

A APPENDIX

A.1 VISUALIZATION OF VARIOUS ATTACKS

To better present the visual effect of various kinds of adversarial attacks, we provide a high-resolution results on Caltech-256 in Figure 8. It turns out that Flow-based attack focus on local spatial vulnerability that mainly blur pixels in some local regions, while RT attacks cause a shape-based global spatial transformation. More importantly, our integrated spatial attacks are more comprehensive in the sense of spatial robustness, combining both local and local spatial sensitivity.

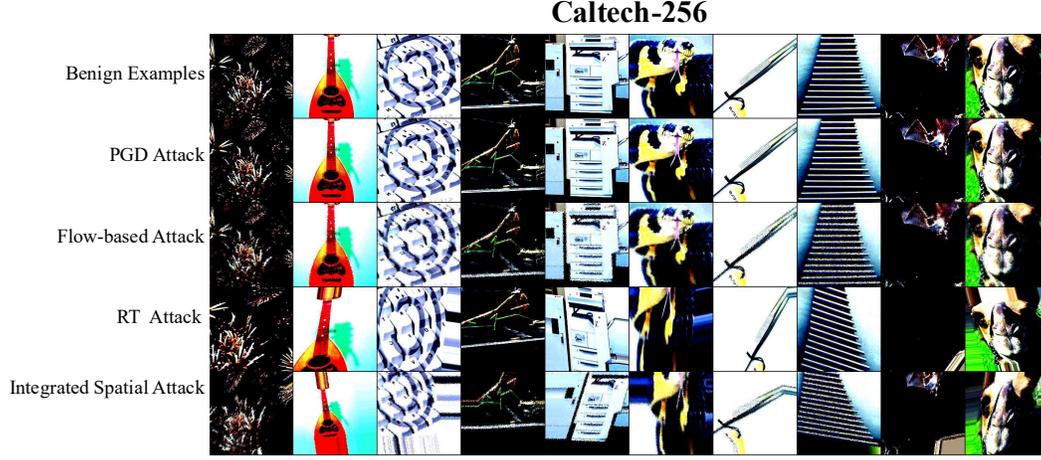


Figure 8: High-resolution images on Caltech-256.

A.2 PROOF OF PROPOSITION 1

Proof. Firstly, we have the following equations according to the definitions of loss function:

$$\begin{aligned}\mathcal{L}_\theta^{\text{CE}}(x_{w_F}, y) &= \log \sum_{i=1}^K \exp(f_\theta^i(x_{w_F})) - f_\theta^y(x_{w_F}) \\ \mathcal{L}_\theta^S(x_{w_F}, y) &= \log \sum_{i \neq y} \exp(f_\theta^i(x_{w_F})) - f_\theta^y(x_{w_F})\end{aligned}\quad (13)$$

Then, we compute their gradients with respect to flow vector x_{w_F} respectively. The gradient of $\mathcal{L}_\theta^{\text{CE}}(x_{w_F}, y)$ is shown as follows:

$$\begin{aligned}\nabla_{w_F} \mathcal{L}_\theta^{\text{CE}}(x_{w_F}, y) &= \frac{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F})) \cdot \nabla_{x_{w_F}} f_\theta^i(x_{w_F}) \cdot \nabla_{w_F} x_{w_F} - \nabla_{x_{w_F}} f_\theta^y(x_{w_F}) \cdot \nabla_{w_F} x_{w_F}}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))} \\ &= \frac{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F})) \nabla_{w_F} x_{w_F} (\nabla_{x_{w_F}} f_\theta^i(x_{w_F}) - \nabla_{x_{w_F}} f_\theta^y(x_{w_F}))}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))}\end{aligned}\quad (14)$$

Similarly, the gradient of $\mathcal{L}_\theta^S(x_{w_F}, y)$ is:

$$\nabla_{w_F} \mathcal{L}_\theta^S(x_{w_F}, y) = \frac{\sum_{i \neq y} \exp(f_\theta^i(x_{w_F})) \nabla_{w_F} x_{w_F} (\nabla_{x_{w_F}} f_\theta^i(x_{w_F}) - \nabla_{x_{w_F}} f_\theta^y(x_{w_F}))}{\sum_{i \neq y} \exp(f_\theta^i(x_{w_F}))}\quad (15)$$

Then we take the multiplication of $\nabla_{w_F} \mathcal{L}_\theta^S(x_{w_F}, y)$ by a term $\frac{\sum_{i \neq y}^K \exp(f_\theta^i(x_{w_F}))}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))}$, finally we attain:

$$\begin{aligned}
& \nabla_{w_F} \mathcal{L}_\theta^S(x_{w_F}, y) \cdot \frac{\sum_{i \neq y}^K \exp(f_\theta^i(x_{w_F}))}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))} \\
&= \frac{\sum_{i \neq y}^K \exp(f_\theta^i(x_{w_F})) \nabla_{w_F} x_{w_F} (\nabla_{x_{w_F}} f_\theta^i(x_{w_F}) - \nabla_{x_{w_F}} f_\theta^y(x_{w_F})) + 0}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))} \\
&= \frac{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F})) \nabla_{w_F} x_{w_F} (\nabla_{x_{w_F}} f_\theta^i(x_{w_F}) - \nabla_{x_{w_F}} f_\theta^y(x_{w_F}))}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))} \\
&= \nabla_{w_F} \mathcal{L}_\theta^{\text{CE}}(x_{w_F}, y)
\end{aligned} \tag{16}$$

Finally, we denote $\frac{\sum_{i \neq y}^K \exp(f_\theta^i(x_{w_F}))}{\sum_{i=1}^K \exp(f_\theta^i(x_{w_F}))}$ as $r(x_{w_F}, y)$. \square

A.3 DETAILS OF LOSS LANDSCAPE

We strictly follow the implementation from (Li et al., 2018) to achieve the desirable visualization of loss landscape of our integrated adversarial attacks with respect to all the differentiable parameters w . Specifically, we view w_F and w_{RT} as two parameterized filters, which is analogical to the ‘‘filter normalization’’ technique proposed by (Li et al., 2018). (Left) We adjust the initialization of variance of w , which then can provide a distant view of loss landscape before the optimization in Eq. 8. It exhibits a highly regular loss landscape, and its non-concavity w.r.t. only rotation and translation (Engstrom et al., 2019) has been tremendously improved. (Middle) We then provide a closer view of the loss landscape before the optimization and it shows a highly convex surface around the w to be optimized, facilitating the following optimization. (Right) After the optimization in Eq. 8 of our integrated spatial attack, we also present the loss landscape around the maxima w^* , exhibiting a highly concave surface as well.

A.4 A SKETCH MAP ABOUT THE RELATIONSHIPS BETWEEN SPATIAL AND SENSITIVITY-BASED ROBUSTNESS

As shown Figure 9, we set the spatial robustness as the X-axis and sensitivity-based robustness as the Y-axis, respectively. As the fundamental trade-off arising from the representation occurs between RT-based spatial robustness and PGD robustness, we place them in a perpendicular position. By contrast, flow-based robustness is set in the intersection of two axis because this kind of local spatial robustness is highly overlapped with sensitivity-based robustness. More importantly, we point out that the comprehensive adversarial robustness is between the two directions of both spatial and sensitivity-based adversarial robustness, which will be the more important research direction we need to adjust in the future.

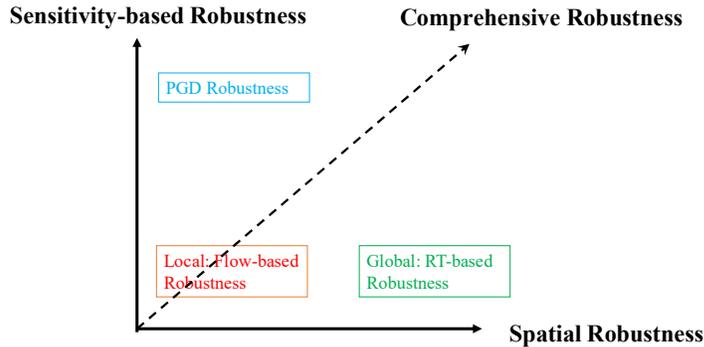


Figure 9: Relationships between spatial and sensitivity-based robustness.

A.5 BACKGROUND OF PARETO OPTIMIZATION

In the multi-objective optimization, we are normally required to involve multiple criteria or objective function to be optimized simultaneously, which has been applied in various fields in science, such as engineering and economics. Pareto optimization, serving as the key part in multi-objective optimization, can provide optimal solutions especially in the presence of multiple conflicting objectives.

Pareto optimization endeavors to achieve the Pareto efficient, a balanced situation between all objectives, where none of the objective functions can be improved in value without degrading some of the other objective values. In particular, in the pareto optimization, a Pareto front will be developed, consisting all the optimal solutions given some specific restraints. For instance, in finance, after the Pareto optimization, Efficient Frontier will be constructed to show the best combinations of risk and expected return that are available.

Based on the aforementioned background, a natural choice in order to balance the relationships between different adversarial robustness is to leverage Pareto optimization and construct a Pareto Front to describe the optimal combinations of available losses, which can provide a promising solution to address the comprehensive adversarial robustness issue.

A.6 DISCUSSION ABOUT MAX ADVERSARIAL TRAINING

We leverage Figure 6 to demonstrate that Max AT can degenerate to a single PGD adversarial training as the strength of PGD attack used in Max AT increases. In order to illustrate such conclusion, we draw the difference of robust accuracy between Max AT and single PGD adversarial training in Figure.6. It can be observed that as the attack for ϵ in PGD robustness in Max AT increases, the difference of three kinds of robust accuracy between Max AT and a single PGD AT tends to vanish.

Further, we consider the generalization issue based on different risks, and then set the risk in Max AT and Ave AT as $\mathcal{R}_{\max} = \max_i \mathcal{R}(f, S_i) = \max_i \mathcal{R}^{S_i}$ and $\mathcal{R}_{\text{ave}} = \frac{1}{m} \sum_{i=1}^m \mathcal{R}(f_{\theta}, S_i) = \frac{1}{m} \sum_{i=1}^m \mathcal{R}^{S_i}$. Proposition 2 illuminates that Max AT is closely associated with some form of Ave AT. This indicates that Max AT is likely to perform similarly with the specific form of Ave AT. Thus, adversarial training with linear combinations of various adversarial losses enjoys a more general form. Based on this form, an optimal combination has the potential to achieve better performance over Max AT and Ave AT. Our further proposed Pareto Adversarial Training provides strong evidence to support this intuition.

Proposition 2. *Given KKT differentiability and qualification conditions, $\exists \lambda_i \geq 0$, such that the risk minimizer in Max AT, i.e., \mathcal{R}_{\max}^* is a first-order stationary point of $\sum_{S_i \in \mathcal{S}} \lambda_i \mathcal{R}^{S_i}$ regardless of the relationship of S_i .*

Proof. Let f as the minimizer, e.g., the neural networks after the optimization:

$$\begin{aligned} f^* &\in \min_f \max_i \mathcal{R}(f, S_i) \\ M^* &= \max_i \mathcal{R}(f, S_i) \end{aligned} \tag{17}$$

Then the optimization can be equivalent to a constrained version:

$$\begin{aligned} \min_{f, M} \quad & M \\ \text{s.t.} \quad & \mathcal{R}(f, S_i) \leq M \text{ for all } S_i \in \mathcal{S} \end{aligned} \tag{18}$$

with Lagrangian $L(f, M, \lambda) = M + \sum_{S_i \in \mathcal{S}} \lambda_i (\mathcal{R}(f, S_i) - M)$. If this optimization problem satisfies KKT condition, then $\exists \lambda_i \geq 0$ with $\nabla_f L(f^*, M^*, \lambda) = 0$ such that

$$\nabla_f |_{f=f^*} \sum_{S_i \in \mathcal{S}} \lambda_i \mathcal{R}(f, S_i) = 0$$

□

Algorithm 1 Bi-level Optimization in Pareto Adversarial Training.

Input: Training data $(\mathcal{X}, \mathcal{Y})$. Batch size M and adjustable hyper-parameter r . Initialization of α as $[1/4, 1/4, 1/4, 1/4]$.

Output: Classifier f_θ .

- 1: **repeat**
- 2: Sample $\{\mathbf{y}_1, \dots, \mathbf{y}_M\}$ and $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ from all training data.
- 3: /* **Step 1: Compute loss in Eq. 11** */
- 4: Compute natural loss \mathcal{L}_{nat} , and adversarial loss $\mathcal{L}_{\text{PGD}}, \mathcal{L}_{\text{Flow}}, \mathcal{L}_{\text{RT}}$ based on natural cross entropy loss, PGD loss and Eq. 2 and 6, respectively.
- 5: /* **Step 2: Upper-level Optimization over θ** */
- 6: Given the current α , update f_θ by descending its stochastic gradient of:

$$\frac{1}{M} \sum_{i=1}^M \mathcal{L}^{\text{CE}}(f_\theta(\mathbf{x}_i), \mathbf{y}_i) = \frac{1}{M} \sum_{i=1}^M \alpha_0 \mathcal{L}_{\text{nat}} + \alpha_1 \mathcal{L}_{\text{PGD}} + \alpha_2 \mathcal{L}_{\text{Flow}} + \alpha_3 \mathcal{L}_{\text{RT}}$$
- 7: /* **Step 3: Lower-level Optimization over α** */
- 8: Compute $\hat{\mu}$ and $\hat{\Sigma}$ by sliding window technique.
- 9: Evaluate P in the quadratic form shown in Eq. 19.
- 10: Solve Eq. 19 via CVXOPT tool to obtain the α .
- 11: **until** Convergence

A.7 OPTIMIZATION ANALYSIS ON THE PARETO ADVERSARIAL TRAINING AND ALGORITHM

Our bi-level optimization within a batch is: (1) θ : SGD (2) α : quadratic programming. Denote the random variables $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ with mean vector μ and covariance matrix Σ . We transform our lower-level optimization regarding α as a standard quadratic form:

$$\begin{aligned} \min_{\theta, \alpha} \quad & \alpha^T P \alpha \\ \text{s.t.} \quad & \begin{bmatrix} 0 & \mu_1 & \mu_2 & \mu_3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \alpha = \begin{bmatrix} r \\ 1 \end{bmatrix} \\ & -\alpha \leq \mathbf{0} \end{aligned} \quad (19)$$

where $P = 8(\text{diag}(\Sigma) + \text{diag}(\mu\mu^T)) - 2(\Sigma + \mu\mu^T)$. We provide the proof of P in the following:

Proof.

$$\begin{aligned} & \sum_{i=0}^3 \sum_{j=0}^3 \mathbb{E}(\alpha_i \mathcal{L}_i - \alpha_j \mathcal{L}_j)^2 \\ &= \sum_{i=0}^3 \sum_{j=0}^3 \mathbb{E}((\alpha_i \mathcal{L}_i - \mathbb{E}(\alpha_i \mathcal{L}_i)) - (\alpha_j \mathcal{L}_j - \mathbb{E}(\alpha_j \mathcal{L}_j)) + (\mathbb{E}(\alpha_i \mathcal{L}_i) - \mathbb{E}(\alpha_j \mathcal{L}_j)))^2 \\ &= \sum_{i=0}^3 \sum_{j=0}^3 \mathbb{E}((\alpha_i \mathcal{L}_i - \mathbb{E}(\alpha_i \mathcal{L}_i) - (\alpha_j \mathcal{L}_j - \mathbb{E}(\alpha_j \mathcal{L}_j)))^2 + (\mathbb{E}(\alpha_i \mathcal{L}_i) - \mathbb{E}(\alpha_j \mathcal{L}_j))^2 + 0) \quad (20) \\ &= \sum_{i=0}^3 \sum_{j=0}^3 (\alpha_i^2 \sigma_{ii} + \alpha_j^2 \sigma_{jj} - 2\alpha_i \alpha_j \sigma_{ij}) + (\alpha_i^2 \mu_i^2 + \alpha_j^2 \mu_j^2 - 2\alpha_i \alpha_j \mu_i \mu_j) \\ &= 8\alpha^T \text{diag}(\Sigma) \alpha - 2\alpha^T \Sigma \alpha + 8\alpha^T \text{diag}(\mu\mu^T) \alpha - 2\alpha^T (\mu\mu^T) \alpha \\ &= \alpha^T (8(\text{diag}(\Sigma) + \text{diag}(\mu\mu^T)) - 2(\Sigma + \mu\mu^T)) \alpha \end{aligned}$$

□

Further, we provide the detailed description of Pareto Adversarial Training algorithm in Algorithm 1.

A.8 IMPLEMENTATION DETAILS ABOUT VARIOUS ATTACKS AND ADVERSARIAL TRAINING

For MNIST comparison, we trained the Simple CNN in (Zhang et al., 2019) on MNIST for 100 epochs. As for the CIFAR-10 dataset, we choose the widely used Pre-Act ResNet-18 with grouped normalization and trained the network for 76 epochs. The other details of our implementation on MNIST and CIFAR-10 are based on (Zhang et al., 2019), while the implementation on Caltech-256 has refer to (Zhang & Zhu, 2019) with 10 epoch to finetune a pretrained ResNet-18.

PGD Attack We apply the widely accepted setting on three datasets. We set step size as 0.01, ϵ as 0.3 on MNIST while the step size is 0.007 and ϵ is 0.031 on both CIFAR-10 and Caltech-256 datasets. To evaluate the different levels of robustness, we conduct the evaluation of PGD attack under 10, 20, 30, 40 iterations on MNIST, and 5, 10, 15, 20 iterations on CIFAR-10 and Caltech-256 datasets.

Flow-based and RT Attacks On MNIST, we set step size α_F and α_{RT} as 0.01 and 0.1, and choose $\epsilon_F, \epsilon_{RT}$ as 0.3. We select 5, 10, 15, 20 as the attack iterations for the evaluation of both two attacks. On CIFAR-10, we set step size α_F as $1e-3$ and α_{RT} as 0.05, and choose $\epsilon_F, \epsilon_{RT}$ as 0.3. We select 3, 5, 10, 15 as the attack iterations for the evaluation of both two attacks. On Caltech-256, we set step size α_F as $1e-5$ and α_{RT} as 0.1, and choose $\epsilon_F, \epsilon_{RT}$ as 0.3 and 1.0 for the two attacks, respectively. We select 3, 5, 10, 15 as the attack iterations for the evaluation of both two attacks.

PGD Adversarial Training We choose PGD iteration as 30, 3, 5 in the PGD adversarial training for the three datasets, plus the same settings as PGD attacks for each dataset respectively.

Integrated Spatial Adversarial Training Our integrated spatial adversarial training is based on our proposed integrated spatial attacks that unifies both Flow-based an RT-based attacks. We set the iterations as 20, 5, 10 and on MNIST, CIFAR-10 and Caltech-256, respectively. Other hyper-parameters are same as those in their corresponding attacks.

Pareto Adversarial Training We select a sequence of r to train multiple Pareto Adversarial training models. On MNIST, we choose r in $[0.2, 0.5, 0.8, 1.0, 1.2, 1.5, 1.8, 2.0, 2.2]$ and $[0.5, 1.0, 1.25, 1.5, 2.25, 3.0, 3.5, 4.0]$ on both CIFAR-10 and Caltech-256. Other parameters follow corresponding method above, respectively.