma 3.1 and Theorem 3.2 can be found in Appendix F.

Consider a 2D data point $\mathbf{x} = [x_1, x_2]^T$ with label y and an adversarial perturbation $\mathbf{\Delta} = [\delta_1, \delta_2]^T$ that is added to $\mathbf{x}$ with $\delta_1 \in [-\epsilon_1, \epsilon_1]$ and $\delta_2 \in [-\epsilon_2, \epsilon_2]$, where ϵ_1 and ϵ_2 are maximum allowed perturbation budgets for δ_1 and δ_2, respectively. Let ℓ be a differentiable loss function and f be the model, The constraint optimization problem (used to generate AEs) can be formulated as follows:

$$\begin{aligned} \max_{\mathbf{\Delta}=[\delta_1,\delta_2]^T} \ & \ell(f(\mathbf{x}+\mathbf{\Delta}), y), \\ \text{subject to } & -\epsilon_1 \leq \delta_1 \leq \epsilon_1, \\ & -\epsilon_2 \leq \delta_2 \leq \epsilon_2. \end{aligned} \tag{5}$$

Then, based on the *Karush–Kuhn–Tucker* (KKT) conditions (Avriel, 2003) for constraint optimization problems, we can analyze the solutions to the above problem as follows.

Lemma 3.1. *Let δ_1^* and δ_2^* be the optimal solutions of Eq. (5). The generated AEs can be categorized into three cases: (i) The expressions of δ_1^* and δ_2^* do not contain ϵ_1 and ϵ_2. (ii) $\delta_1^* = \pm\epsilon_1$ and $\delta_2^* = \pm\epsilon_2$. (iii) $\delta_1^* = \pm\epsilon_1$ and δ_2^* is influenced by ϵ_1, or vise versa.*

From Lemma 3.1, we know that the generated AEs must be within these cases, as KKT provides necessary conditions that δ_1^* and δ_2^* must satisfy. Nevertheless, for different models, the solutions are different. Here we focus on the impact on linear models. Specifically, we consider a linear model $f(\mathbf{x}) = \omega_1 x_1 + \omega_2 x_2 + b$ for this problem, where ω_1 and ω_2 are the weights for pixels x_1 and x_2 respectively. It is clear that x_1 will significantly influence $f(\mathbf{x})$ more compared to x_2 if w_1 is larger than w_2. For simplicity, we use a square loss, which can be expressed as $\ell(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$.

Then, we solve Eq. (5) by the Lagrange multiplier method and show the results in Theorem 3.2.

Theorem 3.2. *Consider a linear model $f(\mathbf{x}) = \omega_1 x_1 + \omega_2 x_2 + b$ and a square loss $\ell(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$. Let δ_1^* and δ_2^* be the optimal solutions of Eq. (5). For case (iii) in Lemma 3.1, we have:*

$$\delta_2^* = \frac{y - f(\mathbf{x}) - \omega_1 \epsilon_1}{\omega_2}, \text{ subject to } \delta_1^* = \epsilon_1,$$

$$\delta_2^* = \frac{y - f(\mathbf{x}) + \omega_1 \epsilon_1}{\omega_2}, \text{ subject to } \delta_1^* = -\epsilon_1,$$

$$\delta_1^* = \frac{y - f(\mathbf{x}) - \omega_2 \epsilon_2}{\omega_1}, \text{ subject to } \delta_2^* = \epsilon_2,$$

$$\delta_1^* = \frac{y - f(\mathbf{x}) + \omega_2 \epsilon_2}{\omega_1}, \text{ subject to } \delta_2^* = -\epsilon_2.$$

From Theorem 3.2, the main takeaway is straightforward: If two pixels have different influences on the model's predictions, it will affect the generation process of AEs, leading to different solutions of the optimal δ^*. Thus, it probably influences the performance of AT.

Remark. Note that, we do not cover how different levels of pixel importance would affect the performance of AT. This is because, during AT, the generated AEs are highly correlated, making the training process quite complicated to analyze in theory. According to recent developments regarding learning with dependent data (Dagan et al., 2019), we can only expect generalization when weak dependence exists in training data. However, after the first training epoch in AT, the model already depends on all training data, meaning that the generated AEs in the following epochs are probably highly dependent on each other. Thus, we leave this as our future work.

3.4. Comparisons with Related Work

We briefly review related work of our method here, and a more detailed version can be found in Appendix E.

Reweighted adversarial training. The idea of using reweighted AT has been studied in the literature. For example, Cai et al. (2018) reweights adversarial data with different PGD iterations K. Wang et al. (2019) reweights the adversarial data with different convergence qualities. More recently, Ding et al. (2020) proposes to reweight adversarial data with instance-dependent perturbation bounds ϵ and Zhang et al. (2021) proposes a geometry-aware instance-reweighted AT framework (GAIRAT) that assigns different weights to adversarial loss based on the distance of data points from the class boundary. Wang et al. (2021) further improves upon GAIRAT, which proposes to use probabilistic margins to reweight AEs since they are continuous and path-independent. Our proposed method is fundamentally different from the existing methods. Existing reweighted AT methods primarily focus on instance-based reweighting, wherein each data instance is treated distinctly. PART *pioneers* a pixel-based reweighting strategy, which allows for distinct treatment of pixel regions within each instance.

Adversarial defenses with attention heatmap. The idea of leveraging attention heatmap to defend against adversarial attacks has also been studied in the literature. For example, Ross & Doshi-Velez (2018) shows that regularizing the gradient-based attribution maps can improve model robustness. Zhou et al. (2021) proposes to use class activation features to remove adversarial noise. Specifically, it crafts AEs by maximally disrupting the class activation features of natural examples and then trains a denoising model to minimize the discrepancies between the class activation features of natural and AEs. This method can be regarded as an adversarial purification method, which purifies adversarial examples towards natural examples. Our method is technically different since PART is based on the AT framework. Our method aims to train a robust model by allocating varying perturbation budgets to different pixel regions according to their importance to the classification decisions. The idea of PART is general and CAM methods only serve as a tool to identify influential pixel regions.

4. Experiments

We demonstrate the main experiment results in this section. More experiment details can be found in Appendix G and more experiment results can be found in Appendix H. The code can be found in `https://github.com/tmlr-group/PART`.

4.1. Experiment Settings

Dataset. We evaluate the effectiveness of PART mainly on three benchmark datasets, i.e., CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011) and TinyImagenet-200 (Wu, 2017). CIFAR-10 comprises 50,000 training and 10,000 test images, distributed across 10 classes. SVHN has 10 classes but consists of 73,257 training and 26,032 test images. To test the performance of our method on large-scale datasets, we follow Zhou et al. (2022) and adopt TinyImagenet-200, which extends the complexity by offering 200 classes, containing 100,000 training, 10,000 validation, and 10,000 test images. For the target models, following the idea in Zhou et al. (2023), we use ResNet (He et al., 2016) for CIFAR-10 and SVHN, and WideResNet (Zagoruyko & Komodakis, 2016) for TinyImagenet-200. Besides, we also evaluate the generalization ability of PART on CIFAR-10-C (Hendrycks & Dietterich, 2019), which is a modification of the original CIFAR-10 by applying 19 different types of common corruptions.

Table 1. Robustness (%) and accuracy (%) of defense methods on *CIFAR-10*, *SVHN* and *TinyImagenet-200*. We use s to denote the save frequency of the mask $\boldsymbol{m}$. We report the averaged results and standard deviations of three runs. We show the most successful defense in **bold**. The performance improvements and degradation are reported in green and red numbers.

Dataset	Method	Natural	PGD-20	MMA	AA
		ResNet-18			
CIFAR-10	AT	82.58 ± 0.14	**43.69 ± 0.28**	41.80 ± 0.10	41.63 ± 0.22
	PART (s = 1)	83.42 ± 0.26 (+ 0.84)	43.65 ± 0.16 (- 0.04)	**41.98 ± 0.03 (+ 0.18)**	**41.74 ± 0.04 (+ 0.11)**
	PART (s = 10)	**83.77 ± 0.15 (+ 1.19)**	43.36 ± 0.21 (- 0.33)	41.83 ± 0.07 (+ 0.03)	41.41 ± 0.14 (- 0.22)
	TRADES	78.16 ± 0.15	48.28 ± 0.05	45.00 ± 0.08	45.05 ± 0.12
	PART-T (s = 1)	79.36 ± 0.31 (+ 1.20)	**48.90 ± 0.14 (+ 0.62)**	**45.90 ± 0.07 (+ 0.90)**	**45.97 ± 0.06 (+ 0.92)**
	PART-T (s = 10)	**80.13 ± 0.16 (+ 1.97)**	48.72 ± 0.11 (+ 0.44)	45.59 ± 0.09 (+ 0.59)	45.60 ± 0.04 (+ 0.55)
	MART	76.82 ± 0.28	49.86 ± 0.32	45.42 ± 0.04	45.10 ± 0.06
	PART-M (s = 1)	78.67 ± 0.10 (+ 1.85)	**50.26 ± 0.17 (+ 0.40)**	**45.53 ± 0.05 (+ 0.11)**	**45.19 ± 0.04 (+ 0.09)**
	PART-M (s = 10)	**80.00 ± 0.15 (+ 3.18)**	49.71 ± 0.12 (- 0.15)	45.14 ± 0.10 (- 0.28)	44.61 ± 0.24 (- 0.49)
		ResNet-18			
SVHN	AT	91.06 ± 0.24	49.83 ± 0.13	47.68 ± 0.06	45.48 ± 0.05
	PART (s = 1)	93.14 ± 0.05 (+ 2.08)	**50.34 ± 0.14 (+ 0.51)**	**48.08 ± 0.09 (+ 0.40)**	**45.67 ± 0.13 (+ 0.19)**
	PART (s = 10)	**93.75 ± 0.07 (+ 2.69)**	50.21 ± 0.10 (+ 0.38)	48.00 ± 0.14 (+ 0.32)	45.61 ± 0.08 (+ 0.13)
	TRADES	88.91 ± 0.28	58.74 ± 0.53	53.29 ± 0.56	52.21 ± 0.47
	PART-T (s = 1)	91.35 ± 0.11 (+ 2.44)	**59.33 ± 0.22 (+ 0.59)**	**54.04 ± 0.16 (+ 0.75)**	**53.07 ± 0.67 (+ 0.86)**
	PART-T (s = 10)	**91.94 ± 0.18 (+ 3.03)**	59.01 ± 0.13 (+ 0.27)	53.80 ± 0.20 (+ 0.51)	52.61 ± 0.24 (+ 0.40)
	MART	89.76 ± 0.08	58.52 ± 0.53	52.42 ± 0.34	49.10 ± 0.23
	PART-M (s = 1)	91.42 ± 0.36 (+ 1.66)	**58.85 ± 0.29 (+ 0.33)**	**52.45 ± 0.03 (+ 0.03)**	**49.92 ± 0.10 (+ 0.82)**
	PART-M (s = 10)	**93.20 ± 0.22 (+ 3.44)**	58.41 ± 0.20 (- 0.11)	52.18 ± 0.14 (- 0.24)	49.25 ± 0.13 (+ 0.15)
		WideResNet-34-10			
TinyImagenet-200	AT	43.51 ± 0.13	11.70 ± 0.08	10.66 ± 0.11	10.53 ± 0.14
	PART (s = 1)	44.87 ± 0.21 (+ 1.36)	**11.93 ± 0.16 (+ 0.23)**	**10.96 ± 0.12 (+ 0.30)**	**10.76 ± 0.06 (+ 0.23)**
	PART (s = 10)	**45.59 ± 0.14 (+ 2.08)**	11.81 ± 0.10 (+ 0.11)	10.91 ± 0.08 (+ 0.25)	10.68 ± 0.10 (+ 0.15)
	TRADES	43.05 ± 0.15	13.86 ± 0.10	12.62 ± 0.16	12.55 ± 0.09
	PART-T (s = 1)	44.31 ± 0.12 (+ 1.26)	**14.08 ± 0.22 (+ 0.22)**	**13.01 ± 0.09 (+ 0.39)**	**12.84 ± 0.14 (+ 0.29)**
	PART-T (s = 10)	**45.16 ± 0.10 (+ 2.11)**	13.98 ± 0.15 (+ 0.12)	12.88 ± 0.12 (+ 0.26)	12.72 ± 0.08 (+ 0.17)
	MART	42.68 ± 0.22	14.77 ± 0.18	13.58 ± 0.13	13.42 ± 0.16
	PART-M (s = 1)	43.75 ± 0.24 (+ 1.07)	**14.93 ± 0.15 (+ 0.16)**	**13.76 ± 0.06 (+ 0.18)**	**13.68 ± 0.13 (+ 0.24)**
	PART-M (s = 10)	**45.02 ± 0.16 (+ 2.34)**	14.65 ± 0.14 (- 0.12)	13.41 ± 0.11 (- 0.17)	13.37 ± 0.15 (- 0.05)

Attack settings. We mainly use three adversarial attacks to evaluate the performances of defenses. They are ℓ_∞-norm PGD (Madry et al., 2018), ℓ_∞-norm MMA (Gao et al., 2022) and ℓ_∞-norm AA (Croce & Hein, 2020a). Among them, AA is a combination of three non-target white-box attacks (Croce & Hein, 2020b) and one targeted black-box attack (Andriushchenko et al., 2020). Recently proposed MMA (Gao et al., 2022) can achieve comparable performance to AA but is much more time efficient. The iteration number for PGD is set to 20 (Zhou et al., 2023), and the target selection number for MMA is set to 3 (Gao et al., 2022), respectively. For AA, we use the same setting as RobustBench (Croce et al., 2020). For all attacks, we set the maximuim allowed perturbation budget ϵ to $8/255$.

Defense settings. Following Zhou et al. (2022), we use three representative AT methods as the baselines: AT (Madry et al., 2018) and two optimized AT methods TRADES (Zhang et al., 2019) and MART (Wang et al., 2020). We set $\lambda = 6$ for both TRADES and MART. For all baseline methods, we use the ℓ_∞-norm non-targeted PGD-10 with random start to craft AEs in the training stage. We set $\epsilon = 8/255$ for all datasets, and $\epsilon^{\text{low}} = 7/255$ for our method. All the defense models are trained using SGD with a momentum of 0.9. We set the initial learning rate to 0.01 with batch size 128 for CIFAR-10 and SVHN. To save time, we set the initial learning rate to 0.02 with batch size 512 for TinyImagenet-200 (Gao et al., 2022; Zhou et al., 2023). We run all the methods for 80 epochs and divide the learning rate by 10 at epoch 60 to avoid robust overfitting (Rice et al., 2020). We set the initial 20 epochs to be the burn-in period.

4.2. Performance Evaluation and Analysis

Defending against general attacks. From Table 1, the results show that our method can notably improve the natural accuracy with little to no degradation in adversarial robustness compared to AT. Despite a marginal reduction in robustness by 0.04% on PGD-20, PART gains more on natural accuracy (e.g., 2.08% on SVHN and 1.36% on TinyImagenet-200). In most cases, PART can improve natural accuracy and robustness simultaneously. To avoid the bias caused by different AT methods, we apply the optimized AT methods TRADES and MART to our method (i.e., PART-T and

Table 2. Robustness (%) of defense methods against adaptive PGD on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. We report the averaged results and standard deviations of three runs. We show the most successful defense in **bold**.

ResNet-18						
Dataset	Method	Adaptive PGD-20	Adaptive PGD-40	Adaptive PGD-60	Adaptive PGD-80	Adaptive PGD-100
CIFAR-10	AT	37.67 ± 0.05	36.98 ± 0.03	36.86 ± 0.07	36.81 ± 0.04	36.72 ± 0.04
	PART	**37.73 ± 0.11**	**37.07 ± 0.08**	**36.89 ± 0.12**	**36.84 ± 0.10**	**36.84 ± 0.07**
	TRADES	43.42 ± 0.13	43.22 ± 0.11	43.19 ± 0.12	43.10 ± 0.08	43.08 ± 0.06
	PART-T	**43.98 ± 0.15**	**43.75 ± 0.09**	**43.73 ± 0.06**	**43.68 ± 0.10**	**43.61 ± 0.03**
	MART	44.60 ± 0.09	44.19 ± 0.14	44.05 ± 0.13	43.98 ± 0.05	43.96 ± 0.08
	PART-M	**44.96 ± 0.21**	**44.51 ± 0.17**	**44.41 ± 0.12**	**44.37 ± 0.06**	**44.35 ± 0.09**

PART-M). Compared to TRADES and MART, our method can still boost natural accuracy (e.g., 1.20% on CIFAR-10, 2.44% on SVHN and 1.26% on TinyImagenet-200 for PART-T, and 1.85% on CIFAR-10, 1.66% on SVHN and 1.07% on TinyImagenet-200) with at most a 0.10% drop in robustness, and thus our method can achieve a better accuracy-robustness trade-off. Notably, even when $s = 10$, PART-based methods consistently improve the accuracy-robustness trade-off. For example, on CIFAR-10, despite a 0.49% drop in AA accuracy for PART-M, there is a 3.18% increase in natural accuracy, resulting in a net gain of +2.69%.

Defending against adaptive attacks. Adaptive attacks assume attackers have all the knowledge about the proposed method, e.g., model architectures, model parameters, and how AEs are generated in PART. As a result, attackers can design a specific attack to break PART (Athalye et al., 2018). Given the details of Pixel-AG, we design an adaptive attack that aims to misguide the model to focus on pixel regions that have little contribution to the correct classification results, and thus break the defense. Technically, this is equivalent to breaking what a robust model currently focuses on. Specifically, we use Pixel-AG with PGD to craft AEs, with an increased ϵ^{low} of $8/255$ and ϵ of $12/255$. As shown in Table 2, despite an overall decrease in robustness, our defense presents a better resilience against adaptive attacks compared to other baseline methods. We provide more results against adaptive MMA in Appendix H.1 and find that our method can consistently outperform baseline methods.

Defending against common corruptions. We examine the generalizability of PART by comparing our method with other baseline methods on the CIFAR-10-C, which introduces a variety of real-world corruptions such as noise, blur, weather, and digital distortions. Our findings indicate that PART-based methods consistently outperform AT-based methods, demonstrating a significant improvement in domain generalization accuracy. This improvement highlights the robustness of PART in handling various types of corruption that the model may encounter in real-world scenarios. Detailed experimental results are provided in Appendix H.2.

Possibility of obfuscated gradients. We consider the five behaviours listed in Athalye et al. (2018) to identify the obfuscated gradients and results show that our method does not cause obfuscated gradients (see Appendix H.3).

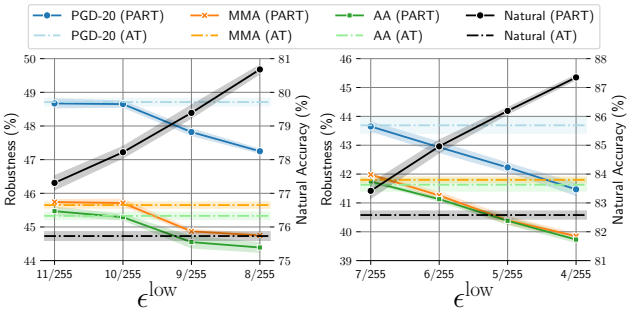


Figure 4. Impact of ϵ^{low} on robustness and accuracy of PART. **Left:** $\epsilon = 12/255$ and $\epsilon^{\text{low}} \in \{11/255, 10/255, 9/255, 8/255\}$. **Right:** $\epsilon = 8/255$, and $\epsilon^{\text{low}} \in \{7/255, 6/255, 5/255, 4/255\}$. Solid lines represent the performance of PART (s = 1), and dashed lines represent the performance of AT. We report the averaged results and standard deviations (i.e., shaded areas) of three runs.

4.3. Ablation Studies

Hyperparameter analysis. We examined how the hyperparameter ϵ^{low} influences our method's performance. Two experiment sets were conducted: first with $\epsilon = 12/255$ and ϵ^{low} ranging from $11/255$ to $8/255$, and second with $\epsilon = 8/255$ and ϵ^{low} ranging from $7/255$ to $4/255$. Results in Figure 4 show that lower ϵ^{low} will lead to a slight drop in robustness with more gains in natural accuracy, and thus improve the robustness-accuracy trade-off. Notably, we find that with a relatively large ϵ, moderately decrease ϵ^{low} leads to significant accuracy gains without affecting robustness (e.g., ϵ^{low} = $11/255$ or $10/255$ when $\epsilon = 12/255$).

Integration with other AE generation methods. We evaluate the effectiveness of our method by incorporating Pixel-AG into a more destructive attack, i.e., MMA (Gao et al., 2022) to generate AEs. With MMA, the performance of PART can be further boosted. (see Appendix H.4).

Integration with other CAM methods. To avoid potential bias caused by different CAM methods, we conduct experi-

ments to compare the performance of PART with different CAM methods such as GradCAM (Selvaraju et al., 2017), XGradCAM (Fu et al., 2020) and LayerCAM (Jiang et al., 2021). We find that these state-of-the-art CAM methods have approximately identical performance (see Appendix H.5). Thus, we argue that the performance of PART is barely affected by the choice of benchmark CAM methods.

Impact of attack iterations on PART. We investigate the impact of attack iterations on PART-based methods and find that attack iterations barely affect the performance of PART-based methods (see Appendix H.6).

4.4. Scalability and Applicability

Scalability of PART. As for whether our method can be scaled up or not, we find that it might be helpful to analyze if the algorithm running complexity will linearly increase when linearly increasing the number of samples or data dimensions. We find that our method *can* be scaled up and provide a detailed analysis in Appendix H.7.

Applicability of PART. We provide a detailed discussion on PART's applicability, including for untargeted attacks and beyond CNNs to *Vision Transformers* (ViTs) in Appendix H.8. Specifically, although CAM requires a target class, it will not affect the applicability of PART-based methods. Furthermore, we find that our idea has the potential to apply to ViTs. However, adversarially train a ViT is resource-consuming and we leave this as future work. In general, PART serves as a general idea rather than a specific method, and CAM is used as one of the tools to realize our idea.

4.5. Training Speed and Memory Consumption

The use of CAM methods will inevitably bring some extra cost. Luckily, we find that updating the mask $\boldsymbol{m}$ for every 10 epochs can effectively mitigate this problem. We use s to denote the save frequency of the mask m. For example, PART (s = 10) means we update m for every 10 epochs. We compare the training speed and the memory consumption of our method to different baseline methods in Appendix H.9.

5. Conclusion

We find that different pixel regions contribute unequally to robustness and accuracy. Motivated by this finding, we propose a new framework called ***P**ixel-reweighted **A**dve**R**sarial **T**raining (PART)*. PART partially reduces the perturbation budget for pixel regions that rarely influence the classification results, which guides the classifier to focus more on the essential part of images, leading to a notable improvement in accuracy-robustness trade-off. In general, we hope this simple yet effective framework could open up a new perspective in AT and lay the groundwork for advanced defenses that account for the discrepancies across pixel regions.

Acknowledgements

JCZ is supported by the Melbourne Research Scholarship and would like to thank Chaojian Yu, Yanming Guo, Weilun Xu and Yuhao Li for productive discussions. FL is supported by the Australian Research Council with grant numbers DP230101540 and DE240101089, and the NSF&CSIRO Responsible AI program with grant number 2303037. TLL is partially supported by the following Australian Research Council projects: FT220100318, DP220102121, LP220100527, LP220200949, and IC190100031.

Impact Statement

This paper presents work whose goal is to advance the field of Adversarial Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: A query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Avriel, M. *Nonlinear Programming: Analysis and Methods*. Courier Corporation, 2003. ISBN 0486432270.

Brendel, W. and Bethge, M. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *ICLR*, 2019.

Cai, Q., Liu, C., and Song, D. Curriculum adversarial training. In *IJCAI*, 2018.

Chefer, H., Schwartz, I., and Wolf, L. Optimizing relevance maps of vision transformers improves robustness. In *NeurIPS*, 2022.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020a.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020b.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Dagan, Y., Daskalakis, C., Dikkala, N., and Jayanti, S. Learning from weakly dependent data under dobrushin's condition. In *COLT*, 2019.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. MMA training: Direct input space margin maximization through adversarial training. In *ICLR*, 2020.

Dong, X., Han, J., Chen, D., Liu, J., Bian, H., Ma, Z., Li, H., Wang, X., Zhang, W., and Yu, N. Robust superpixel-guided attentional adversarial attack. In *CVPR*, 2020.

Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., and Li, B. Axiom-based grad-cam: Towards accurate visualization and explanation of CNNs. In *BMVC*, 2020.

Gao, R., Liu, F., Zhang, J., Han, B., Liu, T., Niu, G., and Sugiyama, M. Maximum mean discrepancy test is aware of adversarial attacks. In *ICML*, 2021.

Gao, R., Wang, J., Zhou, K., Liu, F., Xie, B., Niu, G., Han, B., and Cheng, J. Fast and reliable evaluation of adversarial robustness with minimum-margin attack. In *ICML*, 2022.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

Geirhos, R., Jacobsen, J., Michaelis, C., Zemel, R. S., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2 (11):665–673, 2020. doi: 10.1038/S42256-020-00257-Z.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Hermann, K. L. and Lampinen, A. K. What shapes feature representations? exploring datasets, architectures, and training. In *NeurIPS*, 2020.

Jiang, P., Zhang, C., Hou, Q., Cheng, M., and Wei, Y. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Trans. Image Process.*, 30:5875–5888, 2021.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kumar, K. N., Vishnu, C., Mitra, R., and Mohan, C. K. Black-box adversarial attacks in autonomous vehicle technology. In *AIPR*, 2020.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S. N. R., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. Diffusion models for adversarial purification. In *ICML*, 2022.

Rade, R. and Moosavi-Dezfooli, S. Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICLR*, 2022.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD*, 2016.

Rice, L., Wong, E., and Kolter, J. Z. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Ross, A. S. and Doshi-Velez, F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI*, 2018.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Shi, C., Holtz, C., and Mishne, G. Online adversarial purification based on self-supervised learning. In *ICLR*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2014.

Wang, Q., Liu, F., Han, B., Liu, T., Gong, C., Niu, G., Zhou, M., and Sugiyama, M. Probabilistic margins for instance reweighting in adversarial training. In *NeurIPS*, 2021.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. On the convergence and robustness of adversarial training. In *ICML*, 2019.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020.

Wu, J. Tiny imagenet challenge. 2017. URL https://api.semanticscholar.org/CorpusID:212697711.

Wu, S., Sang, J., Xu, K., Zhang, J., and Yu, J. Attention, please! adversarial defense via activation rectification and preservation. *ACM Trans. Multim. Comput. Commun. Appl.*, 2023.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*, 2018.

Yoon, J., Hwang, S. J., and Lee, J. Adversarial purification with score-based generative models. In *ICML*, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, 2016.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. S. Geometry-aware instance-reweighted adversarial training. In *ICLR*, 2021.

Zhang, S., Liu, F., Yang, J., Yang, Y., Li, C., Han, B., and Tan, M. Detecting adversarial data by probing multiple perturbations using expected perturbation score. In *ICML*, 2023.

Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *CVPR*, 2016.

Zhou, D., Wang, N., Peng, C., Gao, X., Wang, X., Yu, J., and Liu, T. Removing adversarial noise in class activation feature space. In *ICCV*, 2021.

Zhou, D., Wang, N., Han, B., and Liu, T. Modeling adversarial noise for adversarial training. In *ICML*, 2022.

Zhou, D., Wang, N., Yang, H., Gao, X., and Liu, T. Phase-aware adversarial defense for improving adversarial robustness. In *ICML*, 2023.

A. Perturbations with ℓ_2-norm Constraint

When discussing perturbations with ℓ_2-norm constraint, it's not accurate to assume each pixel has the same perturbation budget ϵ. This is because compared to a ℓ_∞-norm constraint, the entire perturbation $\mathbf{\Delta}$ is subject to a global bound, rather than each dimension having an identical perturbation budget. Let the dimension of a natural image $\mathbf{x}$ be d. For a perturbation $\mathbf{\Delta} = [\delta_1, ..., \delta_d]$, we have:

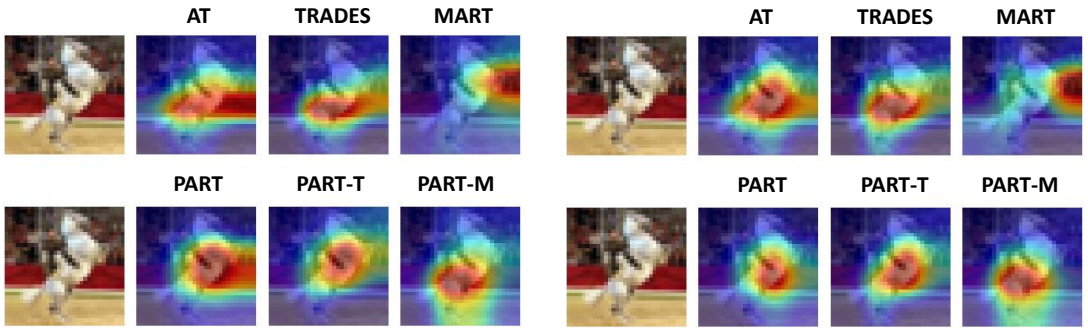
$$\|\mathbf{\Delta}\|_2 = \sqrt{\delta_1^2 + \delta_2^2 + ... + \delta_d^2} \leq \epsilon, \tag{6}$$

where ϵ is the maximum allowed perturbation budget. By Eq. (6), δ_i is not necessarily less than or equal to ϵ, e.g., certain elements might undergo minimal perturbations approaching 0, while others might be more significantly perturbed, as long as the entire vector's ℓ_2-norm remains under ϵ.

Thus, in this paper, the assumption that all pixels have the *same* perturbation budget ϵ is discussed by assuming the perturbations are bounded by ℓ_∞-norm constraint during the generation of AEs in training, i.e., $\|\mathbf{\Delta}\|_\infty \leq \epsilon$.

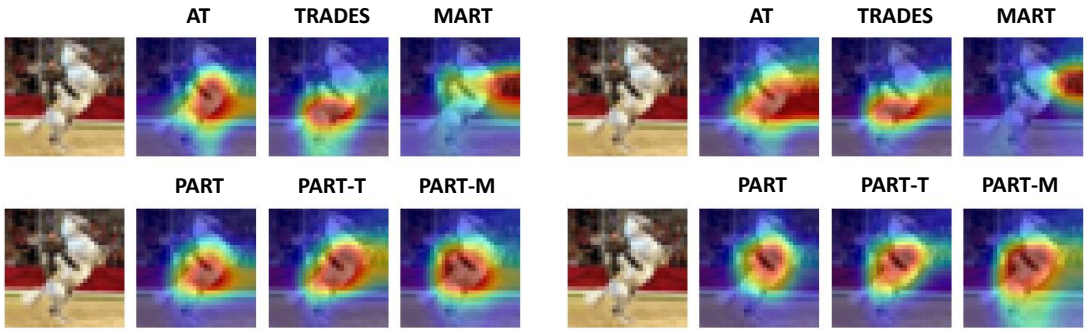
B. Qualitative Results

To deeply understand the performance of PART, we take a close look at the robust feature representations. By emphasizing the important pixel regions during training, we find that compared to AT-based classifiers, PART-based classifiers could indeed be guided *more* towards leveraging semantic information (i.e., the object) in images to make classification decisions. According to Geirhos et al. (2020), deep neural networks often rely on the image background to make classification decisions, neglecting the foreground. Therefore, we treat this as an extra advantage of PART, and this might be one of the key reasons why our method can improve the accuracy-robustness trade-off. We provide more qualitative results in the following figures.



(a) Epoch number = 30

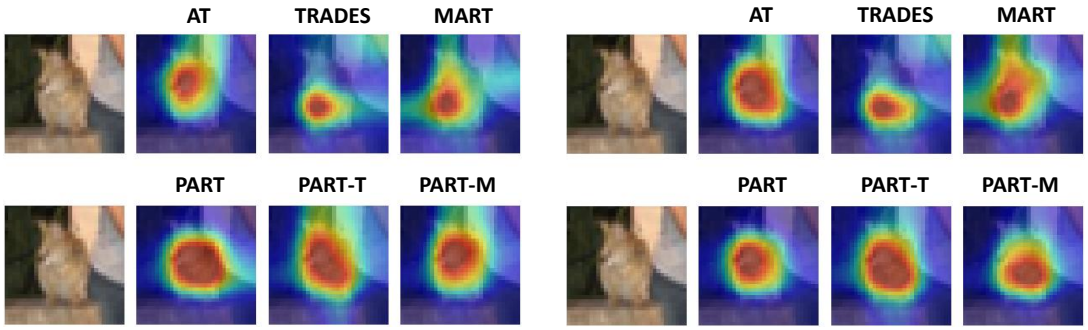
(b) Epoch number = 40



(c) Epoch number = 50

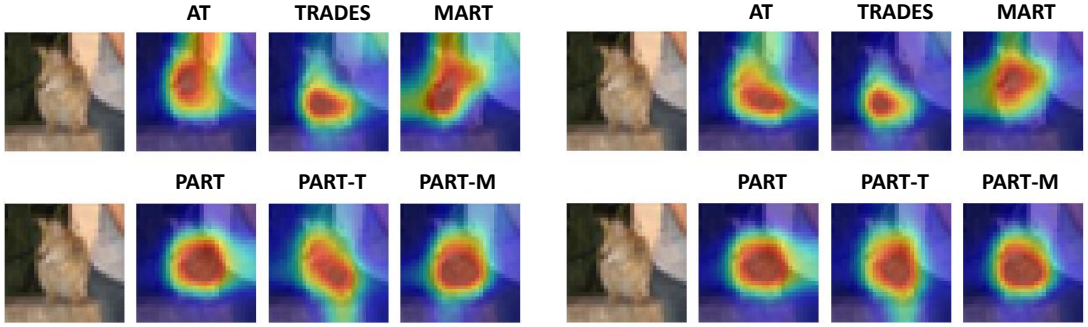
(d) Epoch number = 60

Figure 5. Qualitative results of how attention heatmaps change with epoch number $\in \{30, 40, 50, 60\}$ on *CIFAR-10*.



(a) Epoch number = 30

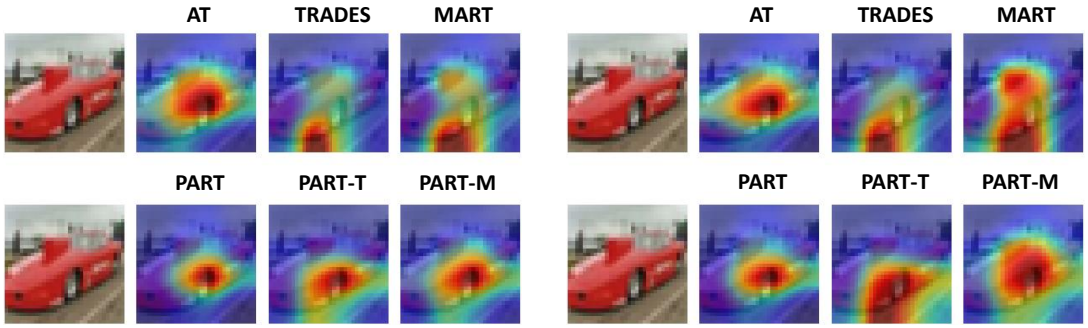
(b) Epoch number = 40



(c) Epoch number = 50

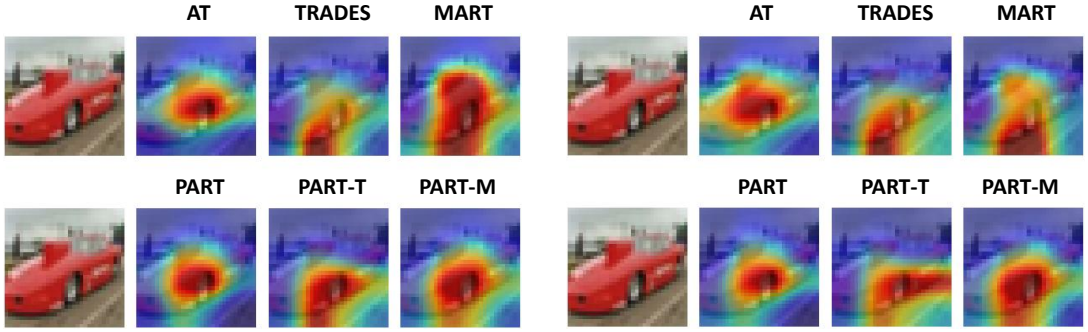
(d) Epoch number = 60

Figure 6. Qualitative results of how attention heatmaps change with epoch number $\in \{30, 40, 50, 60\}$ on *CIFAR-10*.



(a) Epoch number = 30

(b) Epoch number = 40



(c) Epoch number = 50

(d) Epoch number = 60

Figure 7. Qualitative results of how attention heatmaps change with epoch number $\in \{30, 40, 50, 60\}$ on *CIFAR-10*.

C. Notations in Section 3.1

ℓ	A loss function
f	A model
$\mathbf{x}$	A natural image
y	The true label of $\mathbf{x}$
d	The data dimension
$\boldsymbol{\Delta}$	The adversarial perturbation added to $\mathbf{x}$
$\boldsymbol{\Delta}^*$	The optimal solution of $\boldsymbol{\Delta}$
$\|\|\cdot\|\|_\infty$	The ℓ_∞-norm
ϵ	The maximum allowed perturbation budget for important pixels
ϵ^{low}	The maximum allowed perturbation budget for unimportant pixels
$\mathcal{I}^{\text{high}}$	Indexes of important pixels
$\mathcal{I}^{\text{low}}$	Indexes of unimportant pixels
$\boldsymbol{v}$	A function to transform a set to a vector
$\{\delta_i\}_{i\in\mathcal{I}^{\text{high}}}$	A set consisting of important pixels in $\boldsymbol{\Delta}$, i.e., $\boldsymbol{\Delta}^{\text{high}}$
$\{\delta_i\}_{i\in\mathcal{I}^{\text{low}}}$	A set consisting of unimportant pixels in $\boldsymbol{\Delta}$, i.e., $\boldsymbol{\Delta}^{\text{low}}$
$\|\mathcal{I}^{\text{high}}\|$	The dimension of important pixel regions, i.e., d^{high}
$\|\mathcal{I}^{\text{low}}\|$	The dimension of unimportant pixel regions, i.e., d^{low}

D. Algorithms

Algorithm 1 Mask Generation

Input: data dimension d, normalized class activation map $\tilde{L} = [\omega_1, ..., \omega_d]$, maximum allowed perturbation budgets ϵ, ϵ^{low}
Output: mask $\boldsymbol{m}$
Initialize mask $\boldsymbol{m} = \{m_1, ..., m_d\} = \mathbf{1}_d$
for $i = 1, ..., d$ **do**
  if $\omega_i > 1$ **then**
    $m_i = \epsilon^{\text{low}}/\epsilon$
  end if
end for

Algorithm 2 Pixel-reweighted AE Generation (Pixel-AG)

Input: data $\mathbf{x} \in \mathcal{X}$, label $y \in \mathcal{Y}$, model f, loss function ℓ, step size α, number of iterations K for inner optimization, maximum allowed perturbation budget ϵ
Output: adversarial example $\tilde{\mathbf{x}}$
Obtain mask $\boldsymbol{m}$ by Algorithm 1
$\tilde{\mathbf{x}} \leftarrow \mathbf{x}$
for $k = 1, ..., K$ **do**
  $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \boldsymbol{m} \odot \text{clip}(\tilde{\mathbf{x}} + \alpha\text{sign}(\nabla_{\tilde{\mathbf{x}}}\ell(f(\tilde{\mathbf{x}}), y)) - \mathbf{x}, -\epsilon, \epsilon)$
end for

Algorithm 3 Pixel-reweighted Adversarial Training (PART)

Input: network f with parameters $\boldsymbol{\theta}$, training dataset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, learning rate η, number of epochs T, batch size n, numebr of batches N
Output: Robust network f
for epoch $= 1, ..., T$ **do**
  for mini-batch $= 1, ..., N$ **do**
    Read mini-batch $B = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ from S
    for $i = 1, ..., n$ (in parallel) **do**
      Obtain adversarial data $\tilde{\mathbf{x}}_i$ of $\mathbf{x}_i$ by Algorithm 2
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \ell(f(\tilde{\mathbf{x}}_i), y_i)$
  end for
end for

E. Related Work

Adversarial attacks with class activation mapping. Dong et al. (2020) proposes an attack method that leverages superpixel segmentation and class activation mapping to focus on regions of an image that are most influential in classification decisions. It highlights the importance of considering perceptual features and classification-relevant regions in crafting effective AEs.

Our method, on the other hand, leverages class activation mapping to identify important pixel regions and use pixel-reweighted AEs to train a model that is not only robust to adversarial attacks, but also improves natural accuracy.

Adversarial training. To combat the threat of adversarial attacks, a myriad of defense mechanisms have emerged, such as perturbation detection (Ma et al., 2018; Xu et al., 2018; Gao et al., 2021; Zhang et al., 2023), adversarial purification (Shi et al., 2021; Yoon et al., 2021; Nie et al., 2022) and adversarial training (AT) (Madry et al., 2018; Zhang et al., 2019; Wang et al., 2020). Among these, AT stands out as a representative strategy (Goodfellow et al., 2015; Madry et al., 2018), which directly generates and incorporates AEs during the training process, forcing the model to learn the underlying distributions of AEs. Besides vanilla AT (Madry et al., 2018), many alternatives have been proposed. For example, from the perspective of improving objective functions, Zhang et al. (2019) proposes to optimize a surrogate loss function, which is derived based on a theoretical upper bound and a lower bound. Wang et al. (2020) investigates the unique impact of misclassified examples on the eventual robustness. They discover that misclassified examples significantly influence the final robustness and restructure the adversarial risk to include a distinct differentiation of misclassified examples through regularization. From the perspective of reweighting, Cai et al. (2018) reweights adversarial data with different PGD iterations K. Wang et al. (2019) reweights the adversarial data with different convergence qualities. More recently, Ding et al. (2020) proposes to reweight adversarial data with instance-dependent perturbation bounds ϵ and Zhang et al. (2021) proposes a geometry-aware instance-reweighted AT framework (GAIRAT) that assigns different weights to adversarial loss based on the distance of data points from the class boundary. Wang et al. (2021) further improves upon GAIRAT, which proposes to use probabilistic margins to reweight AEs since they are continuous and path-independent.

Our proposed method is fundamentally different from the existing methods. Existing reweighted AT methods primarily focus on instance-based reweighting, wherein each data instance is treated distinctly. Our proposed method, on the other hand, pioneers a pixel-based reweighting strategy, which allows for distinct treatment of pixel regions within each instance. Moreover, the design of PART is orthogonal to the state-of-the-art optimized AT methods such as TRADES (Zhang et al., 2019) and MART (Wang et al., 2020). This compatibility ensures that PART can be seamlessly integrated into these established frameworks, thereby extending its utility.

Adversarial defenses with attention heatmap. The idea of leveraging attention heatmap to defend against adversarial attacks has also been studied in the literature. For example, Ross & Doshi-Velez (2018) shows that regularizing the gradient-based attribution maps can improve model robustness. Zhou et al. (2021) proposes to use class activation features to remove adversarial noise. Specifically, it crafts AEs by maximally disrupting the class activation features of natural examples and then trains a denoising model to minimize the discrepancies between the class activation features of natural and AEs. This method can be regarded as an adversarial purification method, which purifies adversarial examples towards natural examples. Wu et al. (2023) proposes an Attention-based Adversarial Defense (AAD) framework that uses GradCAM to rectify and preserve the visual attention area, which aims to improve the robustness against adversarial attacks by aligning

the visual attention area between adversarial and original images.

Our method is technically different since PART is based on the AT framework. Our method aims to train a robust model by allocating varying perturbation budgets to different pixel regions according to their importance to the classification decisions. CAM methods, in our method, only serve as a tool to identify influential pixel regions.

Class activation mapping. Vanilla CAM (Zhou et al., 2016) is designed for producing visual explanations of decisions made by CNN-based models by computing a coarse localization map highlighting important regions in an image for predicting a concept. Besides vanilla CAM, many improved CAM methods have been proposed. For example, GradCAM (Selvaraju et al., 2017) improves upon CAM by using the gradient information flowing into the last convolutional layer of the CNN to assign importance values to each neuron, enabling the production of class-discriminative visualizations without the need for architectural changes or re-training. XGradCAM (Fu et al., 2020) introduces two axioms to improve the sensitivity and conservation of GradCAM. Specifically, it uses a modified gradient to better capture the importance of each feature map and a normalization term to preserve the spatial information of the feature maps. LayerCAM (Jiang et al., 2021) generates class activation maps not only from the final convolutional layer but also from shallow layers. This allows for both coarse spatial locations and fine-grained object details to be captured.

F. Proof of Lemma 3.1 and Theorem 3.2

Proof. To begin with the proof, we restate the problem setting as follows. Consider a 2D data point $\mathbf{x} = [x_1, x_2]^T$ with label y and an adversarial perturbation $\mathbf{\Delta} = [\delta_1, \delta_2]^T$ that is added to $\mathbf{x}$, with $\delta_1 \in [-\epsilon_1, \epsilon_1]$ and $\delta_2 \in [-\epsilon_2, \epsilon_2]$. We consider a linear model $f(\mathbf{x}) = \omega_1 x_1 + \omega_2 x_2 + b$ for this problem, where ω_1 and ω_2 are the weights for pixels x_1 and x_2 respectively. We use the square loss here as it is differentiable, which can be expressed as $\ell(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$. The objective of our problem is to find $\mathbf{\Delta}$ that can maximize $\ell(f(\mathbf{x} + \mathbf{\Delta}), y)$, which is equivalent to minimizing its negative counterpart. Thus, the constraint optimization problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & -(y - f(\mathbf{x} + \mathbf{\Delta}))^2, \\ \text{subject to} \quad & \delta_1 \le \epsilon_1, -\delta_1 \le \epsilon_1, \delta_2 \le \epsilon_2, -\delta_2 \le \epsilon_2. \end{aligned}$$

By using Lagrange multiplier method, we can construct the following Lagrange function $\mathcal{L}$:

$$\mathcal{L} = -(y - f(\mathbf{x} + \mathbf{\Delta}))^2 + \lambda_1(\delta_1 - \epsilon_1) + \lambda_2(-\delta_1 - \epsilon_1) + \lambda_3(\delta_2 - \epsilon_2) + \lambda_4(-\delta_2 - \epsilon_2). \tag{7}$$

Expanding $\mathcal{L}$, we have:

$$\begin{aligned} \mathcal{L} = & -y^2 + 2y\omega_1 x_1 + 2y\omega_1\delta_1 + 2y\omega_2 x_2 + 2y\omega_2\delta_2 + 2yb - \omega_1^2 x_1^2 - 2\omega_1^2 x_1\delta_1 \\ & - 2\omega_1\omega_2 x_1 x_2 - 2\omega_1\omega_2 x_1\delta_2 - 2\omega_1 x_1 b - \omega_1^2\delta_1^2 - 2\omega_1\omega_2 x_2\delta_1 - 2\omega_1\omega_2\delta_1\delta_2 \\ & - 2\omega_1\delta_1 b - \omega_2^2 x_2^2 - 2\omega_2^2 x_2\delta_2 - 2\omega_2 x_2 b - \omega_2^2\delta_2^2 - 2\omega_2\delta_2 b - b^2 \\ & + \lambda_1\delta_1 - \lambda_1\epsilon_1 - \lambda_2\delta_1 - \lambda_2\epsilon_1 + \lambda_3\delta_2 - \lambda_3\epsilon_2 - \lambda_4\delta_2 - \lambda_4\epsilon_2. \end{aligned} \tag{8}$$

Taking the derivatives with respect to δ_1 and δ_2 and setting them to zero, we have:

$$\frac{\partial\mathcal{L}}{\partial\delta_1} = 2y\omega_1 - 2\omega_1^2 x_1 - 2\omega_1^2\delta_1 - 2\omega_1\omega_2 x_2 - 2\omega_1\omega_2\delta_2 - 2\omega_1 b + \lambda_1 - \lambda_2 = 0. \tag{9}$$

$$\frac{\partial\mathcal{L}}{\partial\delta_2} = 2y\omega_2 - 2\omega_2^2 x_2 - 2\omega_2^2\delta_2 - 2\omega_1\omega_2 x_1 - 2\omega_1\omega_2\delta_1 - 2\omega_2 b + \lambda_3 - \lambda_4 = 0. \tag{10}$$

Solving Eq. (9) and Eq. (10), we can get the expressions for λ_1^*, λ_2^*, λ_3^* and λ_4^*:

$$\begin{aligned} \lambda_1^* &= 2\omega_1^2 x_1 + 2\omega_1^2\delta_1^* + 2\omega_1\omega_2 x_2 + 2\omega_1\omega_2\delta_2^* + 2\omega_1 b - 2y\omega_1 + \lambda_2^*, \\ \lambda_2^* &= 2y\omega_1 - 2\omega_1^2 x_1 - 2\omega_1^2\delta_1^* - 2\omega_1\omega_2 x_2 - 2\omega_1\omega_2\delta_2^* - 2\omega_1 b + \lambda_1^*, \\ \lambda_3^* &= 2\omega_2^2 x_2 + 2\omega_2^2\delta_2^* + 2\omega_1\omega_2 x_1 + 2\omega_1\omega_2\delta_1^* + 2\omega_2 b - 2y\omega_2 + \lambda_4^*, \\ \lambda_4^* &= 2y\omega_2 - 2\omega_2^2 x_2 - 2\omega_2^2\delta_2^* - 2\omega_1\omega_2 x_1 - 2\omega_1\omega_2\delta_1^* - 2\omega_2 b + \lambda_3^*. \end{aligned}$$

This is based on the *Karush–Kuhn–Tucker* (KKT) conditions (Avriel, 2003):

$$\delta_1^* \leq \epsilon_1, -\delta_1^* \leq \epsilon_1, \delta_2^* \leq \epsilon_2, -\delta_2^* \leq \epsilon_2.$$
$$\lambda_1^* \geq 0, \lambda_2^* \geq 0, \lambda_3^* \geq 0, \lambda_4^* \geq 0.$$
$$\lambda_1^*(\delta_1^* - \epsilon_1) = 0, \lambda_2^*(-\delta_1^* - \epsilon_1) = 0, \lambda_3^*(\delta_2^* - \epsilon_2) = 0, \lambda_4^*(-\delta_2^* - \epsilon_2) = 0. \tag{11}$$

Consider Eq. (11), we can further see two conditions:

1. λ_1^* and λ_2^* cannot be greater than 0 simultaneously. Otherwise δ_1^* equals to ϵ_1 and $-\epsilon_1$ simultaneously. This only holds when $\epsilon_1 = -\epsilon_1 = 0$ which means there is no perturbation added to x_1, and thus breaks away from adversarial settings.
2. Similarly, λ_3^* and λ_4^* cannot be greater than 0 simultaneously.

Considering all the conditions, we can summarize the generated AEs into three cases:

1. When $\lambda_1^* = \lambda_2^* = \lambda_3^* = \lambda_4^* = 0$. If we substitute the values of λ^*s into Eq. (8), we can see all the terms related to ϵ_1 and ϵ_2 are eliminated. This means if we take the derivatives of Eq. (8) with respect to δ_1 and δ_2, the optimal δ_1^* and δ_2^* will be some expressions without ϵ_1 and ϵ_2. This means the optimized solutions are inside $(-\epsilon_1, \epsilon_1)$. If δ_1^* and δ_2^* are far from the boundary, moderately change ϵ would hardly affect the results.
2. When one of λ_1^*, λ_2^* is greater than 0, and one of λ_3^*, λ_4^* is greater than 0. Take $(\lambda_1^* > 0, \lambda_2^* = 0, \lambda_3^* > 0, \lambda_4^* = 0)$ as an example, both δ_1^* and δ_2^* reach the boundary condition Eq. (11), i.e., $\delta_1^* = \epsilon_1$ and $\delta_2^* = \epsilon_2$. If we substitute $\delta_1^* = \epsilon_1$ and $\delta_2^* = \epsilon_2$ and λ^*s into Eq. (7), we have:

   $$\mathcal{L} = -(y - f(\mathbf{x}) - \omega_1\epsilon_1 - \omega_2\epsilon_2)^2.$$

   We can see the significance of ϵ_1 and ϵ_2 is different if $\omega_1 \neq \omega_2$.
3. When only one of λ^*s is greater than 0, while others are 0. Take $(\lambda_1^* > 0, \lambda_2^* = \lambda_3^* = \lambda_4^* = 0)$ as an example, then $\delta_1^* = \epsilon_1$ according to Eq. (11). If we substitute $\delta_1^* = \epsilon_1$ into Eq. (10), we have:

   $$\delta_2^* = \frac{y - f(\mathbf{x}) - \omega_1\epsilon_1}{\omega_2}, \text{ subject to } \delta_1^* = \epsilon_1.$$

We list the remaining cases as follows:

1. $(\lambda_1^* = 0, \lambda_2^* > 0, \lambda_3^* = 0, \lambda_4^* > 0)$. In this case, $\delta_1^* = -\epsilon_1$ and $\delta_2^* = -\epsilon_2$.
2. $(\lambda_1^* = 0, \lambda_2^* > 0, \lambda_3^* > 0, \lambda_4^* = 0)$. In this case, $\delta_1^* = -\epsilon_1$ and $\delta_2^* = \epsilon_2$.
3. $(\lambda_1^* > 0, \lambda_2^* = 0, \lambda_3^* = 0, \lambda_4^* > 0)$. In this case, $\delta_1^* = \epsilon_1$ and $\delta_2^* = -\epsilon_2$.
4. $(\lambda_2^* > 0, \lambda_1^* = \lambda_3^* = \lambda_4^* = 0)$, then $\delta_1^* = -\epsilon_1$ according to Eq. (11). If we substitute $\delta_1^* = -\epsilon_1$ into Eq. (10), we have:

   $$\delta_2^* = \frac{y - f(\mathbf{x}) + \omega_1\epsilon_1}{\omega_2}, \text{ subject to } \delta_1^* = -\epsilon_1.$$

5. $(\lambda_3^* > 0, \lambda_1^* = \lambda_2^* = \lambda_4^* = 0)$, then $\delta_2^* = \epsilon_2$ according to Eq. (11). If we substitute $\delta_2^* = \epsilon_2$ into Eq. (9), we have:

   $$\delta_1^* = \frac{y - f(\mathbf{x}) - \omega_2\epsilon_2}{\omega_1}, \text{ subject to } \delta_1^* = \epsilon_2.$$

6. $(\lambda_4^* > 0, \lambda_1^* = \lambda_2^* = \lambda_3^* = 0)$, then $\delta_2^* = -\epsilon_2$ according to Eq. (11). If we substitute $\delta_2^* = -\epsilon$ into Eq. (9), we have:

   $$\delta_1^* = \frac{y - f(\mathbf{x}) + \omega_2\epsilon_2}{\omega_1}, \text{ subject to } \delta_1^* = -\epsilon_2.$$

□

G. Experiment Settings

Dataset. We evaluate the effectiveness of PART on three benchmark datasets, i.e., CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011) and TinyImagenet-200 (Wu, 2017). CIFAR-10 comprises 50,000 training and 10,000 test images, distributed across 10 classes, with a resolution of 32×32. SVHN has 10 classes but consists of 73,257 training and 26,032 test images, maintaining the same 32×32 resolution. To test the performance of our method on large-scale datasets, we follow Zhou et al. (2022) and adopt TinyImagenet-200, which extends the complexity by offering 200 classes with a higher resolution of 64×64, containing 100,000 training, 10,000 validation, and 10,000 test images. For the target models, following the idea in Zhou et al. (2023), we use ResNet (He et al., 2016) for CIFAR-10 and SVHN, and WideResNet (Zagoruyko & Komodakis, 2016) for TinyImagenet-200. Besides, we also evaluate the generalization ability of PART on CIFAR-10-C (Hendrycks & Dietterich, 2019) to see whether our method can improve corruption robustness. CIFAR-10-C was created by applying 19 different types of algorithmically generated corruptions to the original CIFAR-10. These corruptions simulate various real-world image degradations.

Attack settings. We mainly use three types of adversarial attacks to evaluate the performances of defenses. They are ℓ_∞-norm PGD (Madry et al., 2018), ℓ_∞-norm MMA (Gao et al., 2022) and ℓ_∞-norm AA (Croce & Hein, 2020a). Among them, AA is a combination of three non-target white-box attacks (Croce & Hein, 2020b) and one targeted black-box attack (Andriushchenko et al., 2020). Recently proposed MMA (Gao et al., 2022) can achieve comparable performance compared to AA but is much more time efficient. The iteration number for PGD is set to 20 (Zhou et al., 2023), and the target selection number for MMA is set to 3 (Gao et al., 2022), respectively. For AA, we use the same setting as RobustBench (Croce et al., 2020). For all attacks, we set the maximuim allowed perturbation budget ϵ to $8/255$.

Defense settings. Following Zhou et al. (2022), we use three representative AT methods as the baselines: AT (Madry et al., 2018) and two optimized AT methods TRADES (Zhang et al., 2019) and MART (Wang et al., 2020). We set λ = 6 for both TRADES and MART. For all baseline methods, we use the ℓ_∞-norm non-targeted PGD-10 with random start to craft AEs in the training stage. We set $\epsilon = 8/255$ for all datasets, and $\epsilon^{\text{low}} = 7/255$ for our method. All the defense models are trained using SGD with a momentum of 0.9. We set the initial learning rate to 0.01 with batch size 128 for CIFAR-10 and SVHN. To save time, we set the initial learning rate to 0.02 with batch size 512 for TinyImagenet-200 (Gao et al., 2022; Zhou et al., 2023). The step size α is set to $2/255$ for CIFAR-10 and TinyImagenet-200, and is set to $1/255$ for SVHN. The weight decay is 0.0002 for CIFAR-10, 0.0035 for SVHN and 0.0005 for TinyImagenet-200. We run all the methods for 80 epochs and divide the learning rate by 10 at epoch 60 to avoid robust overfitting (Rice et al., 2020). For PART, we set the initial 20 epochs to be the burn-in period.

H. Additional Experiments

H.1. Adaptive MMA Attack

Table 3. Robustness (%) of defense methods against adaptive MMA on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. We report the averaged results and standard deviations of three runs. We show the most successful defense in **bold**.

ResNet-18						
Dataset	Method	MMA-20	MMA-40	MMA-60	MMA-80	MMA-100
CIFAR-10	AT	35.36 ± 0.10	35.02 ± 0.05	34.93 ± 0.09	34.86 ± 0.06	34.85 ± 0.07
	PART	$\mathbf{35.67 \pm 0.07}$	$\mathbf{35.35 \pm 0.11}$	$\mathbf{35.29 \pm 0.13}$	$\mathbf{35.29 \pm 0.09}$	$\mathbf{35.17 \pm 0.05}$
	TRADES	40.14 ± 0.08	39.89 ± 0.12	39.93 ± 0.05	39.87 ± 0.08	39.82 ± 0.03
	PART-T	$\mathbf{40.78 \pm 0.13}$	$\mathbf{40.57 \pm 0.11}$	$\mathbf{40.51 \pm 0.08}$	$\mathbf{40.49 \pm 0.05}$	$\mathbf{40.48 \pm 0.02}$
	MART	39.14 ± 0.06	38.79 ± 0.13	38.80 ± 0.10	38.79 ± 0.05	38.74 ± 0.08
	PART-M	$\mathbf{40.56 \pm 0.11}$	$\mathbf{40.26 \pm 0.07}$	$\mathbf{40.23 \pm 0.12}$	$\mathbf{40.21 \pm 0.08}$	$\mathbf{40.20 \pm 0.07}$

For adaptive attacks, we conduct an additional experiment to test the robustness of defense methods against adaptive MMA (see Table 3). The choice of MMA over AA for adaptive attacks is due to AA's time-consuming nature as an ensemble of multiple attacks. MMA, in contrast, offers greater time efficiency and comparable performance to AA.

H.2. Generalizability of PART on Common Corruptions

Table 4. Accuracy (%) of defense methods on *CIFAR-10-C*. We use s to denote the save frequency of the mask m. Here we use $s = 10$. Images are corrupted at severity 1. Target model is ResNet-18. We show the most successful defense in **bold**.

ResNet-18						
Corruption	AT	PART (s = 10)	TRADES	PART-T (s = 10)	MART	PART-M (s = 10)
Gaussian Noise	81.05	**82.46**	76.05	**79.42**	77.57	**79.51**
Shot Noise	81.11	**82.83**	75.91	**79.70**	77.66	**79.74**
Impulse Noise	79.42	**80.76**	74.59	**78.03**	76.12	**78.16**
Speckle Noise	81.42	**82.68**	75.97	**79.68**	77.63	**79.79**
Defocus Blur	81.07	**82.48**	76.06	**79.42**	77.59	**79.50**
Glass Blur	77.82	**78.20**	72.60	**76.26**	74.17	**76.57**
Motion Blur	79.30	**80.13**	74.28	**77.43**	75.35	**77.45**
Zoom Blur	78.87	**79.30**	73.27	**76.74**	74.10	**76.74**
Gaussian Blur	81.05	**82.46**	76.05	**79.42**	77.57	**79.51**
Snow	81.09	**82.01**	76.13	**79.43**	77.63	**79.34**
Frost	78.96	**80.04**	73.90	**76.60**	75.01	**75.80**
Fog	79.34	**80.18**	72.95	**77.15**	75.29	**78.26**
Brightness	81.89	**83.22**	76.87	**80.16**	78.60	**80.26**
Spatter	81.03	**82.03**	75.81	**79.31**	77.62	**79.39**
Contrast	77.09	**77.67**	70.08	**75.00**	71.90	**75.81**
Elastic Transform	77.32	**78.16**	71.99	**75.39**	73.13	**75.68**
Pixelate	81.09	**82.63**	76.05	**79.48**	77.45	**79.43**
JPEG Compression	80.50	**81.91**	75.71	**79.09**	77.15	**79.29**
Saturate	78.01	**79.34**	73.59	**76.34**	74.70	**75.52**

H.3. Possibility of Obfuscated Gradients

We consider the five behaviours listed in (Athalye et al., 2018) to identify the obfuscated gradients:

(i). We find that *one-step attacks do not perform better than iterative attacks*. The accuracy of our method against PGD-1 is 76.31% (vs 43.65% against PGD-20). (ii). We find that *black-box attacks have lower attack success rates than white-box attacks*. We use ResNet-18 with AT as the surrogate model to generate AEs. The accuracy of our method against PGD-20 is 59.17% (vs 43.65% in the white-box setting). (iii). We find that *unbounded attacks reach 100% success*. The accuracy of our method against PGD-20 with $\epsilon = 255/255$ is 0%. (iv). We find that *random sampling does not find AEs*. For samples that are not successfully attacked by PGD, we randomly sample 100 points within the ϵ-ball and do not find adversarial data. (v). We find that *increasing distortion bound increases success.* The accuracy of our method against PGD-20 with increasing ϵ ($8/255$, $16/255$, $32/255$ and $64/255$) is 43.65%, 10.70%, 0.49% and 0%.

These results show that our method does not cause obfuscated gradients.

H.4. Different AE Generation Methods

Table 5. Comparison of PART's performance with different AE Generation methods on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. We report the averaged results and standard deviations of three runs.

ResNet-18						
Dataset	AE Generation	Method	Natural	PGD-20	MMA	AA
CIFAR-10	PGD-10	AT	82.58 ± 0.05	**43.69 ± 0.28**	41.80 ± 0.10	41.63 ± 0.22
		PART	**83.42 ± 0.26**	43.65 ± 0.06	**41.98 ± 0.03**	**41.74 ± 0.04**
	MMA	AT	81.76 ± 0.11	44.76 ± 0.14	42.31 ± 0.13	42.04 ± 0.15
		PART	**83.55 ± 0.28**	**44.99 ± 0.14**	**42.50 ± 0.22**	**42.09 ± 0.24**

H.5. Different CAM Methods

Table 6. Comparison of PART's performance with different CAM methods on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. We report the averaged results and standard deviations of three runs.

ResNet-18						
Dataset	Method	CAM	Natural	PGD-20	MMA	AA
CIFAR-10	PART	GradCAM	83.42 ± 0.26	43.65 ± 0.06	41.98 ± 0.03	41.74 ± 0.04
		XGradCAM	83.34 ± 0.18	43.53 ± 0.08	41.97 ± 0.05	41.74 ± 0.02
		LayerCAM	83.38 ± 0.21	43.67 ± 0.11	42.07 ± 0.09	42.03 ± 0.16

H.6. Impact of Attack Iterations

Table 7. Robustness (%) of defense methods against PGD with different iterations on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. We report the averaged results and standard deviations of three runs. We show the most successful defense in **bold**.

ResNet-18						
Dataset	Method	PGD-10	PGD-40	PGD-60	PGD-80	PGD-100
CIFAR-10	AT	44.83 ± 0.13	43.00 ± 0.10	42.83 ± 0.07	42.81 ± 0.03	42.81 ± 0.03
	PART	$\mathbf{45.20 \pm 0.17}$	$\mathbf{43.20 \pm 0.14}$	$\mathbf{43.09 \pm 0.09}$	$\mathbf{43.08 \pm 0.10}$	$\mathbf{42.93 \pm 0.07}$
	TRADES	48.81 ± 0.21	48.19 ± 0.13	48.16 ± 0.15	48.14 ± 0.08	48.08 ± 0.04
	PART-T	$\mathbf{49.41 \pm 0.11}$	$\mathbf{48.65 \pm 0.10}$	$\mathbf{48.64 \pm 0.13}$	$\mathbf{48.64 \pm 0.04}$	$\mathbf{48.62 \pm 0.03}$
	MART	49.98 ± 0.08	49.66 ± 0.16	49.66 ± 0.06	49.54 ± 0.03	49.47 ± 0.05
	PART-M	$\mathbf{50.50 \pm 0.19}$	$\mathbf{50.19 \pm 0.15}$	$\mathbf{50.09 \pm 0.04}$	$\mathbf{50.06 \pm 0.05}$	$\mathbf{50.05 \pm 0.02}$

Table 8. Robustness (%) and Accuracy (%) of PART against PGD with different iterations during training on *CIFAR-10*. We set the save frequency of the mask $\boldsymbol{m}$ to be 1. The target model is ResNet-18. We report the averaged results and standard deviations of three runs.

ResNet-18					
Dataset	Method	Natural	PGD-20	MMA	AA
CIFAR-10	PART (PGD-10)	83.42 ± 0.26	43.65 ± 0.16	41.98 ± 0.03	41.74 ± 0.04
	PART (PGD-20)	83.44 ± 0.19	43.64 ± 0.13	42.02 ± 0.13	41.82 ± 0.08
	PART (PGD-40)	83.36 ± 0.21	43.82 ± 0.08	42.09 ± 0.07	41.86 ± 0.11
	PART (PGD-60)	83.30 ± 0.15	44.02 ± 0.12	42.18 ± 0.05	41.91 ± 0.09

We conduct extra experiments to analyze the impact of attack iterations on the performance of CAM methods. Specifically, we test the robustness of defense methods against PGD with different iterations on CIFAR-10 (see Table 7). With the increase of attack iterations, the robustness of defense methods will decrease. This is because the possibility of finding worst-case examples will increase with more attack iterations. The effectiveness of CAM technology itself, however, is rarely influenced by attack iterations, as our method can consistently outperform baseline methods. Furthermore, we take a close look at how the number of attack iterations during training would affect the final performance of CAM methods (see Table 8). Similarly, if we increase the attack iterations during training, the model will become more robust as the model learns more worst-case examples during training. At the same time, the natural accuracy has a marginal decrease. Overall, we conclude that the performance of our method is stable and CAM methods are rarely affected by the attack iterations.

H.7. Scalability of PART

As for whether our method can be scaled up or not, we find that it might be helpful to analyze if the algorithm running complexity will linearly increase when linearly increasing the number of samples or data dimensions. The generation of class activation mapping mainly involves global average pooling, which has a complexity of $O(H_{\text{last}} \times W_{\text{last}} \times C_{\text{last}})$, where $H_{\text{last}}, W_{\text{last}}$, and C_{last} are the height, width, and number of channels in the last convolutional layer. The subsequent

generation of the heatmap involves a weighted combination of these pooled values with the feature maps, which also have a complexity proportional to the size of the feature map. When data linearly increases, the complexity of GradCAM is mainly influenced by two factors: the size of the input data and the structure of the CNN. When considering a single input sample with increased dimensions, such as a doubled width and height, the total pixel count quadruples due to the area being a product of these dimensions. This increases the computational complexity of convolutional layers linearly, as they process more pixels. However, as convolution is a local operation, this increase is generally manageable. Thus, for a single sample, GradCAM's computational complexity grows approximately linearly with the input data size. For multiple samples, the complexity scales linearly with the dataset size, as more samples require processing. Therefore, overall, we argue that PART-based methods *can* be scaled up. However, in practice, training a robust model on large datasets for AT-based methods is often resource-consuming. Thus, how to design a more efficient AT framework is always an open challenge.

H.8. Applicability of PART

Applicability of PART to untargeted attacks. CAM requires a target class to generate the attention heatmap. Therefore, some concerns may arise such as CAM might limit the effectiveness of our method in the context of untargeted attacks. However, we would like to clarify that in the PART framework, the role of CAM is primarily to identify important pixel regions that significantly influence the model's output. The key here is that the utilization of CAM is not to optimize these regions for a specific target class, but rather to create a mask that discerns areas of varying influence on the model's decision-making process. Once these important regions are identified via CAM, we convert this information into a mask. This mask is then used to differentially reweight the adversarial perturbations in the generation of AEs. This process is *independent* of whether the attack is targeted or untargeted. Therefore, although CAM requires a target class, it will not affect the applicability of PART-based methods to untargeted attacks.

Applicability of PART to Vision Transformers. CAM is specific to CNNs and not directly applicable to other architectures such as *Vision Transformers* (ViTs). However, the mechanism of ViTs allows them to produce a similar attention heatmap as CAM methods, as shown by Chefer et al. (2022), who improve *Out-Of-Distribution* (OOD) data generalization by focusing ViTs' attention on classification objects. This suggests the idea of PART *can* be extended to ViTs by using their regularized attention maps to reweigh adversarial examples (AEs). Nevertheless, adversarially train a ViT is resource-consuming and thus we leave this as future work. In general, we want to emphasize that PART is a general idea rather than a specific method and CAM is one of the tools to realize our idea. The main goal of our work is to provide insights on how to design an effective AT method by counting the fundamental discrepancies of pixel regions across images.

H.9. Extra Cost Introduced by CAM Methods

Table 9. Computational time (hours : minutes : seconds) and memory consumption (MB) of defense methods on *CIFAR-10*.

ResNet-18						
Dataset	GPU	Method	Training Speed	Difference	Memory Consumption	Difference
CIFAR-10	2*NVIDIA A100	AT	01:14:37	00:56:28	4608MB	345MB
		PART	02:11:05		4953MB	
		TRADES	01:44:19	01:02:09	4697MB	322MB
		PART-T	02:46:28		5019MB	
		MART	01:09:23	00:57:46	4627MB	338MB
		PART-M	02:07:09		4965MB	

The use of CAM methods will inevitably bring some extra cost. Luckily, we find that updating the mask $\boldsymbol{m}$ for every 10 epochs can effectively mitigate this problem. Regarding memory consumption, the majority of the memory is allocated for storing checkpoints, with only a small portion attributed to CAM technology. We compare the computational time (hours : minutes : seconds) and the memory consumption (MB) of our method to different AT methods. See Table 9 for more details.