RESILIENCE TO MULTIPLE ATTACKS VIA ADVERSARIALLY TRAINED MIMO ENSEMBLES

Anonymous authors
Paper under double-blind review

ABSTRACT

While ensemble methods have been widely used for robustness against random perturbations (*i.e.,* the average case), ensemble approaches for robustness against adversarial perturbations (*i.e.,* the worst case) have remained elusive despite multiple prior attempts. We show that ensemble methods can improve adversarial robustness to multiple attacks if the ensemble is *adversarially diverse*, which is defined by two properties: 1) the sub-models are adversarially robust themselves and yet 2) adversarial attacks do not transfer easily between sub-models. While at first glance, creating such an ensemble would seem computationally expensive, we demonstrate that an adversarially diverse ensemble can be trained with minimal computational overhead via a Multiple-Input Multiple-Output (MIMO) model. Specifically, we propose to train a MIMO model with adversarial training (*MAT*), where each sub-model can be trained on a different attack type. When computing gradients for generating adversarial examples during training, we use the gradient with respect to the ensemble objective. This has a two-fold benefit: 1) it only requires 1 backward pass and 2) the cross-gradient information between the models promotes robustness against transferable attacks. We empirically demonstrate that *MAT* produces an ensemble of models that is adversarially diverse and significantly improves performance over single models or vanilla ensembles while being comparable to previous state-of-the-art methods. On MNIST, we obtain 99.5% clean accuracy and $(88.6\%, 57.1\%, 71.6\%)$ against $(\ell_\infty, \ell_2, \ell_1)$ attacks, and on CIFAR10, we achieve 79.7% clean accuracy and $(47.9\%, 61.8\%, 47.6\%)$ against $(\ell_\infty, \ell_2, \ell_1)$ attacks, which are comparable to previous state-of-the-art methods.

1 INTRODUCTION

The arms race between attacks and defenses against DNNs has created a diversity of attack types. These include black-box vs white-box attacks and gradient-based vs gradient-free attacks. Ideally, adversarially robust models would be robust against all types of attack. Many defenses have been developed to protect DNNs from adversarial attacks. For instance, gradient masking [23], input denoising [36, 17], and hidden layer randomization [18], are three techniques that in their time were heralded as strong defense techniques. However, [1, 3, 31] show that all of the above methods fail against adaptive adversaries, *i.e.,* perturbations crafted with full knowledge of the defense measure. Among all attempts to make the deep model resilient, Adversarial Training (AT) [19] has stood the test of time and is regarded as the gold standard for its resilience to white-box adversaries [20]. AT trains the deep models with adversarial examples instead of with benign training examples. Due to its empirical robustness, this method remains state-of-the-art against a wide range of attacks. However, AT suffers from two fundamental drawbacks. First, it is computationally expensive, specifically the method to generate effective AEs. Second, the choice of attack type used during training significantly affects the type of robustness that is observed. For example, previous work [30, 25] shows that the model adversarial trained using the ℓ_∞ attack is robust against ℓ_1 and ℓ_2 PGD attacks but vulnerable against gradient-free attacks. This phenomenon reveals that the robustness of adversarial training implicitly

masks out the gradient information for ℓ_1 and ℓ_2 attacks. Previous works [30, 20] try to revise adversarial training by training the model using the worst case of PGD adversarial examples among different attack metrics and achieve better results. MSD [20] revises the Tramèr and Boneh's method [30] by calculating $\ell_1, \ell_2, \ell_\infty$ attacks for *each step* of PGD iteration and updates with the one that achieves the largest loss. This method achieved better performance against $\ell_1, \ell_2, \ell_\infty$ attacks. However, though MSD is faster than [30], it still requires three forward passes for each step of the PGD attack, which is about 2.5 times the training time of standard AT. Furthermore, MSD compromises on the clean accuracy more than AT.

As ensemble methods have proven to be effective in robustness to random perturbations (*i.e.,*, the average case), one natural idea to improve adversarial robustness is to exploit ensemble methods (or model averaging). Ensemble methods combine the predictions of sub-models based on some vote functions (like averaging or majority vote). Previous attempts of using ensemble methods for adversarial robustness have been shown to be vulnerable under strong adversaries [37, 21, 26, 32, 12]. They mainly failed for two reasons. First, if the ensemble is diverse but the models are weak (*i.e.,* not adversarially robust), an adaptive attack only needs to find a *single* AE that lies in the intersection of the AE space of the sub-models (*i.e.,* a single worst case), which is easy [12]. Intuitively, this finds a single perturbation that is a "common mistake" across sub-models. Second, adversarial examples are very easy to transfer between similar models, *i.e.,* the adversarial perturbation generated from one sub-model may easily mislead another sub-model especially when they share the similar training set or architecture [22]; and the transferability of attacks remains true even if the sub-models are adversarially robust themselves (see vanilla ensemble models in 4.3 of the experiments). Thus, neither diversity of the models nor adversarial robustness of each of the sub-models by themselves can improve adversarial robustness.

To overcome these two prior failures of ensemble methods, we propose to encourage *adversarial diversity* which is defined to have two key features simultaneously: 1) adversarial robustness of each sub-model and 2) robustness against transfer attacks between sub-models. Additionally, we would like to do this with minimal computational overhead because AT is already very computationally expensive. Our approach is inspired by a recent work called multi-input-multi-output (MIMO) training [10]. MIMO can train M sub-models simultaneously to form a diverse ensemble, and crucially the training time of the ensemble is almost the same as that of a single model. In this work, we show that by combining MIMO configuration with AT, it is possible to attain robustness against multiple attack types with an adversarially diverse ensemble, i.e., we can achieve adversarial diversity. The results in § 4.1 and § 4.2 show that by sharing model parameters, the sub-models act as an implicit way of bringing adversarial diversity while also remaining robust. Then in § 4.3, we present that our *MAT* adversarially trained with each sub-model bestowing resilience to a particular attack metric leads to robustness against multiple types of perturbations. We achieve robustness accuracy $(88.6\%, 57.1\%, 71.6\%)$ against $(\ell_\infty, \ell_2, \ell_1)$ attack with epsilon $(10, 2, 0.3)$ on MNIST. On CIFAR10 dataset, our model achieves robustness accuracy $(47.9\%, 61.8\%, 47.6\%)$ against $(\ell_1, \ell_2, \ell_\infty)$ attack with radius $(12, 0.5, 0.03)$. Finally, in § 4.4, we provide an alternative application that one can utilize our MIMO+AT (*MAT*) by training the *MAT* with all sub-models trained with the same attack to strengthen previous AT methods to achieve better adversarial robustness without loss of clean accuracy. We provide three examples in this section as ℓ_∞ AT, ℓ_2 AT, and MSD. We summarize our contributions as follows:

1. We shed light on why prior ensemble methods may fail and hypothesize that ensemble methods can improve adversarial robustness if they are *adversarially diverse*, which is defined by two properties: 1) the sub-models are adversarially robust themselves and yet 2) adversarial attacks do not transfer easily between sub-models. **2.**We propose to use MIMO AT (*MAT*) to train such an ensemble in a computationally efficient way and discuss how the cross-gradient information between the sub-models during training can increase the robustness to transfer attacks between models. **3.** We empirically demonstrate that *MAT* is adversarially diverse and improves significantly over AT models and vanilla ensembles while being comparable or better than prior state-of-the-art methods for multi-attack robustness. Code and pre-trained models for reproducing our experiments have been uploaded to the anonymized repository `https://github.com/anonymous-lab-ml/mimo-adv/`.

2 PRELIMINARY

In this section, we review some key concepts related to adversarial examples (AEs), adversarial training (AT), and MIMO structure, which will be used through the rest of paper.

2.1 ADVERSARIAL ATTACKS

PGD Attack. These attacks seek a point x' in a neighborhood of the benign example x with perturbation δ, which maximizes the loss $\mathcal{L}$. AE x is generated as: $x' = x + \arg\max_{\delta \in \mathcal{B}_{\|\cdot\|_p}(\epsilon)} \mathcal{L}(f_\theta(x+\delta), y^*)$, where ϵ denotes the attack radius, f_θ is the DNN with parameter θ, y^* is the true label. A general PGD attack adopts gradient-type algorithm to iterate on the perturbation δ as the the following

$$\delta^{(t+1)} = \Pi_{\mathcal{B}_{\|\cdot\|_p}(\epsilon)} \left(\delta^{(t)} + \alpha \cdot \mathcal{V}_p \left(\nabla_x \mathcal{L}(f_\theta(x), y^*) \right) \right), \quad \text{with} \quad \mathcal{V}_p(\delta) = \underset{v \in \mathcal{B}_{\|\cdot\|_p}(1)}{\arg\max} \; v^\top \delta. \tag{1}$$

where p is the attack metric (usually ℓ_p norm), α is the step size, Π is the operator projecting the gradient onto the ℓ_p ball with radius ϵ (denoted by $\mathcal{B}_{\|\cdot\|_p}(\epsilon)$), $\mathcal{V}_p$ is the operator mapping the gradient to the steepest descent direction on the ℓ_p ball with radius 1. Note that $\mathcal{V}_p$ is a fundamentally different operator than Π, and in the case of the $p = \infty$, $\mathcal{V}_\infty$ reduces to the sign of the gradient (as in the FGSM attack). Intuitively, PGD adversarial attack uses iterate with a fixed step size α along the steepest direction $\mathcal{V}_p \left(\nabla_x \mathcal{L}(f_\theta(x), y^*) \right)$. FGSM [9], BIM [15] and [19] can all be treated as variants of the PGD Attack.

2.2 ADVERSARIAL TRAINING

Adversarial Training (AT) is regarded as one of the most practical yet effective methods for defending against adversarial attacks. This has stood the “test of time" through multiple iterations of proposed defenses and attacks that discover vulnerabilities in them. AT seeks to build a robust model against adversarial attacks by producing AEs and injecting them into training data by solving a saddle point optimization problem:

$$\theta^* = \arg\min_\theta \sum_i \max_{\delta_i \in \mathcal{B}_{\|\cdot\|_p}(\epsilon)} \mathcal{L}\left(f_\theta(x_i + \delta_i), y_i^* \right). \tag{2}$$

Here i denote the i-th batch number of training set (x_i, y_i^*). The inner maximization seeks to find the adversarial directions δ_i given the current classifier f_θ, and the outer minimization tries to find the best θ to minimize the summation of all batches' loss. PGD AT is widely accepted as a standard approximation of inner maximization methods since it is a universal first-order attack algorithm [19]. Though it roughly needs $K+1$ times more computation compared to standard training, PGD-AT remains the de facto standard for AT since it walks the balance between strength of attack and computational cost of generating AEs. The drawback of AT is that solving the inner maximization requires a pre-defined attack metric p, which only contributes to robustness under ℓ_p attacks. [30]. In this paper, we show how the adversarial robustness can be achieved through AT but crucially, one can generalize the defense to multiple attack types and achieve this at about the exact cost as AT.

2.3 MIMO ENSEMBLE

Ensemble methods have been widely used to improve a model's calibration and robustness over out-of-distribution samples by averaging over multiple neural network predictions [10, 34, 35]. However, ensemble over M deep models (sub-models) is usually M times more expensive from a computational and a memory standpoint. Furthermore, to improve uncertainty estimation, sub-models often sacrifice individual performance to achieve diversified predictions.

To address the computational drawbacks of vanilla ensembles, [10] proposed an ensemble structure called MIMO Ensemble. It showed a surprising result: the benefits of using multiple predictions can be achieved 'for free' under a single model's forward pass. In particular, they showed that, using a multi-input multi-output (MIMO) configuration, one can utilize a single model's capacity to train multiple sub-models that independently learn the task at hand. Since all the layers are shared except for the input layer and the output layer, this architecture uses only about 1% more parameters. Training a MIMO model has almost the same time complexity as training a standard model while keeping the sub-models diverse without any special regularization and parameter tuning. During training, MIMO uses gradient method to solve following optimization problem for M sub-models.

$$\underset{\theta}{\text{minimize}} \sum_{i} \sum_{m=1}^{M} \mathcal{L}(\underbrace{f_\theta(x_{i,1}, ..., x_{i,M})_m}_{y_{i,m}}, y^*_{i,m}), \tag{3}$$

where $(x_{i,1}, ...x_{i,M})$ are i.i.d sampled from the training set, $m \in \{1, \ldots, M\}$ denotes the m-th subnetwork, $x_{i,m}$ denotes the input of the i-th batch for the m-th subnetwork, $y_{i,m}, y^*_{i,m}$ represent the outputs from the model and the ground truth respectively, and $\mathcal{L}$ is the cross entropy loss. Notice that the overall objective is the summation of each sub-model's loss. The separation structure promotes the sub-models' prediction to be less correlated because the output y_j is independent to x_i when $i \neq j$. During testing, the same inputs are copied and fed to all the M sub-models, then MIMO decides the prediction by averaging $y_{i,m}$.

3 Our Approach: *MAT*

Given the weaknesses of prior ensemble methods, we seek an algorithm that produces an *adversarially diverse* ensemble. We define *adversarial diversity* as an ensemble where 1) the sub-models are adversarially robust themselves and 2) the sub-models are robust to transfer attacks from other sub-models. To achieve this, we propose to combine MIMO and AT, an approach we call *MAT*.

There are two questions to adversarially training an ensemble: 1) What loss function should be used to update the sub-model parameters (e.g., the loss for the single sub-model or the ensemble loss)? and 2) How should adversarial examples be created during training? For the first question, one could just update each sub-model independently using the loss of each sub-model—as is often done in standard ensemble methods. However, the prior MIMO work [10] suggests that taking the gradient with respect to the ensemble loss is actually better for clean accuracy (as well as being more computationally efficient). Thus, we adopt this. The second question about which loss to use for creating adversarial examples during training is a bit trickier. Naïvely, it would seem that gradients with respect to the sub-models' loss would be the best as in standard AT. However, we will discuss how gradients with respect to the *total* loss across sub-models actually provides a key benefit of making the models more robust to transfer attacks while also being computationally efficient (which was their original purpose).

3.1 Naïve MIMO Adversarial Training

The foundation of *MAT* is our M-MIMO adversarial training method. We want to train an ensemble that is adversarially diverse. To encourage diversity, we can use different attack types for each sub-model. For this, we wish to learn network parameters that minimize the sum of losses to the perturbations introduced by each attack. As is common in the literature, and without loss of generality, we will use three PGD attacks ℓ_1, ℓ_2, and ℓ_∞ to instantiate our network. As a naïve approach, we may aim to solve the following problem

$$\min_{\theta} \sum_{i} \sum_{m=1}^{M} \max_{\delta_{i,m} \in \mathcal{B}_{\|\cdot\|_{p_m}}(\epsilon_m)} \mathcal{L}(\underbrace{f_\theta(x_{i,1} + \delta_{i,1}, ..., x_{i,M} + \delta_{i,M})_m}_{y'_{i,m}}, y^*_{i,m}). \tag{4}$$

For each batch i, the inner maximization seeks to find the worst-case adversarial perturbations $\delta_{i,m}$ given the current classifier f_θ. Notice that each subnetwork m finds its own perturbation $\delta_{i,m}$ inside different balls $\mathcal{B}_{\|\cdot\|_{p_m}}(\epsilon_m)$.

The inner maximization, if solved by the K-step PGD, for each subnetwork m the perturbation is updated at iteration $k+1$ as follows:

$$\delta_{i,m}^{(k+1)} = \Pi_{\mathcal{B}_{\|\cdot\|_{p_m}}(\epsilon_m)} \left(\delta_{i,m}^{(k)} + \alpha_m \cdot \mathcal{V}\left(\nabla_{x_{i,m}} \mathcal{L}(y'_{i,m}, y^*_{i,m}) \right) \right). \tag{5}$$

The outer minimization is general SGD. Notice that the difference of equation 5 and vanilla AT perturbation direction equation 1 lies in that equation 5 allows for different projections for different attack types depending on $\mathcal{B}_{\|\cdot\|_{p_m}}$, which results in potential robustness over diverse attacks.

3.2 Ensemble Adversarial Training (EAT)

While the naïve approach may produce models that are adversarially robust by themselves, it does not encourage robustness to transfer attacks between sub-models and additionally, it is computationally expensive since it requires multiple backward passes (M) for each loss function.

Therefore, to further improve the adversarial diversity *MAT*, and also reduce the computation burden, we propose the ensemble adversarial training (*EAT*) method. The standard adversarial training, with respect to each sub-model, shown in Equation 2, solves the inner maximization by calculating the gradient of its own loss to the input x. In our *MAT* setting, this requires m backward propagations. The one step of inner maximization is given by

$$\delta_{i,m}^{(k+1)} = \Pi_{\mathcal{B}_{\|\cdot\|_{p_m}}(\epsilon_m)} \left(\delta_{i,m}^{(k)} + \alpha_m \cdot \mathcal{V}_{p_m} \left(\nabla_{x_m} \sum_{m=1}^{M} \mathcal{L}(y'_{i,m}, y^*_{i,m}) \right) \right). \tag{6}$$

For a single batch i and sub-model j, *EAT* approximates the true gradient inside PGD attack up to a gap, which is defined as *cross gradients* as follows. We introduce two properties of the cross gradients.

$$\textit{cross gradients}(i,j) \triangleq \nabla_{x_{i,j}} \sum_{m \neq j} \mathcal{L}(y'_{i,m}, y^*_{i,m}) = \underbrace{\nabla_{x_{i,j}} \sum_{m=1}^{M} \mathcal{L}(y'_{i,m}, y^*_{i,m})}_{\text{approximate gradient}} - \underbrace{\nabla_{x_{i,j}} \mathcal{L}(y'_{i,j}, y^*_{i,j})}_{\text{true gradient}}. \tag{7}$$

Preventing adversarial examples crossing the models. These cross gradients describe the change of y_j w.r.t the change of x_i locally, where $i \neq j$. Though the outer minimization weakens the correlation between the sub-models as we mentioned in Section 2.3, there's a possibility that the adversarial perturbation of x_i could flip the j-th sub-model. Maximizing over the cross gradients produces adversarial perturbation, namely, *cross adversarial examples*. Introducing this extra term during training reduces the worst-case correlation between sub-models, which contributes to robustness to the cross AES.

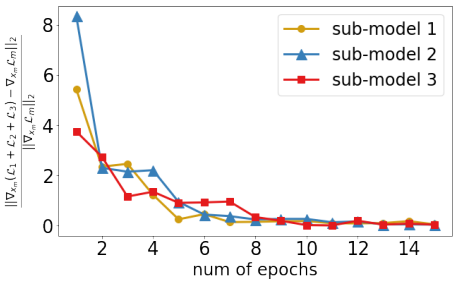

Figure 1: It shows the cross gradients $(\cdot, j)$ ***converge to zero quickly, where*** $j = 1, 2, 3$ ***each sub-model*** j ***equipped with*** $\ell_1, \ell_2,$ ***and*** ℓ_∞ ***AT respectively.***

Convergence with training. MIMO standard training shows that their design of objective function (equation 3) induces the separation of sub-models, *i.e.,* the sub-model i will have little impact on the output of sub-model j ($i \neq j$). This separation structure explains why our method works. It can be validated by Fig. 1, where we observed the cross gradients approaching 0 with more epochs of training. Notice that the final cross gradient is not strictly 0, showing that there are still weak correlation between sub-models.

4 EVALUATION

In § 4.1, we present that the sub-models trained by our *MAT* are diverse, and in § 4.2, we show that each sub-model is robust against adversarial examples. Then in § 4.3, we present experimental results on *MAT* versus state-of-the-art multi-robustness defenses for a variety of attacks. We show that *MAT* equipped with robust and diverse sub-models achieves state-of-the-art robustness against multiple types of adversarial attacks on the MNIST and CIFAR10. Notice that our method only requires $K+1$ forward and backward pass for each batch during training which is comparable to standard AT. (Time complexity analysis is in the appendix B). Next, in § 4.4, we implement our *MAT* with three sub-models trained with same adversarial attack. Our results show that *MAT* could bring better robustness without any trade-off in time and clean accuracy.

Table 1: Adversarial Transferability of two independent AT models

	model 1	model 2
model 1	216	91.2%
model 2	96.4%	199

***Table 2: Adversarial Transferability of* MAT**

ℓ_2 attack	sub 1	sub 2	sub 3
sub-model 1	334	49.1%	64.1%
sub-model 2	71.8%	270	79.2%
sub-model 3	27.5%	19.4%	298

Note: *Table at position (i, j) means that we generate adversarial examples from sub-model i and use it to attack sub-model j; at position (i, i) means the valid number of adversarial examples per 1000 examples.*

4.1 DIVERSITY: ADVERSARIAL TRANSFERABILITY AMONG SUB-MODELS

In this section, we observe how the ensemble sub-models perform against adversarial attacks. To better understand our *MAT* method. we evaluate *MAT* against ℓ_2 adversarial attacks. The results (Tables 1 and 2) show that our model trained using *MAT* not only gains multiple robustness by the ensemble adversarial training (EAT), but the sub-models are diverse enough to prevent effective adversarial transferability. A group of classifiers could benefit from the ensemble if and only if each member is diverse enough. There are several different measures trying to quantify the property [14, 37]. In this experiment, we use standard AT to independently train two models with the exact same network architecture as well as the hyperparameters. We test the success rate of adversarial examples generated in model 1 against the model 2. Among the first 1000 test examples for model 1, the adversary succeeds on 216 examples. We use these examples to attack model 2, and 197 of them succeed on model 2, which means the rate of transferability is 91.2%. Table 2 shows the transferability of *MAT*, where sub-models 1, 2, and 3 have been trained against respectively ℓ_1, ℓ_2, and ℓ_∞ PGD attacks. We notice that the transferability rate is much lower compared to vanilla ensembles. The adversarial examples generated in sub-model 3 only have 27.5% success on sub-model 1. Even in the worst case, out of the 270 adversarial examples generated for sub-model 2, 52 examples failed to transfer to sub-model 3. Notice we show the result here for the ℓ_2 PGD attack as that brings out the worst case transferability of *MAT*.

Table 3: Robustness accuracy of sub-models

	MAT	sub 1	sub 2	sub 3
ℓ_∞ PGD	89.2%	85.0%	33.1%	87.1%
ℓ_2 PGD	66.6%	63.7%	69.0%	61.0%
ℓ_1 PGD	71.9%	72.0%	71.4%	67.8%

4.2 ROBUSTNESS OF SUB-MODELS

In this section, we empirically show that our sub-models are robust against $\ell_\infty, \ell_2, \ell_1$ attacks. The results are summarized in Table 3. We observe that using *MAT*, each sub-model is relatively robust against all adversarial examples. Combining the fact that the adversarial examples are harder to transfer between these sub-models as shown above, *MAT* achieves the desideratum of adversaral diversity. This includes the property that the overall network achieves better performance compared to the sub-models. The only weak sub-model is sub-model 2, which shows vulnerability against ℓ_∞ PGD attack, though, the ensemble classifier still performs better than any individual AT model.

Table 4: Main results on MNIST

	standard training	ℓ_∞	ℓ_2	ℓ_1	Vanilla Ensemble	MSD	MAT (ours)
clean accuracy	99.5%	99.3%	98.6%	98.9%	99.0%	98.5%	**99.5%**
PGD ℓ_∞ attack	0.0%	96.0%	0.8%	0.0%	4.6%	62.2%	89.2%
HSJ attack	12.3%	98.0%	44.2%	12.1%	35.7%	87.4%	91.2%
All ℓ_∞ attacks	0.0%	96.0%	0.8%	0%	2.6%	62.2%	88.6%
PGD ℓ_2 attack	0.0%	78.1%	76.8%	11.4%	57.2%	72.0%	66.6%
C&W attack	0.0%	39.7%	76.8%	12.6%	60.1%	69.8%	59.4%
Boundary attack	0.7%	17.7%	78.2%	21.2%	46.9%	70.6%	77.8%
All ℓ_2 attacks	0.0%	8.7%	74.4%	9.8%	44.8%	67.3%	57.1%
PGD ℓ_1 attack	0.0%	53.3%	79.0%	19.9%	63.2%	63.7%	71.9%
Salt & Pepper attack	74.3%	64.9%	96.2%	72.5%	68.4%	84.5%	96.0%
All ℓ_1 attacks	0.0%	53.3%	78.2%	19.9%	60.1%	62.7%	71.6%
All attacks	0.0%	6.5%	0.8%	0.0%	1.5%	54.7%	51.7%

4.3 MAIN RESULT

In this section, we present the main results of *MAT* against various types of attacks. PGD ℓ_∞ attack [19], PGD ℓ_2 attack, sparse PGD ℓ_1 attack [30], and Carlini & Wagner ℓ_2 [5] attacks are four types of gradient-based attacks. We also test the gradient-masking effect of our model by evaluating the robustness performance against gradient-free attacks, namely HopSkipJump attack (HSJ) [6], Boundary attack [2], and Salt and Pepper Noise attack [24]. We also report the robustness over the union of multiple attacks, i.e., all ℓ_∞, ℓ_2, ℓ_1 attacks. We compare *MAT* with several baseline models. First, we have three AT models with attack metric ℓ_∞, ℓ_2, and ℓ_1, respectively. We also evaluate the vanilla ensemble of these three models. We also compare our method with MSD [20], the state-of-the-art defense against multiple perturbations. Our main results on MNIST and CIFAR10 datasets are summarized in Table 4 and Table 5. *MAT* significantly boosts all the previous methods without extra cost (all the models here share the same hyperparameters and the network structure). For ℓ_∞ PGD training, *MAT* achieves 95.6% accuracy outperforming AT by 0.7%. For ℓ_2 PGD AT, our methods also increasing the robust accuracy against ℓ_2 attacks by 1.3%. We also test *MAT* coupled with MSD. The results show that on ℓ_∞ attacks and ℓ_1 attacks, we significantly boost the MSD method by 20.2% and 28.2% respectively. The drawback here is that our method has lower accuracy against ℓ_2 based attack, which requires further hyperparameter tuning to favor the ℓ_2 attack relative to the others. The results show

Table 5: Main results on CIFAR10

	standard training	ℓ_∞	ℓ_2	ℓ_1	Vanilla Ensemble	MSD	MAT (ours)
clean accuracy	95.3%	84.9%	89.3%	93.6%	84.9%	81.1%	78.9%
PGD ℓ_∞ attack	0.0%	48.8%	0.0%	0.0%	0.0%	48.0%	47.9%
HSJ attack	7.5%	79.3%	69.0%	12.9%	67.3%	72.4%	74.7%
All ℓ_∞ attacks	0.0%	46.7%	0.0%	0.0%	0.0%	48.0%	47.9%
PGD ℓ_2 attack	4.3%	61.1%	79.5%	0.0%	61.2%	68.2%	64.5%
C&W attack	3.4%	60.6%	79.3%	0.0%	57.5%	64.7%	63.3%
Boundary attack	25.2%	19.4%	82.1%	1.6%	23.2%	69.2%	64.2%
All ℓ_2 attacks	1.9%	18.2%	73.3%	0.0%	19.1%	64.3%	61.8%
PGD ℓ_1 attack	13.1%	18.2%	34.0%	0.0%	14.2%	53.4%	47.6%
Salt & Pepper attack	53.5%	65.7%	75.4%	61.3%	65.6%	73.9%	75.0%
All ℓ_1 attacks	12.8%	18.0%	33.3%	0.0%	14.2%	53.4%	47.6%
All attacks	0.0%	16.2%	0.0%	0.0%	0.0%	47.0%	46.4%

our robustness against various of attacks among three types of attack metric, *i.e.,*, ℓ_∞, ℓ_2, and ℓ_1 attacks. Our results reveal an ensemble of strong models could benefit the overall model's robustness. We further compare our method with the vanilla ensemble with three sub-models trained separately by $\ell_1, \ell_2, \ell_\infty$ PGD attacks on ResNet18. The vanilla ensemble shows its vulnerability to all three kinds of adversarial attacks though it

combines a three times larger model and requires three times longer training. *MAT* achieves 88.6% against the union of ℓ_∞ attacks on the MNIST dataset, which surpasses the previous state-of-the-art by 26.4% percent. For defending against the union of adversarial attacks, our method attains 51.7% robustness accuracy. On the CIFAR10 dataset, we also achieve strong robustness against multiple kinds of attack. Crucially, the vanilla ensemble fails against the ℓ_∞ attacks while *MAT* is competitive against the state-of-the-art MSD, while being computationally more efficient ($K+1$ forward passes for *MAT* versus $(1+M)K+1$ for MSD, with the same number of backward passes $K+1$).

Figure 2 shows the robustness curve our model and baseline models against PGD attack with different attack strengths. The key takeaway is that if internally within *MAT*, we perform MSD at each iteration to select the most damaging attack, this gives us greater adversarial robustness than the original MSD. However, this removes the computational advantage *MAT* had over MSD.

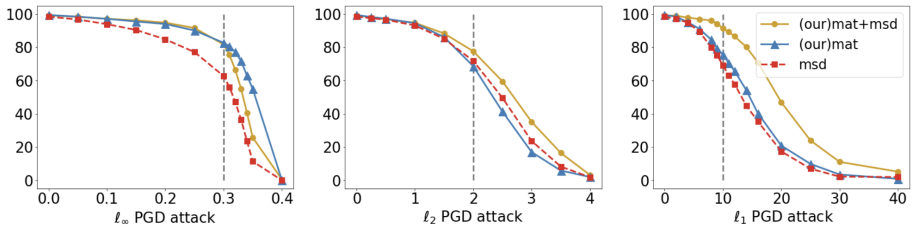

Figure 2: Test accuracy curves of* MAT *+MSD,* MAT *and MSD on MNIST under PGD ℓ_∞, ℓ_2, and ℓ_1 attacks with varying attack radius.

4.4 SUB-MODEL TRAINED WITH SAME ADVERSARIAL ATTACK

As mentioned, *MAT* can also be integrated with other AT techniques to strengthen the model. In this section, we empirically show the three experiments where we combine *MAT* with three previous methods, ℓ_∞ AT, ℓ_2 AT, and MSD respectively. The result in Table 6 shows that *MAT* achieves higher robustness accuracy in most scenarios even against the attack type that AT approach has been designed for. . Notice that all these models are trained with the same models and hyperparameters. As we observe, for MSD+MAT, we also bring 20.4% extra robustness against ℓ_∞ attacks, and 29.2% more robustness accuracy against ℓ_1 attacks with 0.8% extra clean accuracy. The drawback is we trade off 27% robust accuracy on ℓ_2 attacks. This is due to the imbalance of choices of the hyperparameters giving relative weight to the three attack types ℓ_1, ℓ_2, and ℓ_∞. .

Table 6: Sub-models trained with same Adversarial Attacks on MNIST

MAT + ℓ_∞ AT

	ℓ_∞	MAT+ℓ_∞
clean accuracy	**99.3%**	99.2%
PGD-ℓ_∞ Attack	96.0%	**96.1%**
HSJ Attack	**98.0%**	97.8%
AutoAttack	94.9%	**95.6%**
All ℓ_∞ Attacks	94.9%	**95.6%**

MAT + ℓ_2 AT

	ℓ_2	MAT+ℓ_2
clean accuracy	**98.6%**	98.5%
PGD-ℓ_2 Attack	**76.8%**	76.4%
C & W Attack	76.8%	**77.2%**
Boundary Attack	78.2%	**79.0%**
All ℓ_2 Attacks	74.4%	**75.7%**

MAT + MSD

	MSD	MAT+MSD
clean accuracy	98.5%	**99.3%**
PGD-ℓ_∞ Attack	62.2%	**82.4%**
HSJ Attack	87.4%	**97.3%**
All ℓ_∞ Attacks	62.2%	**82.4%**
PGD-ℓ_2 Attack	72.0%	**77.7%**
C & W Attack	**69.8%**	62.9%
Boundary Attack	**70.6%**	55.0%
All ℓ_2 Attacks	**67.3%**	39.7%
PGD-ℓ_1 Attack	63.7%	**93.7%**
Salt & Pepper Attack	84.5%	**94.9%**
All ℓ_1 Attacks	62.7%	**91.9%**
All Attacks	**54.7%**	39.5%

5 RELATED WORK

Adversarial Attacks. In this paper, we mainly focus on white-box attacks, the strongest kind against deep neural architectures [4]. FGSM [9] introduces a one-step attack which is fast, and addresses the demands of tasks where large amounts of AEs are needed. PGD [19] is an iterative version of FGSM, which is well balanced between computation and performance. PGD attacks are widely used to evaluate DNN models' robustness. C&W attack [5] employs an optimization that minimizes the margin loss f and distance measured by ℓ_p norm. Generally, FGSM attacks are not powerful enough to provide good robustness in AT, whereas C&W attacks are far too computationally expensive to use in AT. Thus, like most others, we use PGD for generating AEs during AT.

Adversarial Defense. Many defenses have been proposed, with most of them being broken by adaptive attacks [1, 3, 31]. Adversarial Training [19], though expensive in computation, has emerged as the best practical way to create models resilient to white-box attacks. A slew of literature has developed variants of AT differing in the way samples are created [8], the way the models are iterated upon [27, 33], and the way different attacks are combined [30, 20]. Our ensemble method is orthogonal to these approaches and therefore can be combined with these, with varying levels of additional design and engineering effort, such as to design the right combining function across the sub-models.

Adversarial Defense against multiple types of perturbation. [20] tries to build a robust model against multiple perturbations by introducing multiple steepest direction PGD attacks during training; their method ensembles three different attacks into one, called *multi steepest direction*. Our method instead seeks an ensemble over *models*. [25] introduces a multiple variational autoencoder to build a model robust to either ℓ_0 and ℓ_2 or ℓ_0 and ℓ_∞. However their method is not easy to apply to larger datasets. [29] proposed a provable defense against all ℓ_p norm with robust radius $3 - 5$ times smaller than state-of-the-art heuristic defense methods [20]. [16] trains DNNs against imperceptible adversarial examples, which are approximated using DNNs. While it shows robustness against a wide range of attacks, its performance against specific attacks is limited. Our approach *MAT* outperforms prior solutions in this space, either as we have proven empirically for the latest approach [20] or through the author-published numbers on the same dataset.

6 CONCLUSION AND DISCUSSION

In this paper, we present an adversarially diverse ensemble method called *MAT* in which each sub-model achieves both robustness and diversity. We develop the idea of learning an ensemble via adversarial training (AT) and adapt the structure of a recently discovered MIMO architecture. With our optimization technique called ensemble adversarial training (EAT), we require the same number of forward and backward passes as AT and yet achieve generalized robustness against multiple attack types.

MAT achieves state-of-the-art adversarial robustness and outperforms MIMO (expectedly) [10] and vanilla ensemble of AT, where MIMO shows diversity but not robustness, and vanilla ensemble of AT is robust but not diverse. Furthermore, for defending multiple adversarial examples, we achieve state-of-the-art performance which is comparable to MSD [20], while being computationally superior.

A drawback of our scheme is that if the network size is constrained, it may not have the capacity to accommodate multiple sub-models. Another drawback that is common to most AT methods is that the AE perturbation hyperparameters must be manually selected (we simply use the parameters from MSD [20]). Methods for automatically selecting the best hyperparameters using zero-th order optimization would broaden the applicability of our approach.

REPRODUCIBILITY STATEMENT

We have publicly released all the code, the datasets and models used, along with the implementation of the attacks. To help with the reproducibility of the experiments, we make available all of our code, the trained models, the training hyperparameter values, and the raw results from our testing. We point to the exact attacks that we have used (which are from existing codebases) in our evaluation. These are all to be found at the anonymized Github link `https://github.com/anonymous-lab-ml/mimo-adv/`.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 10–15 Jul 2018.

[2] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[3] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[4] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[6] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. HopSkipJumpAttack: a query-efficient decision-based adversarial attack. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.

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

[8] Jinhao Dong and Tong Lin. Margingan: Adversarial training in semi-supervised learning. *Advances in neural information processing systems*, 32:10440–10449, 2019.

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In International Conference on Learning Representations, 2015*, 2015.

[10] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021.

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

[12] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC, August 2017. USENIX Association.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017.

[16] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021.

[17] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[18] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[20] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6640–6650, 2020.

[21] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

[22] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[23] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017.

[24] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models, 2018.

[25] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

[26] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. *arXiv preprint arXiv:2004.10162*, 2020.

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

[28] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[29] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9155–9166. PMLR, 13–18 Jul 2020.

[30] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[31] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1633–1645. Curran Associates, Inc., 2020.

[32] Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32:8646–8656, 2019.

[33] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, pages 6586–6595. PMLR, 2019.

[34] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *International Conference on Learning Representations*, 2020.

[35] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 2020.

[36] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[37] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2019.

A CODE AND PSEUDO CODE OF *MAT*

A.1 CODE

Code for reproducing our experiments have been uploaded to the anonymized repository: `https://github.com/anonymous-lab-ml/mimo-adv/`

A.2 PESUDO CODE

Algorithm 1: MIMO adversarial training with approximate gradient (inner maximization)

```
input  : network f_θ, data_size N, batch_size B, data x, label y*, pgd attack steps: K,
         num_of_subnetwork: M
initialization;
/* iterate over all training examples                                    */
for i = 0 to ceil(N/B) do
    x_adv ← maximize(f_θ, x[i], y[i]);
    θ ← minimize(f_θ, x_adv, y[i]);
end

def maximize (f_θ, x, y*):
    adv = x ;                                          // adv = (adv_1, ··· , adv_M)
    for k = 1 to K do
        y_1, ..., y_M = f_θ(adv_1, ··· , adv_M);
        loss = 0;
        for m = 1 to M do
            loss ← loss + CELoss(y_m, y*_m);
        end
        grad = ∇_adv loss ;
        ;                                              // approximate gradient step
        for m = 1 to M do
            adv_m ← adv_m + Π_{B_{||·||_{p_m}(ε_m)}} (α_m · V_m(grad));
        end
    end
    return adv;

def minimize (f_θ, x, y*):
    loss = 0;
    y_1, ..., y_M = f_θ(x_1, ··· , x_M);
    for m = 1 to M do
        loss ← max(loss, CELoss(y_m, y*_m));
    end
    grad = ∇_θ(loss);
    θ ← θ − s · grad;
    return θ
```

B TIME COMPLEXITY OF DIFFERENT AT METHODS

Compared to standard training (*i.e.,* without ensemble), vanilla AT using K-step PGD attack requires $K+1$ times more computation, due to the K forward and backward passes individually for generating each AE plus the outer minimization. Furthermore, if we try to use M-vanilla ensemble AT to promote robustness

over multiple attacks, it will require M times more computation compared to vanilla AT. This becomes infeasible in practice, *e.g.,* for CIFAR-10 with ResNet-50, 100 epochs. It takes 40 times of training time to train one single model compared to standard training. To reduce the computation, we have the insight to replace the vanilla ensemble with a MIMO architecture and to train for all the M attack types within the same network, each within its own subnetwork. This allows us to calculate all subnetworks' outputs $y_i, i = 1, \dots, M$ in one forward pass instead of M passes, *i.e.,* $(y_1, \dots, y_M) = f_\theta(x_1, \dots, x_M)$, instead of $(y_1, \dots, y_M) = (f_{\theta_1}(x_1), \cdots, f_{\theta_M}(x_M))$ in case of the M vanilla ensemble. This approximation works because we can fit M subnetworks within the original network, *i.e.,* without needing to expand the size of the original network. This is because of the residual capacity in even reasonably sized networks (like ResNet-50 in our case) is sufficient.

Thus, M-MIMO AT reduces the number of forward passes to be similar to that of vanilla AT training. However, for each subnetwork, one iteration of PGD attack still needs one backward pass, which leaves the computation of backward pass almost as high as that of naïve ensemble with M networks. The computation cost per iteration of different methods is summarized in Table 7.

Table 7: Computation comparison between different methods (for each batch of data,* i.e., *each iteration of the outer minimization)

Methods	Forward passes	Backwards passes
Standard Training	1	1
Standard MIMO Training [10]	1	1
Vanilla AT (K-step PGD) [19]	$K+1$	$K+1$
M-Vanilla Ensemble AT (K-step PGD)	$M(K+1)$	$M(K+1)$
Multi-perturbation AT (K-step PGD) [30]	$MK+M$	$MK+M$
M-MSD AT (K-step PGD) [20]	$(1+M)K+1$	$K+1$
M-MIMO AT without Approx. AT(K-step PGD)	$K+1$	$MK+1$
(Ours) ***MAT***: M-MIMO Approx. AT (K-step PGD)	$K+1$	$K+1$
(Ours) M-MIMO MSD (K-step PGD)	$(1+M)K+1$	$K+1$

C Experiments setting

C.1 Hyperparameters for training the models

Models. We use ResNet18 [11] for all MNIST dataset. We also reproduce MSD results using the same neural network in their original paper. For CIFAR10, we use ResNet50.

Optimizers. For MNIST dataset, we use Adam optimizer [13]. For CIFAR10 dataset, we use SGD optimizier with momentum 0.9, and weight decay 5×10^{-4}.

Step size. We use the step size scheduler follows [28]. For MNIST, the step size linearly increase from 0 to 10^{-3} over the first 40% epochs, and down to 0 over the last 60% epochs. For CIFAR10, the step size linearly increase from 0 to 0.1 over the first 40% epochs, and down to 0.005 over the next 40% epochs, and finally back down to 0 in the last 20% epochs. The overall number of epochs is 25 for *MAT* and *MAT* + MSD. Notice we use

Hyperparameters for adversarial training. For all MNIST dataset, the inner maximization is optimized by $(\ell_\infty, \ell_2, \ell_1)$ PGD attacks with max iteration 100, radius $\epsilon = (0.3, 2.0, 10)$ and step size $\alpha = (0.01, 0.1, 0.8)$ respectively. For all CIFAR10 dataset, the inner maximization is optimized by $(\ell_\infty, \ell_2, \ell_1)$ PGD attacks with max iteration 50, attack radius $\epsilon = (0.03, 0.5, 12)$ and step size $\alpha = (0.003, 0.02, 1.0)$ respectively.

C.2 Hyperparameters for adversarial attacks used for robust test

For all the results, we evaluate the first 1000 test examples. To verify the performance of *MAT* and compare it to the performance of the baseline Adversarial Training, we evaluate the performance of our model against $\ell_\infty, \ell_2, \ell_1$ PGD attacks [19], Carlini Wagner [5] ℓ_2 attack,and two decision-based black-box attack: Boundary attack [2] and HopSkipJump attack[6].

For PGD attacks, To guarantee the convergence, we increase the max iteration to 200. For ℓ_1 PGD attack, we fixed the number of pixels to perturb per iteration as 20 during test. The other hyperparameters stay the same.

For other attacks, we set the hyperparameters to be consistent to the original works.

D The difference between the true gradient and approximate gradient after the steepest direction mapping operator $\mathcal{V}$

We claim that the cross gradient does influence the optimization, since the the small value of cross gradient may affect the gradient value after the steepest descent direction. In Figure 3, we define $E_p = \frac{\sum(\mathcal{V}_p(\nabla_{x_m}(\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_3)) \oplus \mathcal{V}_p(\nabla_{x_m}\mathcal{L}_m))}{|x_m|}$, where $|\cdot|$ denotes the cardinality of the x_m, $\oplus$ denotes XOR operation, and $p = \{1, \infty\}$. E_p calculates the fraction of entries that are different in the approximate gradient *after* steepest-descent projection compared to the true gradient. Notice that the XOR operator with a summation calculates the overall number of different pixels the two gradients have.$|x_m|$ denotes the overall pixels the gradient has. Figure 3 shows that the difference between the approximate gradient and the true gradient after the steep ascent direction mapping, where it decreases during training. Compared to Figure 1, which shows the difference between the true gradient and our approximation. The y axis denotes the fraction of pixels that changed by the cross gradient term. We can see from 3, especially, E_∞ does not converge to 0. They term might perform an important role to bring multiple robustness for the sub-model trained against a single attacks.

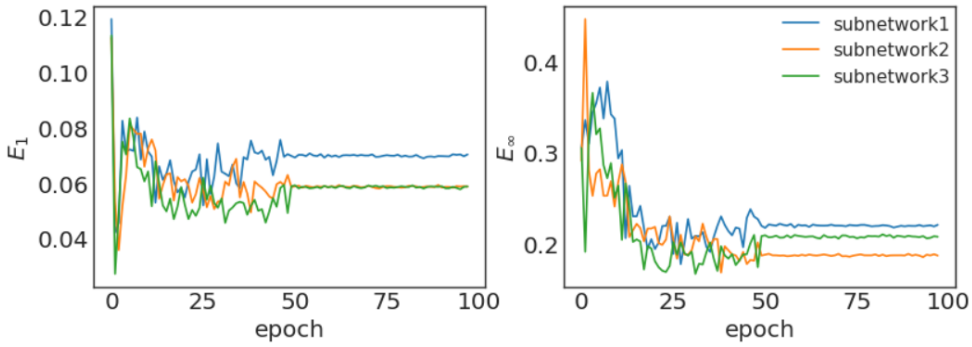

Figure 3: Approximate gradient difference caused by steepest descent operator for ℓ_1, ℓ_∞ norms. The disagreement E between the true gradient and approximate gradient converge with training.

E EXTRA EVALUATION: PERFORMANCE UNDER AUTOATTACK

AutoAttack [7] is a recent attack combined with AutoPGD, FAB attack and Square Attack. This evaluation doesn't requires parameter-tuning and also combined with gradient-based and gradient-free attacks. Since this is a ℓ_∞ attack, we will not report the model trained using ℓ_1 and ℓ_2 PGD attacks.

Table 8: Performance of each models against AutoAttack on MNIST

	standard training	ℓ_∞ AT	MSD	MAT+ℓ_∞	MAT	MAT+MSD
AutoAttack	0.1%	94.9%	57.0%	$\mathbf{95.6}\%$	82.3%	38.7%

Table 9: Performance of each models against AutoAttack on CIFAR10

	standard training	ℓ_∞ AT	MSD	MAT+ℓ_∞	MAT
AutoAttack	0.1%	78.3%	57.0%	79.1%	56.4%