# GENERATING ADVERSARIAL EXAMPLES WITH TASK ORIENTED MULTI-OBJECTIVE OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep learning models, even the-state-of-the-art ones, are highly vulnerable to adversarial examples. Adversarial training is one of the most efficient methods to improve the model's robustness. The key factor for the success of adversarial training is the capability to generate qualified and divergent adversarial examples which satisfy some objectives/goals (e.g., finding adversarial examples that maximize the model losses for simultaneously attacking multiple models). Therefore, multi-objective optimization (MOO) is a natural tool for adversarial example generation, where we search adversarial examples simultaneously maximizing some objectives/goals. However, we observe that a naive application of MOO tends to maximize all objectives/goals equally, without caring if an objective/goal has been achieved yet. This leads to useless effort to further improve the goal-achieved tasks, while putting less focus on the goal-unachieved tasks. In this paper, we propose Task Oriented MOO to address this issue, in the context where we can explicitly define the goal achievement for a task. Our principle is to only maintain the goal-achieved tasks, while letting the optimizer spend more effort on improving the goal-unachieved tasks. We conduct comprehensive experiments for our Task Oriented MOO on various adversarial example generation schemes. The experimental results firmly demonstrate the merit of our proposed approach.

## 1 INTRODUCTION

Deep neural networks are powerful models that achieve impressive performance across various domains such as bioinformatics (Spencer et al., 2015), speech recognition (Hinton et al., 2012), computer vision (He et al., 2016), and natural language processing (Vaswani et al., 2017). Despite achieving state-of-the-art performance, these models are extremely fragile, as one can easily craft small and imperceptible adversarial perturbations of input data to fool them, hence resulting in high misclassifications (Szegedy et al., 2014; Goodfellow et al., 2015). Accordingly, adversarial training (AT) (Madry et al., 2018; Zhang et al., 2019) has been proven to be one of the most efficient approaches to strengthen model robustness (Athalye et al., 2018). AT requires challenging models with divergent and qualified adversarial examples (Madry et al., 2018; Zhang et al., 2019; Bui et al., 2021) so that the robustified models can defend against adversarial examples. Therefore, generating adversarial examples is an important research topic in Adversarial Machine Learning (AML). Several perturbation based attacks have been proposed, notably PGD (Madry et al., 2018), CW (Carlini & Wagner, 2017), and AutoAttack (Croce & Hein, 2020). Most of them aim to optimize a single objective/goal, e.g., maximizing the cross-entropy (CE) loss w.r.t. the ground-truth label (Goodfellow et al., 2015; Madry et al., 2018), maximizing the Kullback-Leibler (KL) divergence w.r.t. the predicted probabilities of a benign example (Zhang et al., 2019), or minimizing a combination of perturbation size and predicted loss to a targeted class as in Carlini & Wagner (2017).

However, in many contexts, we need to find qualified adversarial examples satisfying multiple objectives/goals, e.g., finding an adversarial example that can *attack simultaneously multiple models* in an ensemble model (Pang et al., 2019; Bui et al., 2021), finding an universal perturbation that can *attack simultaneously multiple benign examples* (Moosavi-Dezfooli et al., 2017). Obviously, these adversarial generations have a nature of multi-objective problem rather than a single-objective one. Consequently, using *single-objective* adversarial examples leads to a much less adversarial robustness in ensemble learning as discussed in Section 4.2 and Appendix D.2.

Multi-Objective Optimization (MOO) (Désidéri, 2012) is an optimization problem to find a Pareto optimality that aims to optimize multiple objective functions. In a nutshell, MOO is a natural tool for the aforementioned multi-objective adversarial generations. However, a direct and naive application of MOO to generating robust adversarial examples for multiple models or ensemble of transformations does not work satisfactorily (cf. Appendix E). Concretely, it can be observed that the tasks are not optimized equally. The optimizing process focuses too much on one dominating task and can be trapped easily by it, hence leading to downgraded attack performances.

Intuitively, for multi-objective adversarial generations, we can explicitly investigate if an objective or a task achieves or fails to achieve its goal (e.g., the current adversarial example can fool a model successfully or unsuccessfully in multiple models). To avoid some tasks dominating others during the optimization process, we can favour more the tasks that are failing and pay less attention to the tasks that are performing well. For example, in the context of attacking multiple models, we update an adversarial example $x^a$ to favor the models that $x^a$ has not attacked successfully yet, while trying to maintain the attack capability of $x^a$ on the already successful models. In this way, we expect that no task really dominates others and all tasks can be updated equally to fulfill their goals.

Bearing this in mind, we propose a new framework named **TA**sk Oriented **M**ulti-**O**bjective **O**ptimization (TA-MOO) with multi-objective adversarial generations as the demonstrating applications. Specifically, we learn a weight vector (i.e., each dimension is the weight for a task) lying on a simplex corresponding to all tasks. To favor the unsuccessful tasks while maintaining the success of the successful ones, we propose a geometry-based regularization term that represents the distance between the original simplex and a reduced simplex which involves the weight vectors for the currently unsuccessful tasks only. Furthermore, along with the original quadratic term of the standard MOO helping to improve all tasks, minimizing our geometry-based regularization term encourages the weights of the goal-achieved tasks to be as small as possible, while inspiring those for the goal-unachieved ones to have a sum close to 1. By doing so, we aim to focus more on improving the goal-unachieved tasks, while still maintain the performance of goal-achieved tasks.

Most related work to ours is Wang et al. (2021), which considers the worst-case performance across all tasks. However, this original principle reduces the generalizability to other tasks. To mitigate this issue, a specific regularization was proposed to balance all tasks' weights. Our work, which casts an adversarial generation task as a multi-objective optimization problem, is conceptually different from that work, although both methods can be applied to similar tasks. Further discussion about relate work can be found in Appendix A.

To summarize, our contributions in this work include: **(C1)** Ours is the first work that views an adversarial generation task as a multi-objective optimization problem and applies multi-objective optimization to tackle this adversarial generation task. We show that if there is no gradient dominating issue, MOO can outperform the naive strategy by a large margin. **(C2)** To mitigate the shortcomings of the original MOO (Désidéri, 2012; Sener & Koltun, 2018) when applying to multi-objective adversarial generation, we propose the TA-MOO framework which employs a geometry-based regularization term to favor the unsuccessful tasks, while trying to maintain the performance of the already successful tasks. **(C3)** We conduct comprehensive experiments for three adversarial generation tasks and one adversarial training task including: (i) attacking multiple models, (ii) learning universal perturbation, (iii) attacking over many data transformations, and (iv) adversarial training on ensemble learning setting. The experimental results show that our TA-MOO outperforms the baselines by a wide margin on the three aforementioned adversarial generation tasks. More importantly, our adversary brings a great benefit on improving adversarial robustness.

## 2 BACKGROUND

We revisit the background of multi-objective optimization (MOO), which lays the foundation for our task-oriented MOO in the sequel. Given multiple objective functions $f(\delta) := [f_1(\delta), ..., f_m(\delta)]$ where each $f_i : \mathbb{R}^d \to \mathbb{R}$, we aim to find the Pareto optimal solution that simultaneously maximizes all objective functions:

$$\max_{\delta} f(\delta) := [f_1(\delta), ..., f_m(\delta)]. \tag{1}$$

While there are a variety of MOO solvers (Miettinen, 2012; Ehrgott, 2005), in this paper, we adapt from the multi-gradient descent algorithm (MGDA) that was proposed suitably for end-to-end learning by Désidéri (2012). Specifically, MGDA combines the gradients of individual objectives to a single

optimal direction that increases all objectives simultaneously. The optimal direction corresponds to the minimum-norm point that can be found by solving the quadratic programming problem:

$$w^* = \text{argmin}_{w \in \Delta_m} w^T Q w, \tag{2}$$

where $\Delta_m = \left\{\pi \in \mathbb{R}_+^m : \|\pi\|_1 = 1\right\}$ is the $m$-simplex and $Q \in \mathbb{R}^{m \times m}$ is the matrix with $Q_{ij} = \nabla_\delta f_i(\delta)^T \nabla_\delta f_j(\delta)$. Finally, the solution of the problem 1 can be found iteratively with each update step $\delta = \delta + \eta g$ where $g$ is the combined gradient $g = \sum_{i=1}^m w_i^* \nabla_\delta f_i(\delta)$ and $\eta > 0$ is a sufficiently small learning rate. Furthermore, Désidéri (2012) also proved that by using an appropriate learning rate at each step, we reach the Pareto optimality point $\delta^*$ at which there exist $w \in \Delta_m$ such that $\sum_{i=1}^m w_i \nabla_\delta f_i(\delta^*) = \mathbf{0}$.

# 3 Our Proposed Method

## 3.1 Task Oriented Multi-Objective Optimization

We now present our **TA**sk Oriented **M**ulti-**O**bjective **O**ptimization (TA-MOO). We consider the MOO problem in (1) where each task $\mathcal{T}_i \, (i = 1, ..., m)$ corresponds to the objective function $f_i(\delta) \; (i = 1, ..., m)$. Additionally, assume that given a task $\mathcal{T}_i$, we can explicitly observe if this task has currently achieved its goal (e.g., the current adversarial example $x$ can fool successfully the model $f_i$), which is named a goal-achieved task. We also name a task that has not achieved its goal a goal-unachieved task. Different from the standard MOO, which equally pays equal attention to all tasks, our TA-MOO focuses on improving the currently goal-unachieved tasks, while trying to maintain the performance of the goal-achieved tasks. By this principle, we expect all tasks would be equally improved to simultaneously achieve their goals.

To be more precise, we depart from $\delta_0$ and consecutively update in $L$ steps to obtain the sequence $\delta_1, \delta_2, ..., \delta_L$ that approaches the optimal solution. Considering the $t$-th step (i.e., $1 \leq t \leq L$), we currently have $\delta_t$ and need to update it to obtain $\delta_{t+1}$. We examine the tasks that have achieved their goals already and denote them as $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_s$ without the loss of generalization. Here we note that the list of goal-achieved tasks is empty if $s = 0$ and the list of goal-unachieved tasks is empty if $s = m$. Specifically, to find $\delta_{t+1}$, we first solve the following optimization problem (OP):

$$w^* = \text{argmin}_{w \in \Delta_m} \left\{ w^T Q w + \lambda \Omega(w) \right\}, \tag{3}$$

where $Q \in \mathbb{R}^{m \times m}$ with $Q_{ij} = \nabla_\delta f_i(\delta_t)^T \nabla_\delta f_j(\delta_t)$, $\lambda > 0$ is a trade-off parameter, and $\Omega(w)$ is a regularization term to let the weights focus more on the goal-unachieved tasks. We next compute the combined gradient $g_t$ and update $\delta_t$ as:

$$g_t = \sum_{i=1}^m w_i^* \nabla_\delta f_i(\delta_t) \text{ and } \delta_{t+1} = \delta_t + \eta g_t.$$

The OP in (3) consists of two terms. The first term $w^T Q w$ ensures that all tasks are improving, while the second term $\Omega(w)$ serves as the regularization to restrict the goal-achieved tasks $\mathcal{T}_1, ..., \mathcal{T}_s$ by setting the corresponding weights $w_1, ..., w_s$ as small as possible.

Before getting into the details of the regularization, we emphasize that to impose the constraint $w \in \Delta_m$, we parameterize $w = \text{softmax}(\alpha)$ with $\alpha \in \mathbb{R}^m$ and solve the OP in (3) using gradient descent. In what follows, we discuss our proposed geometry-based regularization term $\Omega(w)$.

**Simplex-based regularization.** Let $\mathcal{S}_u = \left\{\beta = [\beta_i]_{i=s+1}^m \in \mathbb{R}_+^{m-s} : \sum_{i=s+1}^m \beta_i = 1\right\}$ be a simplex w.r.t. the goal-unachieved tasks and $\mathcal{S} = \{\mathbf{0}_s\} \times \mathcal{S}_u$ be the extended simplex, where $\mathbf{0}_s$ is the $s$-dimensional vector of all zeros. We define the regularization term $\Omega(w)$ as the distance from $w$ to the extended simplex $\mathcal{S}$:

$$\Omega(w) = d(w, \mathcal{S}) = \min_{\pi \in \mathcal{S}} \|w - \pi\|_2^2. \tag{4}$$

Because $\mathcal{S}$ is a compact and convex set and $\|w - \pi\|_2^2$ is a differentiable and convex function, the optimization problem in (4) has a unique global minimizer $\Omega(w) = \|w - \text{proj}_{\mathcal{S}}(w)\|_2^2$, where the projection $\text{proj}_{\mathcal{S}}(w)$ is defined as

$$\text{proj}_{\mathcal{S}}(w) = \text{argmin}_{\pi \in \mathcal{S}} \|w - \pi\|_2^2.$$

The following lemma shows us how to find the projection $\text{proj}_{\mathcal{S}}(w)$ and evaluate $\Omega(w)$.

**Lemma 1.** *Sorting $w_{s+1:m}$ into $u_{s+1:m}$ such that $u_{s+1} \geq u_{s+2} \geq ... \geq u_m$. Defining $\rho = \max\left\{s+1 \leq i \leq m : u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) > 0\right\}$. Denoting $\gamma = \frac{1}{\rho}\left(1 - \sum_{i=s+1}^{\rho} u_i\right)$, the projection $proj_{\mathcal{S}}(w)$ can be computed as*

$$proj_{\mathcal{S}}(w)_i = \begin{cases} 0 & 1 \leq i \leq s \\ \max\{w_i + \gamma, 0\} & otherwise \end{cases}$$

*Furthermore, the regularization $\Omega(w)$ has the form:*

$$\Omega(w) = \sum_{i=1}^{s} w_i^2 + \sum_{i=s+1}^{m} \left(w_i - \max\{w_i + \gamma, 0\}\right)^2. \tag{5}$$

With further algebraic manipulations, $\Omega(w)$ can be significantly simplified as shown in Theorem 1.

**Theorem 1.** *The regularization $\Omega(w)$ has the following closed-form:*

$$\Omega(w) = \sum_{i=1}^{s} w_i^2 + \frac{1}{m-s}\left(1 - \sum_{i=s+1}^{m} w_i\right)^2. \tag{6}$$

The proof of Lemma 1 and Theorem 1 can be found in Appendix B.1. Evidently, the regularization term in Eq. (6) in Theorem 1 encourages the weights $w_{1:s}$ associated with the goal-achieved tasks to be as small as possible and the weights $w_{s+1:m}$associated with the goal-unachieved tasks to move closer to the simplex $\mathcal{S}_u$ (i.e., $\sum_{i=s+1}^{m} w_i$ is closer to 1).

**Parameterized TA-MOO.** Algorithm 1 summarizes the key steps of our TA-MOO. We use gradient descent to find solution $\delta$ for the OP 1 in $L$ steps and at each iteration we solve the OP in 3 in $K$ steps using gradient descent solver with the parameterization $w = \text{softmax}(\alpha)$. To reduce computational cost, at each iteration we reuse the previous solution $\alpha$ and use a few steps $K$ (i.e., $K \leq 10$) to get new solution. We then compute the combined gradient $g_t$ and finally update $\delta_t$ to $\delta_{t+1}$ using the combined gradient $g_t$ (or sign$(g_t)$ in the case of $L_\infty$ norm). The projecting operation in step 13 is to project $\delta$ to a valid space specifying to applications that we introduce hereon.

**Algorithm 1** Pseudocode for Parameterized TA-MOO.

**Input:** Multi-objective functions $f_{1:m}(\delta)$. $\delta$'s solver with L update steps and learning rate $\eta_\delta$. $w$'s Gradient Descent Solver (GD) with K update steps and learning rate $\eta_w$ and variable $\alpha$. The softmax function denotes by $\sigma$. Tradeoff parameter $\lambda$.

**Output:** The optimal solution $\delta^*$.

1: Initialize $\delta_0$ (e.g., $\delta_0 \sim \mathcal{U}(-\epsilon, \epsilon)$).
2: Initialize $\alpha_0 = [\alpha_0^i]_{i=1}^m$ with $\alpha_0^i = 1/m$.
3: **for** $t = 0$ to $L-1$ **do**
4: Collect list of tasks' gradients $\{\nabla_\delta f_i(\delta_t)\}_{i=1}^m$.
5: Compute $Q$ with $Q_{ij} = \nabla_\delta f_i(\delta_t)^T \nabla_\delta f_j(\delta_t)$.
6: Initialize $\alpha_{t+1} = \alpha_t$
7: **for** $k = 0$ to $K-1$ **do**
8: Compute $\mathcal{L}(\alpha_{t+1}) = \sigma(\alpha_{t+1})^T Q \sigma(\alpha_{t+1}) + \lambda\Omega(\sigma(\alpha_{t+1}))$.
9: Update $\alpha_{t+1} = \alpha_{t+1} - \eta_w \nabla_\alpha \mathcal{L}(\alpha_{t+1})$.
10: **end for**
11: Compute the combined gradient $g_t = \sum_{i=1}^m \sigma(\alpha_{t+1,i}) \nabla_\delta f_i(\delta_t)$.
12: Update $\delta_{t+1} = \delta_t + \eta_\delta g_t$.
13: Project $\delta_{t+1}$ to a valid space (specific to domain, e.g., $\|\delta\| \leq \epsilon$).
14: **end for**
15: Output $\delta^* = \delta_L$.

### 3.2 Applications in Adversarial Generation

Although TA-MOO is a general framework, we in this paper focus on its applications in adversarial generation. Following Wang et al. (2021), we consider three tasks of generating adversarial examples.

**Generating adversarial examples for an ensemble model.** Considering an ensemble classifier with multiple classification models $h_1, h_2, ..., h_m$, where $h_i(x) \in \Delta_M = \left\{\pi \in \mathbb{R}_+^M : \|\pi\|_1 = 1\right\}$ with the number of classes $M$. Given a data sample $x$, our aim is to find an adversarial example $x^a = x + \delta$ that can successfully attack all the models. Specifically, we consider a set of tasks each of which, $\mathcal{T}_i$, is about whether $x + \delta$ can successfully attack model $h_i$, defined as:

$$\mathbb{I}\left\{\text{argmax}_{1 \leq k \leq M} h_i(x + \delta, k) \neq y\right\},$$

where $y$ is the ground truth label of $x$, $\mathbb{I}$ is the indicator function and $h_i(x, k)$ returns the probability to predict $x$ to the class $k$. To find a perturbation $\delta$ that can attack successfully all models, we solve the following multi-objective optimization problem:

$$\max_{\delta : \|\delta\| \leq \epsilon} \left[f_1(\delta), ..., f_m(\delta)\right],$$

where $f_i(\delta) = \ell(h_i(x + \delta), y)$ with the loss function $\ell$ which could be the cross-entropy (CE) loss (Madry et al., 2018), the Kullback-Leibler (KL) loss (Zhang et al., 2019), or the Carlini-Wagner (CW) loss (Carlini & Wagner, 2017).

**Generating universal perturbations.** Considering a single classification model $h$ with $h(x) \in \Delta_M$ and a batch of data samples $x_1, x_2, ..., x_B$, we would like to find a perturbation $\delta$ with $\|\delta\| \leq \epsilon$ such that $x_i^a = x_i + \delta, i = 1, ..., B$, are adversarial examples. We define the task $\mathcal{T}_i$ as finding the adversarial example $x_i^a = x_i + \delta$ for data sample $x_i$. For each task $\mathcal{T}_i$, we can define its goal as finding successfully the adversarial example $x_i^a$:

$$\mathbb{I}\left\{\text{argmax}_{1 \leq k \leq M} h(x_i^a, k) \neq \text{argmax}_{1 \leq k \leq M} h(x_i, k)\right\}.$$

To find the perturbation $\delta$, we solve the following multi-objective optimization problem:

$$\max_{\delta : \|\delta\| \leq \epsilon} \left[f_1(\delta), ..., f_m(\delta)\right],$$

where $f_i(\delta) = \ell(h(x_i^a), y_i) = \ell(h(x_i + \delta), y_i)$ with $y_i$ the ground-truth label of $x_i$.

**Generating adversarial examples against transformations.** Considering a single classification model $h$ and $m$ categories of data transformation $\mathcal{P}_{1:m}$ (e.g., rotation, lighting, and translation). Our goal is to find an adversarial attack that is robust to these data transformations. Specifically, given a benign example $x$, we would like to learn a perturbation $\delta$ with $\|\delta\| \leq \epsilon$ that can successfully attack the model after any transformation $t_i \sim \mathcal{P}_i$ is applied. To formulate as an MOO problem, we consider the task $\mathcal{T}_i$ as finding the adversarial example $x_i^a = t_i(x + \delta)$ with $t_i \sim \mathcal{P}_i$. For each task $\mathcal{T}_i$, we can define the goal as finding successfully the adversarial example $x_i^a$:

$$\mathbb{I}\left\{\text{argmax}_{1 \leq k \leq M} h(x_i^a, k) \neq \text{argmax}_{1 \leq k \leq M} h(x, k)\right\}.$$

To find the perturbation $\delta$, we solve the following multi-objective optimization problem:

$$\max_{\delta : \|\delta\| \leq \epsilon} \left[f_1(\delta), ..., f_m(\delta)\right],$$

where $f_i(\delta) = \mathbb{E}_{t_i \sim \mathcal{P}_i}\left[\ell(h(t_i(x + \delta)), y)\right]$ with $y$ the ground-truth label of $x$.

## 4 EXPERIMENTS

In this section, we provide extensive experiments across four settings: (i) generating adversarial examples for ensemble of models (ENS, Sec 4.1), (ii) generating universal perturbation (UNI, Sec 4.3) , (iii) generating robust adversarial examples against Ensemble of Transformations (EoT, Sec 4.4), and (iv) adversarial training for ensemble of models (AT, Sec 4.2). The details of each setting can be found in Appendix C.

**General settings.** Through our experiments, we use six common architectures for the classifier including ResNet18 (He et al., 2016), VGG16 (Simonyan & Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), EfficientNet (Tan & Le, 2019), MobileNet Howard et al. (2017), and WideResNet Zagoruyko & Komodakis (2016) with the implementation [1]. We evaluate on the full testing set of

[1] https://github.com/kuangliu/pytorch-cifar

two benchmark datasets which are CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). We observed that the attack performance is saturated with standard training models. Therefore, to make the job of adversaries more challenging, we use Adversarial Training with PGD-AT (Madry et al., 2018) to robustify the models and use these robust models as the victim models in our experiments.

**Evaluation metrics.** We use three metrics to evaluate the attack performance including (i) A-All: the Attack Success Rate (ASR) when an adversarial example can achieve goals in all tasks. This is considered as the most important metric to indicate how well one method can achieve in all tasks; (ii)A-Avg: the average Attack Success Rate over all tasks which indicate the average attacking performance; (iii)$\{\text{A-i}\}_{i=1}^{K}$: Attack Success Rate in each individual task. For reading comprehension purposes, if necessary the highest/second highest performance in each experimental setting is highlighted in **Bold**/Underline and the most important metric(s) is emphasized in blue color.

**Baseline methods.** We compare our method with the **Uniform** strategy which assigns the same weight for all tasks and the **MinMax** method (Wang et al., 2021) which examines only the worst-case performance across all tasks. To increase the generality to other tasks, MinMax requires a regularization to balance between the average and the worst-case performance. We use the same attack setting for all methods: the attack is the $L_\infty$ untargeted attack with 100 steps, step size $\eta_\delta = 2/255$ and perturbation limitation $\epsilon = 8/255$. The GD solver in TA-MOO uses 10 steps with learning rate $\eta_w = 0.005$. Further detail can be found in Appendix C.

### 4.1 Adversarial Examples for Ensemble of Models (ENS)

**Experimental setting.** In our experiment, we use an ensemble of four adversarially trained models: ResNet18, VGG16, GoogLeNet, and EfficientNet. The architecture is the same for both the CIFAR10 and CIFAR100 datasets except for the last layer which corresponds with the number of classes in each dataset. The final output of the ensemble is an average of the probability outputs (i.e., output of the softmax layer). We use three different losses as an object for generating adversarial examples including CE (Madry et al., 2018), KL (Zhang et al., 2019), and CW (Carlini & Wagner, 2017).

***Results 1: TA-MOO achieves the best performance.*** Table 1 shows the results of attacking the ensemble model on the CIFAR10 and CIFAR100 datasets. It can be seen that TA-MOO significantly outperforms the baselines and achieves the best performance in all the settings. For example, the improvement over the Uniform strategy is around 10% on both datasets with the CE loss. Comparing to the MinMax method, the biggest improvement is around 3% for CIFAR10 with CW loss and the lowest one is around 0.6% with the KL loss. The improvement can be observed in all the settings, showing the generality of the proposed method.

***Results 2: When does not MOO work?*** It can be observed that MOO falls behind all other methods, even compared with the Uniform strategy. Our hypothesis for the failure of MOO is that in the original setting with an ensemble of 4 diverse architectures (i.e., ResNet18, VGG16, GoogLeNet, and EfficientNet) there is one task that dominates the others and makes MOO become trapped (i.e., focusing on improving the dominant task). To verify our hypothesis, we measure the gradient norm $\|\nabla_\delta f_i(\delta)\|$ corresponding to each model and the final weight $w$ of 1000 samples and report the results in Table 2. It can be seen that the EfficientNet has a much lower gradient strength, therefore, it has a much higher weight. This explains the highest ASR observed in EfficientNet and the large gap of 19% (56.11% in EfficientNet and 37.05% in GoogLeNet). To further confirm our hypothesis, we provide an additional experiment on a non-diverse ensemble model which consists of 4 individual ResNet18 models. It can be observed that in the non-diverse setting, the gradient strengths are more balanced across models, indicating that no task dominates others. As a result, MOO shows its effectiveness by outperforming the Uniform strategy by 4.3% in A-All.

***Results 3: The importance of the Task-Oriented regularization.*** It can be observed from Table 2 that in the diverse setting, TA-MOO has a much lower gap (4%) between the highest ASR (53.4% at EfficientNet) and the lowest one (49.29% at GoogLeNet) compared to MOO ( 19%). Moreover, while the ASR of EfficientNet is lower by 2.7%, the ASRs of all other architectures have been improved considerably (i.e., 12% in GoogLeNet). This improvement shows the importance of the Task-Oriented regularization, which helps to avoid being trapped by one dominating task, as happened in MOO. For the non-diverse setting, when no task dominates others, TA-MOO still shows its effectiveness when improving the ASR in all tasks by around 5%. The significant improvement can be observed in all settings (except the setting on EfficientNet with the CIFAR10 dataset) as shown in Table 1, and demonstrates the generality of the Task-Oriented regularization.

Table 1: Evaluation of Attacking Ensemble model on the CIFAR10 and CIFAR100 datasets.

| | | CW | | CE | | KL | |
|---|---|---|---|---|---|---|---|
| | | A-All | A-Avg | A-All | A-Avg | A-All | A-Avg |
| CIFAR10 | Uniform | 26.37 | **41.13** | 28.21 | 48.34 | 17.44 | 32.85 |
| | MinMax | 27.53 | 41.20 | 35.75 | **51.56** | 19.97 | **33.13** |
| | MOO | 18.87 | 34.24 | 25.16 | 44.76 | 15.69 | 29.54 |
| | TA-MOO | **30.65** | 40.41 | **38.01** | 51.10 | **20.56** | 31.42 |
| CIFAR100 | Uniform | 52.82 | **67.39** | 55.86 | 72.62 | 38.57 | **54.88** |
| | MinMax | 54.96 | 66.92 | 63.70 | 75.44 | 40.67 | 53.83 |
| | MOO | 51.16 | 65.87 | 58.17 | 73.19 | 39.18 | 53.44 |
| | TA-MOO | **55.73** | 67.02 | **64.89** | **75.85** | **41.97** | 53.76 |

Table 2: Attacking Ensemble model with a diverse set D={R-ResNet18, V-VGG16, G-GoogLeNet, E-EfficientNet} and non-diverse set ND={4 ResNets}. $w$ represents the final $w$ of MOO (mean $\pm$ std). $\|\nabla_\delta f_i(\delta)\|$ represents the gradient norm of each model (mean $\pm$ std).

| | | A-All | A-Avg | R/R1 | V/R2 | G/R3 | E/R4 |
|---|---|---|---|---|---|---|---|
| D | $\|\nabla_\delta f_i(\delta)\|$ | - | - | $7.15 \pm 6.87$ | $4.29 \pm 4.64$ | $7.35 \pm 7.21$ | $0.98 \pm 0.72$ |
| | $w$ | - | - | $0.15 \pm 0.14$ | $0.17 \pm 0.13$ | $0.15 \pm 0.14$ | $0.53 \pm 0.29$ |
| | Uniform | 28.21 | 48.34 | 48.89 | 49.08 | 48.38 | 47.03 |
| | MOO | 25.16 | 44.76 | 39.06 | 46.83 | 37.05 | 56.11 |
| | TA-MOO | 38.01 | 51.10 | 49.55 | 52.15 | 49.29 | 53.40 |
| ND | $\|\nabla_\delta f_i(\delta)\|$ | - | - | $8.41 \pm 8.22$ | $6.68 \pm 6.95$ | $7.36 \pm 6.03$ | $5.67 \pm 6.09$ |
| | $w$ | - | - | $0.23 \pm 0.21$ | $0.24 \pm 0.17$ | $0.23 \pm 0.19$ | $0.30 \pm 0.21$ |
| | Uniform | 28.17 | 48.75 | 51.94 | 45.55 | 54.15 | 43.34 |
| | MOO | 32.50 | 52.21 | 53.25 | 49.05 | 56.80 | 49.76 |
| | TA-MOO | 41.01 | 57.33 | 58.88 | 55.32 | 60.81 | 54.29 |

***Results 4: TA-MOO achieves the best transferability on a diverse set of ensembles.***

Table 3 reports the SAR-All metric of transferred adversarial examples crafted from a source ensemble (RME) on attacking target ensembles (e.g., RMEVW is an ensemble of 5 models). A higher number indicates a higher success rate of attacking a target model, therefore, also implies a higher transferability of adversarial examples. It can be seen that our TA-MOO adversary achieves the highest attacking performance on the whitebox attack setting, with a huge gap of 9.24% success rate over the Uniform strategy. Our method also achieves the highest transferability regardless diversity of a target ensemble. More specifically, on target models such as REV, MEV, and RMEV, where members in the source ensemble (RME) are also in the target ensemble, our TA-MOO significantly outperforms the Uniform strategy, with the highest improvement is 5.19% observed on target model RMEV. On the target models EVW and MVW which are less similar to the source model, our method still outperforms the Uniform strategy by 1.46% and 1.65%. The superior performance of our adversary on the transferability shows another benefit of using multi-objective optimization in generating adversarial examples. By reaching the intersection of all members' adversarial regions, our adversary is capable to generate a common vulnerable pattern on an input image shared across architectures, therefore, increasing the transferability of adversarial examples. More discussion can be found in Appendix D.1.

### 4.2 Adversarial Training with TA-MOO for Ensemble of Models (ENS)

We conduct adversarial training with adversarial examples generated by our MOO and TA-MOO attacks to verify the quality of these adversarial examples and report results on Table 4. The detailed setting and more experimental results can be found in Appendix D.2. **Result 1: Reducing transferability.** It can be seen that the SAR-All of MOO-AT and TA-MOO-AT are much lower than that on other methods. More specifically, the gap of SAR-All between PGD-AT and TA-MOO-AT is (5.33%) 6.13% on the (non) diverse setting. The lower SAR-All indicating that adversarial examples are harder to transfer among ensemble members on the TA-MOO-AT model than on the PGD-AT

Table 3: Evaluation on the Transferability of adversarial examples. Each cell (row-ith, column-jth) reports SAR (higher is better) of adversarial examples from the same source architecture (RME) with an adversary at row-ith to attack an ensemble at column-jth. Each architecture has been denoted by symbols such as R: ResNet18, M: MobileNet, E: EfficientNet, V: VGG16, W: WideResNet. For examples, RME represents for an ensemble of ResNet18, MobileNet and EfficientNet.

| | RME | RVW | EVW | MVW | REV | MEV | RMEV | RMEVW |
|---|---|---|---|---|---|---|---|---|
| Uniform | 31.73 | 25.03 | 22.13 | 22.73 | 29.50 | 28.44 | 26.95 | 20.50 |
| MinMax | 40.01 | 23.75 | 22.39 | 23.34 | 32.57 | 32.75 | 31.85 | 21.99 |
| MOO | 35.20 | 24.25 | 22.94 | 23.76 | 30.65 | 32.28 | 29.49 | 21.77 |
| TA-MOO | **40.97** | **25.13** | **23.59** | **24.38** | **33.00** | **33.05** | **32.14** | **23.04** |

Table 4: Robustness evaluation of Adversarial Training methods on the CIFAR10 dataset. RME represents an ensemble of ResNet18 (R), MobileNet (M) and EfficientNet E), while MobiX3 represents an ensemble of three MobileNets. NAT and ADV measure the natural accuracy and the robust accuracy against PGD-Linf attack (↑the higher the better). Other metrics measure the success attack rate (SAR) of adversarial examples generated by the same PGD-Linf attack on fooling each single member and all members of the ensemble (↓the lower the better).

| | MobiX3 | | | | RME | | | |
|---|---|---|---|---|---|---|---|---|
| | NAT↑ | ADV↑ | A-All↓ | A-Avg↓ | NAT↑ | ADV↑ | A-All↓ | A-Avg↓ |
| PGD-AT | **80.43** | 32.78 | 54.34 | 73.89 | **86.52** | 37.36 | 49.01 | 69.75 |
| MinMax-AT | 79.01 | 37.28 | 50.28 | **66.77** | 83.16 | 40.40 | 46.91 | 65.73 |
| MOO-AT | 79.38 | 33.04 | **46.28** | 74.36 | 82.04 | 37.48 | 45.24 | 70.11 |
| TA-MOO-AT | 79.22 | **38.22** | 48.21 | 67.83 | 82.59 | **41.32** | **43.68** | **65.09** |

model. **Result 2: Producing more robust single members.** The comparison of average SAR shows that adversarial training with TA-MOO produces more robust single models than PGD-AT does. More specifically, the average robust accuracy (measured by 100% - *A-Avg*) of TA-MOO-AT is 32.17%, an improvement of 6.06% over PGD-AT in the non-diverse setting, while there is an improvement of 4.66% in the diverse setting. **Result 3: Adversarial training with TA-MOO achieves the best robustness.** More specifically, on the non-divese setting, TA-MOO-AT achives 38.22% robust accuracy, an improvement of 1% over MinMax-AT and 5.44% over standard PGD-AT. On the diverse setting, the improvement over MinMax-AT and PGD-AT are 0.9% and 4%, respectively. The root of the improvement is the ability to generate stronger adversarial examples in the the sense that they can challenge not only the entire ensemble model but also all single members. These adversarial examples lie in the joint insecure region of members (i.e., the low confidence region of multiple classes), therefore, making the decision boundaries more separate. As a result, adversarial training with TA-MOO produces more robust single models (i.e., lower SAR-Avg) and significantly reduces the transferability of adversarial examples among members (i.e., lower SAR-All). These two conditions explain the best ensemble adversarial robustness achieved by TA-MOO.

### 4.3 Universal Perturbation (UNI)

**Experimental setting.** We follow the experimental setup in Wang et al. (2021), where the full test set (10k images) is randomly divided into equal-size groups (K images per group). The comparison has been conducted on the CIFAR10 and CIFAR100 datasets, with an adversarially trained ResNet18 model and CW loss. We observed that the ASR-All was mostly zero, indicating that it is difficult to generate a general perturbation for all data points. Therefore, in Table 5 we use ASR-Avg to compare the performances of the methods. More experiments on VGG16 and EfficientNet models can be found in Appendix D.3.

**Results.** Table 5 shows the evaluation of generating universal perturbations on the CIFAR10 and CIFAR100 datasets, respectively. $K$ represents the number of images that are using the same perturbation. The larger the value of $K$, the harder it is to generate a universal perturbation that can be applied successfully to all images. It can be seen that with a small number of tasks (i.e., $K$=4), MOO and TA-MOO achieve lower performance than the MinMax method. However, with a large number of tasks (i.e, $K \geq 8$), MOO and TA-MOO show their effectiveness and achieve the

Table 5: Evaluation of generating Universal Perturbation on the CIFAR10 and CIFAR100 datasets.

| | CIFAR10 | | | | | CIFAR100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K=4 | K=8 | K=12 | K=16 | K=20 | K=4 | K=8 | K=12 | K=16 | K=20 |
| Uniform | 37.52 | 30.34 | 27.41 | 25.52 | 24.31 | 65.40 | 58.99 | 55.33 | 53.02 | 51.49 |
| MinMax | **50.13** | 33.68 | 20.46 | 15.74 | 14.73 | **74.73** | 62.29 | 52.05 | 45.26 | 42.33 |
| MOO | 43.80 | 35.92 | 31.41 | 28.75 | 26.83 | 69.35 | 62.72 | 57.72 | 54.12 | 52.25 |
| TA-MOO | 48.00 | **39.31** | **34.96** | **31.84** | **30.12** | 72.74 | **68.06** | **62.33** | **57.48** | **54.12** |

Table 6: Robust adversarial examples against transformations evaluation. I: Identity, H: Horizontal flip, V: Vertical flip, C: Center crop, G: Adjust gamma, B: Adjust brightness, R: Rotation.

| | | A-All | A-Avg | I | H | V | C | G | B | R |
|---|---|---|---|---|---|---|---|---|---|---|
| C10 | Uniform | 25.98 | **55.33** | **44.85** | 41.58 | 82.90 | **72.56** | **45.92** | **49.59** | **49.93** |
| | MinMax | 30.54 | 52.20 | 43.31 | 41.59 | 78.80 | 64.83 | 44.38 | 46.53 | 45.97 |
| | MOO | 21.25 | 49.81 | 36.23 | 33.93 | **87.47** | 71.05 | 37.68 | 40.21 | 42.12 |
| | TA-MOO | **31.10** | 55.26 | 44.15 | **41.86** | 85.19 | 71.86 | 45.53 | 48.70 | 49.54 |
| C100 | Uniform | 56.19 | 76.23 | **70.43** | 69.01 | 87.66 | 87.36 | 71.40 | 74.25 | 73.47 |
| | MinMax | 59.75 | 75.72 | 70.13 | 69.26 | 87.45 | 86.03 | 71.54 | 73.30 | 72.32 |
| | MOO | 53.17 | 74.21 | 66.96 | 65.68 | **89.16** | 87.03 | 68.49 | 71.11 | 71.06 |
| | TA-MOO | **60.88** | **76.71** | **70.43** | **69.37** | 89.11 | **87.95** | **71.70** | **74.73** | **73.69** |

best performance. More specifically, on the CIFAR10 dataset, the improvements of MOO over the Uniform strategy are 5.6%, 4%, 3.2%, and 2.5% with $K = 8$, $K = 12$, $K = 16$, and $K = 20$, respectively. On the same setting, TA-MOO significantly improves MOO by around 4% in all the $K$ settings and consistently achieves the best performance. Unlike the ENS setting, in the UNI setting, MOO consistently achieves better performance than the Uniform strategy . This improvement can be explained by the fact that in the UNI setting with the same architecture and data transformation, no task dominates the others. There will be a case (a group) when one sample is extremely close to/far from the decision boundary, and hence easier/harder to fool. However, in the entire test set with a large number of groups, the issue of dominating tasks is lessened.

### 4.4 Robust Adversarial Examples against Transformations (EoT)

**Results.** Table 6 shows the evaluation on the CIFAR10 and CIFAR100 datasets with 7 common data transformations. It can be observed that (i) MOO has a lower performance than the baselines, (ii) the Task Oriented regularization significantly boosts the performance, and (iii) our TA-MOO method achieves the best performance on both settings and outperforms the MinMax method 0.6% and 1.1% in the CIFAR10 and the CIFAR100 experiments, respectively. The low performance of MOO in observation (i) is again caused by the issue of one task dominating others. In the EoT setting, it is because of the V-vertical flip transformation as shown in Table 6. Observation (ii) provides another piece of evidence to support the effectiveness of the Task-Oriented regularization for MOO. This regularization boosts the ASRs in all the tasks (except V - the dominant one), increases the average ASR by 5.45% and 2.5% in the CIFAR10 and CIFAR100 experiments, respectively.

## 5 Conclusion

In this paper, we propose Task Oriented Multi-Objective Optimization (TA-MOO), with specific applications to adversarial generation tasks. We develop a geometry-based regularization term to favor the goal-unachieved tasks, while trying to maintain the the goal-achieved tasks. We conduct comprehensive experiments to showcase the merit of our proposed approach on generating adversarial examples and adversarial training. On the other hand, there are acknowledged limitations of our method such as weaknesses of the gradient-based solver and lacking theory on algorithm's convergence which might be target for future works.

## REFERENCES


Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In International Conference on Machine Learning, pp. 274–283, 2018. 1, D.8

Emil Björnson and Eduard Jorswieck. Optimal resource allocation in coordinated multi-cell systems. Now Publishers Inc, 2013. E.3

Emil Bjornson, Eduard Axel Jorswieck, Mérouane Debbah, and Bjorn Ottersten. Multiobjective signal processing optimization: The way to balance conflicting metrics in 5g systems. IEEE Signal Processing Magazine, 31(6):14–23, 2014. E.3

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. A

Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In Advances in Neural Information Processing Systems, pp. 12861–12871, 2019. D.7

Anh Bui, Trung Le, He Zhao, Paul Montague, Olivier deVel, Tamas Abraham, and Dinh Phung. Improving adversarial robustness by enforcing local and global compactness. arXiv preprint arXiv:2007.05123, 2020. A

Anh Tuan Bui, Trung Le, He Zhao, Paul Montague, Olivier deVel, Tamas Abraham, and Dinh Phung. Improving ensemble robustness by collaboratively promoting and demoting adversarial robustness. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 6831–6839, 2021. 1

Rafael Caballero, Lourdes Rey, Francisco Ruiz, and Mercedes González. An algorithmic package for the resolution and analysis of convex multiple objective problems. In Multiple criteria decision making, pp. 275–284. Springer, 1997. E.3

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp), pp. 39–57. IEEE, 2017. 1, 3.2, 4.1, A, C, D.7

Carlos A Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information systems, 1(3):269–308, 1999. E.3

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. arXiv preprint arXiv:2003.01690, 2020. 1, A, D.7

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2021. URL https://openreview.net/forum?id=SSKZPJCt7B. D.8

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In Multi-objective evolutionary optimisation for product design and manufacturing, pp. 3–34. Springer, 2011. A, E.3, E.3

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Mathematique, 350(5-6):313–318, 2012. 1, 2, 2, A

Shangchen Du, Shan You, Xiaojie Li, Jianlong Wu, Fei Wang, Chen Qian, and Changshui Zhang. Agree to disagree: Adaptive ensemble knowledge distillation in gradient space. Advances in Neural Information Processing Systems, 33, 2020. A

Matthias Ehrgott. Multicriteria optimization, volume 491. Springer Science & Business Media, 2005. 2

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6572. 1, A

Pengxin Guo, Yuancheng Xu, Baijiong Lin, and Yu Zhang. Multi-task adversarial attack. arXiv preprint arXiv:2011.09824, 2020. A

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, pp. 770–778, 2016. 1, 4, C

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6):82–97, 2012. 1

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017. 4, C

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 4, C

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In Artificial intelligence safety and security, pp. 99–112. Chapman and Hall/CRC, 2018. A

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. Advances in neural information processing systems, 32, 2019. A

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. Advances in neural information processing systems, 29, 2016. A

Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. Advances in Neural Information Processing Systems, 34, 2021. A

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. 1, 3.2, 4, 4.1, A, C, C, D.7

Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In International Conference on Machine Learning, pp. 6597–6607. PMLR, 2020. A

Kaisa Miettinen. Nonlinear multiobjective optimization, volume 12. Springer Science & Business Media, 2012. 2

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1765–1773, 2017. 1, A

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 4970–4979. PMLR, 09–15 Jun 2019. 1, A

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? Advances in Neural Information Processing Systems, 33:3533–3545, 2020. D.8

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. Advances in neural information processing systems, 31, 2018. 1, A, B.2, E.3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 4, C

M. Spencer, J. Eickholt, and J. Cheng. A deep learning network approach to ab initio protein secondary structure prediction. IEEE/ACM Trans. Comput. Biol. Bioinformatics, 12(1):103–112, January 2015. ISSN 1545-5963. 1

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun (eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014. URL http://arxiv.org/abs/1312.6199. 1

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015. 4, C

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning, pp. 6105–6114. PMLR, 2019. 4, C

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998–6008, 2017. 1

Jingkang Wang, Tianyun Zhang, Sijia Liu, Pin-Yu Chen, Jiacen Xu, Makan Fardad, and Bo Li. Adversarial attack generation empowered by min-max optimization. Advances in Neural Information Processing Systems, 34, 2021. 1, 3.2, 4, 4.3, A, C, C, E.4

Weiran Wang and Miguel A Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. arXiv preprint arXiv:1309.1541, 2013. B.1

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In International Conference on Learning Representations, 2019. D.8

Feiyang Ye, Baijiong Lin, Zhixiong Yue, Pengxin Guo, Qiao Xiao, and Yu Zhang. Multi-objective meta learning. Advances in Neural Information Processing Systems, 34, 2021. A

S. Zagoruyko and N. Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016. 4, C

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 7472–7482, 2019. 1, 3.2, 4.1, A, C

# APPENDIX

The Appendix provides technical and experimental details as well as auxiliary aspects to complement the main paper. Briefly, it contains the following:

- Appendix A: Discussion on related work.
- Appendix B: Detailed proof and an illustration of our methods.
- Appendix C: Detailed description of experimental settings.
- Appendix D.1: Additional experiments on transferability of adversarial examples in the ENS setting.
- Appendix D.2: Additional experiments on adversarial training with our methods.
- Appendix D.3: Additional experiments on the UNI setting.
- Appendix D.4: Additional experiments on the EoT setting.
- Appendix D.5: Additional comparison on speed of generating adversarial examples.
- Appendix D.6: Additional experiments on sensitivity to hyper-parameters.
- Appendix D.7: Additional comparison with standard attacks on attacking performance.
- Appendix D.8: Additional experiments on attacking the ImageNet dataset.
- Appendix E.1: Additional discussions on the dominating issue and when MOO can work.
- Appendix E.2: A summarary on the importance of Task-Oriented regularization.
- Appendix E.3: Discussion on the limitation of MOO solver.
- Appendix E.4: Discussion on correlation between the objective loss and attack performance.

## A Related Work

**Multi-Objective Optimization for multi-task learning.** (Désidéri, 2012) proposed a multi-gradient descent algorithm for multi-objective optimization (MOO) which opens the door for the applications of MOO in machine learning and deep learning. Inspired by Désidéri (2012), MOO has been applied in multi-task learning (MTL) (Sener & Koltun, 2018; Mahapatra & Rajan, 2020), few-shot learning (Ye et al., 2021), and knowledge distillation (Du et al., 2020). Specifically, the work of Sener & Koltun (2018) viewed multi-task learning as a multi-objective optimization problem, where a task network consists of a shared feature extractor and a task-specific predictor. The work of Mahapatra & Rajan (2020) developed a gradient-based multi-objective MTL algorithm to find a solution that satisfies the user preferences. The work of Lin et al. (2019) proposed a Pareto MTL to find a set of well-distributed Pareto solutions which can represent different trade-offs among different tasks. Recently, the work of Liu et al. (2021) leveraged MOO with Stein Variational Gradient Descent (Liu & Wang, 2016) to diversify the solutions of MOO. Additionally, the work of Ye et al. (2021) proposed a bi-level MOO which can be applied to few-shot learning. Finally, the work of Du et al. (2020) applied MOO to enable knowledge distillation from multiple teachers.

**Generating adversarial examples with single-objective and multi-objective optimizations.** Generating qualified adversarial examples is crucial for adversarial training (Madry et al., 2018; Zhang et al., 2019). Many perturbation based attacks have been proposed, notably FGSM (Goodfellow et al., 2015), PGD (Madry et al., 2018), TRADES (Zhang et al., 2019), CW (Carlini & Wagner, 2017), BIM (Kurakin et al., 2018), and AutoAttack (Croce & Hein, 2020). Most adversarial attacks aim to maximize a single objective, e.g., maximizing the cross-entropy (CE) loss w.r.t. the ground-truth label (Madry et al., 2018), maximizing the Kullback-Leibler (KL) divergence w.r.t. the predicted probabilities of a benign example (Zhang et al., 2019), or maximizing the CW loss (Carlini & Wagner, 2017). However, in some contexts, we need to generate adversarial examples maximizing multiple objectives or goals, e.g., attacking multiple models (Pang et al., 2019; Bui et al., 2020) or finding universal perturbations (Moosavi-Dezfooli et al., 2017).

To this end, the work of Wang et al. (2021) examined the worst-case scenario by casting the problem of interest as a min-max problem for finding the weight of each task. However, this principle leads to

a problem of lacking generality in other tasks. To mitigate the issue, Wang et al. (2021) proposed a regularization to strike a balance between the average and the worst-case performance. The final optimization was formulated as follow:

$$\max_{\delta:\|\delta\|\leq\epsilon} \min_{w\in\Delta_m} \sum_{i=1}^{K} w_i f_i(\delta) + \frac{\gamma}{2}\|w - 1/K\|_2^2,$$

Where $f_i(v)$ is the victim model's loss (i.e., cross entropy loss or KL divergence) and $\gamma > 0$ is the regularization parameter. The authors used the bisection method (Boyd et al., 2004) with project gradient descent for the inner minimization and project gradient ascent for the outer maximization. There are several major differences in comparison to our MOO and TA-MOO methods: (i) In principle, MinMax considers the worst-case performance only while our methods improve performance of all tasks simultaneously. (ii) MinMax weighs the tasks' losses to find the minimal weighted sum loss in its inner minimization, however, as discussed in Section E.4 the higher weighted loss does not directly imply the higher success rate in attacking multi-tasks simultaneously. In contrast, our methods use multi-gradient descent algorithm (Deb, 2011) in order to increase losses of all tasks simultaneously. (iii) The original principle of MinMax leads to the biasing problem to the worst-case task. The above regularization has been used to mitigate the issue, however, it considers all tasks equally. In contrast, our TA-MOO takes goal-achievement status of each task into account and focuses more on the goal-unachieved tasks.

Recently, Guo et al. (2020) proposed a multi-task adversarial attack and demonstrated on the universal perturbation problem. However, while Wang et al. (2021) and ours can be classified as an iterative optimization-based attack, Guo et al. (2020) requires a generative model in order to generate adversarial examples. While this line of attack is faster than optimization-based attacks at the inference phase, it requires to train a generator on several tasks beforehand. Due to the difference in setting, we do not compare with that work in this paper.

# B Further Details of the Proposed Method

## B.1 Proofs

**Lemma 1.** *Sorting $w_{s+1:m}$ into $u_{s+1:m}$ such that $u_{s+1} \geq u_{s+2} \geq ... \geq u_m$. Defining $\rho = \max\left\{s+1 \leq i \leq m : u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) > 0\right\}$. Denoting $\gamma = \frac{1}{\rho}\left(1 - \sum_{i=s+1}^{\rho} u_i\right)$, the projection $proj_{\mathcal{S}}(w)$ can be computed as*

$$proj_{\mathcal{S}}(w)_i = \begin{cases} 0 & 1 \leq i \leq s \\ \max\{w_i + \gamma, 0\} & otherwise \end{cases}$$

*Furthermore, the regularization $\underset{s}{\Omega}(w)$ has the form:*

$$\Omega(w) = \sum_{i=1}^{s} w_i^2 + \sum_{i=s+1}^{m} \left(w_i - \max\{w_i + \gamma, 0\}\right)^2. \tag{5}$$

*Proof.* The proof is based on Wang & Carreira-Perpinán (2013) with modifications. We need to solve the following OP:

$$\begin{aligned} &\min_{\pi} \frac{1}{2}\|w - \pi\|_2^2 \\ \text{s.t. }: &\pi \geq \mathbf{0} \\ &\|\pi\|_1 = 1. \end{aligned}$$

We note that $\pi_1 = ... = \pi_s = 0$. The OP of interest reduces to

$$\begin{aligned} &\min_{\pi_{s+1:m}} \frac{1}{2}\sum_{i=s+1}^{m} (\pi_i - w_i)^2 \\ \text{s.t. }: &\pi_{s+1:m} \geq \mathbf{0} \\ &\sum_{i=s+1}^{m} \pi_i = 1. \end{aligned}$$

Using the Karush-Kuhn-Tucker (KKT) theorem, we construct the following Lagrange function:

$$\mathcal{L}(\pi, \gamma, \beta) = \frac{1}{2} \sum_{i=s+1}^{m} (\pi_i - w_i)^2 - \gamma \left( \sum_{i=s+1}^{m} \pi_i - 1 \right) - \sum_{i=s+1}^{m} \beta_i \pi_i.$$

Setting the derivative w.r.t. $\pi_i$ to zeros and using the KKT conditions, we obtain:

$$\begin{aligned} \pi_i - w_i - \gamma - \beta_i = 0, \forall i = s+1, ..., m \\ \sum_{i=s+1}^{m} \pi_i =& 1 \\ \beta_i \geq 0, \pi_i \geq 0, &\beta_i \pi_i = 0, \forall i = s+1, ..., m. \end{aligned}$$

If $\pi_i > 0$, $\beta_i = 0$, hence $\pi_i = w_i + \gamma > 0$. Otherwise, if $\pi_i = 0$, $w_i + \gamma = -\beta_i \leq 0$. Therefore, $w_{s+1:m}$ has the same order as $\pi_{s+1:m}$ and we can arrange them as:

$$\pi_{s+1} \geq \pi_{s+2} \geq ... \geq \pi_\rho > \pi_{\rho-1} = ... = \pi_m = 0.$$

$$u_{s+1} = w_{s+1} \geq u_{s+2} = w_{s+2} \geq .... \geq u_p = w_p \geq u_{\rho-1} = w_{\rho-1} \geq ... \geq u_m = w_m \geq 0.$$

It appears that $1 = \sum_{i=s+1}^{m} \pi_i = \sum_{i=s+1}^{\rho} \pi_i = \sum_{i=s+1}^{\rho} (w_i + \gamma) = \sum_{i=s+1}^{\rho} w_i + (\rho - s)\gamma$. Hence, we gain $\gamma = \frac{1}{\rho - s}\left[1 - \sum_{i=s+1}^{\rho} w_i\right] = \frac{1}{\rho - s}\left[1 - \sum_{i=s+1}^{\rho} u_i\right]$. We now prove that $\rho = \max\left\{s+1 \leq i \leq m : u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) > 0\right\}$. □

- For $i = \rho$, we have

$$u_\rho + \frac{1}{\rho - s}\left(1 - \sum_{j=s+1}^{\rho} u_j\right) = u_\rho + \gamma = w_\rho + \gamma > 0.$$

- For $i < \rho$, we have

$$\begin{aligned} u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) &= \frac{1}{i-s}\left((i-s)u_i + 1 - \sum_{j=s+1}^{i} u_j\right) \\ &= \frac{1}{i-s}\left[(i-s)w_i + \sum_{j=s+1}^{\rho-1} \pi_j - \sum_{j=s+1}^{i} w_j\right] \\ &= \frac{1}{i-s}\left[(i-s)w_i + \sum_{j=i+1}^{\rho-1} \pi_j + \sum_{j=s+1}^{i} (\pi_j - w_j)\right] \\ &= \frac{1}{i-s}\left[(i-s)(w_i + \gamma) + \sum_{j=i+1}^{\rho-1} \pi_j\right] \\ &= \frac{1}{i-s}\left[(i-s)\pi_i + \sum_{j=i+1}^{\rho-1} \pi_j\right] > 0. \end{aligned}$$

- For $i > \rho$, we have

$$
\begin{aligned}
u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) &= \frac{1}{i-s}\left((i-s)u_i + 1 - \sum_{j=s+1}^{i} u_j\right) \\
&= \frac{1}{i-s}\left((i-s)w_i + \sum_{j=s+1}^{\rho-1} \pi_j - \sum_{j=s+1}^{i} w_j\right) \\
&= \frac{1}{i-s}\left((i-s)w_i + \sum_{j=s+1}^{\rho-1} (\pi_j - w_j) - \sum_{j=\rho}^{i} w_j\right) \\
&= \frac{1}{i-s}\left((i-s)w_i + (\rho-s-1)\gamma - \sum_{j=\rho}^{i} w_j\right) \\
&= \frac{1}{i-s}\left((\rho-s-1)(w_i+\gamma) + \sum_{j=\rho}^{i}(w_i - w_j)\right) \leq 0.
\end{aligned}
$$

Therefore, $\rho = \max\left\{s+1 \leq i \leq m : u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) > 0\right\}$. Finally, we also have $\pi_i = \max\{w_i + \gamma, 0\}, i = s+1, ..., m$ and $\pi_i = 0, i = 1, ..., s$.

**Theorem 1.** *The regularization $\Omega(w)$ has the following closed-form:*

$$
\Omega(w) = \sum_{i=1}^{s} w_i^2 + \frac{1}{m-s}\left(1 - \sum_{i=s+1}^{m} w_i\right)^2. \tag{6}
$$

*Proof.* Recall $\rho = \max\left\{s+1 \leq i \leq m : u_i + \frac{1}{i-s}\left(1 - \sum_{j=s+1}^{i} u_j\right) > 0\right\}$. Therefore, $\rho = m$ because we have

$$
u_m + \frac{1}{m-s}\left(1 - \sum_{j=s+1}^{m} u_j\right) = w_m + \frac{1}{m-s}\left(1 - \sum_{j=s+1}^{m} w_j\right) = w_m + \frac{\sum_{j=1}^{s} w_j}{m-s} > 0.
$$

It follows that

$$
\gamma = \frac{1}{m-s}\left(1 - \sum_{i=s+1}^{m} u_i\right) = \frac{1}{m-s}\left(1 - \sum_{i=s+1}^{m} w_i\right) \geq 0.
$$

$$
\text{proj}_{\mathcal{S}}(w)_i = \begin{cases} 0 & 1 \leq i \leq s \\ \max\{w_i + \gamma, 0\} = w_i + \gamma & \text{otherwise} \end{cases}
$$

$$
\begin{aligned}
\Omega(w) &= \sum_{i=1}^{s} w_i^2 + \sum_{i=s+1}^{m} \left(w_i - \max\{w_i + \gamma, 0\}\right)^2 \\
&= \sum_{i=1}^{s} w_i^2 + \sum_{i=s+1}^{m} \gamma^2 = \sum_{i=1}^{s} w_i^2 + (m-s)\gamma^2 \\
&= \sum_{i=1}^{s} w_i^2 + \frac{1}{m-s}\left(1 - \sum_{i=s+1}^{m} w_i\right)^2.
\end{aligned}
$$

□

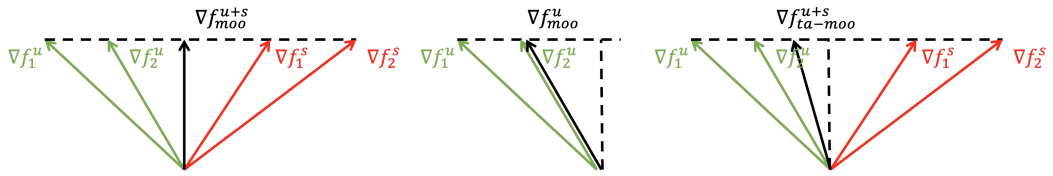



Figure 1: Visualization of standard MOO and TA-MOO solutions in a scenario of 2 goal-achieved tasks ($\nabla f_{1,2}^s$) and 2 goal-unachieved tasks ($\nabla f_{1,2}^u$). (left) MOO; (middle) MOO on the set of goal-unachieved tasks only; (right) TA-MOO with a solution focuses more on the goal-unachieved tasks.

### B.2 Illustrations of How MOO and TA-MOO Work

Figure 1 illustrates solutions of MOO and TA-MOO in a scenario of 2 goal-achieved tasks (with corresponding gradients $\nabla f_{1,2}^s$) and 2 goal-unachieved tasks (with corresponding gradients $\nabla f_{1,2}^u$). As illustrated in the left figure, with standard MOO method, where all the tasks' gradients have been considered regardless their status and the solution associated with the minimal norm is the perpendicular vector as suggested by geometry (Sener & Koltun, 2018). If considering the goal-unachieved tasks only as in the middle case, the MOO solution is the edge case. However, this extreme strategy ignores all the goal-achieved tasks which might lead to the instability. The Task Oriented regularization strikes a balance between the two aforementioned strategies as illustrated in the right figure. The method focuses more on improving the goal-unachieved tasks while spend less effort to maintain the goal-achieved tasks. With $\lambda = 0$ the TA-MOO optimal solution is equivalent to the standard MOO optimal solution while it becomes the MOO solution in the case of the goal-unachieved tasks only when $\lambda \to \infty$.

## C Experimental settings

**General settings.** Through our experiments, we use six common architectures including ResNet18 (He et al., 2016), VGG16 (Simonyan & Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), EfficientNet (B0) (Tan & Le, 2019), MobileNet Howard et al. (2017), and WideResNet (with depth 34 and widen factor 10) Zagoruyko & Komodakis (2016) with the implementation of `https://github.com/kuangliu/pytorch-cifar`. We evaluate on the full testing set (10k) of two benchmark datasets which are CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). More specifically, the two datasets have 50k training images and 10k testing images, respectively, with the same image resolution of $32 \times 32 \times 3$. However, while the CIFAR10 dataset has 10 classes, the CIFAR100 dataset has 100 classes and fewer images per class. Therefore, in general, an adversary is easier to attack a CIFAR100 model than a CIFAR10 one as shown in Table 7. We observed that the attack performance is saturated with standard training models. Therefore, to make the job of adversaries more challenging, we use Adversarial Training with PGD-AT (Madry et al., 2018) to robustify the models and use these robust models as victim models in our experiments. Specifically, we use the SGD optimizer (momentum 0.9 and weight decay $5 \times 10^{-4}$) and Cosine Annealing Scheduler to adjust the learning rate with an initial value of 0.1 and train a model in 200 epochs as suggested in the implementation above. We use PGD-AT $L_\infty$ (Madry et al., 2018) with the same setting for both CIFAR10 and CIFAR100 datasets, i.e., perturbation limitation $\epsilon = 8/255$, $k = 20$ steps, and step size $\eta = 2/255$.

**Method settings.** In this work, we evaluate all the methods in the untargeted attack setting with $L_\infty$ norm. The attack parameters are the same among methods, i.e., number of attack steps 100, attack budget $\epsilon = 8/255$ and step size $\eta_\delta = 2/255$. In our method, we use $K$=10 to update the weight in each step with learning rate $\eta_w = 0.005$. Tradeoff parameter $\lambda = 100$ in all experiments. In MinMax (Wang et al., 2021), we use the same $\gamma = 3$ for all settings and use the authors' implementation [2].

[2] https://github.com/wangjksjtu/minmax-adv

Table 7: Robustness performance of models in the experiments

| | CIFAR10 | | CIFAR100 | |
|---|---|---|---|---|
| | Nat-Acc | Adv-Acc | Nat-Acc | Adv-Acc |
| ResNet18 | 86.47 | 42.14 | 59.64 | 18.62 |
| VGG16 | 84.24 | 40.88 | 55.27 | 16.41 |
| GoogLeNet | 88.26 | 41.26 | 63.10 | 19.16 |
| EfficientNet | 74.52 | 41.36 | 57.67 | 19.90 |
| MobileNet | 76.52 | 31.12 | - | - |
| WideResNet | 88.13 | 48.62 | - | - |

Table 8: Data transformation setting. $U$ represents uniform sampling function and $p$ represents probability to excuse a transformation (e.g., flipping).

| | Deterministic | Stochastic |
|---|---|---|
| Identity | Identity | Identity |
| Horizontal flip | $p = 1$ | $p = 0.5$ |
| Vertical flip | $p = 1$ | $p = 0.5$ |
| Center crop | scale $= 0.6$ | scale $= U(0.6, 1.0)$ |
| Adjust brightness | factor $= 1.3$ | factor $= U(1.0, 1.3)$ |
| Rotation | angle $= 10$ | angle $= U(-10\ , 10\ )$ |
| Adjust gamma | gamma $= 1.3$ | $gamma = U(0.7, 1.3)$ |

**Attacking ensemble model settings.** In our experiment, we use an ensemble of four adversarially trained models: ResNet18, VGG16, GoogLeNet, and EfficientNet. The architecture is the same for both the CIFAR10 and CIFAR100 datasets except for the last layer which corresponds with the number of classes in each dataset. The final output of the ensemble is an average of the probability outputs (i.e., output of the softmax layer). We use three different losses as an object for generating adversarial examples including Cross Entropy (CE) (Madry et al., 2018), Kullback-Leibler divergence (KL) (Zhang et al., 2019), and CW loss (Carlini & Wagner, 2017).

**Universal perturbation settings.** We follow the experimental setup in Wang et al. (2021), such that the full test set (10k images) is randomly divided into equal-size groups (K images per group). The comparison has been conducted on the CIFAR10 and CIFAR100 datasets, and CW loss. We use adversarial trained ResNet18, VGG16 and EfficientNet as base models. We observed that the ASR-All was mostly zero, indicating that it is difficult to generate a general perturbation for all data points. Therefore, we use ASR-Avg to compare the performances of the methods.

**Robust adversarial examples against transformations settings.** In our experiment, we use 7 common data transformations including I-Identity, H-Horizontal flip, V-Vertical flip, C-Center crop, B-Adjust brightness, R-Rotation, and G-Adjust gamma. The parameter setting for each transformation has been shown in Table 8. In the deterministic setting, a transformation has been fixed with one specific parameter, e.g., center cropping with a scale of 0.6 or adjusting brightness with a factor of 1.3. While in the stochastic setting, a transformation has been uniformly sampled from its family, e.g., center cropping with a random scale in range (0.6, 1.0) or adjusting brightness with a random factor in range (1.0, 1.3). The experiment has been conducted on adversarially trained ResNet18 model with the CW loss.

# D Additional Experiments

## D.1 Transferability of adversarial examples in the ENS setting

We conduct an additional experiment to evaluate the transferability of our adversarial examples. We use an ensemble (RME) of three models: ResNet18, MobileNet, and EfficientNet as a source model and apply different adversaries to generate adversarial examples to this ensemble. We then use these adversarial examples to attack other ensemble architectures (target models), for example, RMEVW is an ensemble of 5 models including ResNet18, MobileNet, EfficientNet, VGG16 and

Table 9: Evaluation on the Transferability of adversarial examples. Each cell (row-ith, column-jth) reports SAR (higher is better) of adversarial examples from the same source architecture (RME) with an adversary at row-ith to attack an ensemble at column-jth. Each architecture has been denoted by symbols such as R: ResNet18, M: MobileNet, E: EfficientNet, V: VGG16, W: WideResNet. For examples, RME represents for an ensemble of ResNet18, MobileNet and EfficientNet. The highest/second highest performance is highlighted in **Bold**/Underline. The table is copied from Table 3 in the main paper for reading comprehension purpose.

| | RME | RVW | EVW | MVW | REV | MEV | RMEV | RMEVW |
|---|---|---|---|---|---|---|---|---|
| Uniform | 31.73 | 25.03 | 22.13 | 22.73 | 29.50 | 28.44 | 26.95 | 20.50 |
| MinMax | 40.01 | 23.75 | 22.39 | 23.34 | 32.57 | 32.75 | 31.85 | 21.99 |
| MOO | 35.20 | 24.25 | 22.94 | 23.76 | 30.65 | 32.28 | 29.49 | 21.77 |
| TA-MOO | **40.97** | **25.13** | **23.59** | **24.38** | **33.00** | **33.05** | **32.14** | **23.04** |

WideResNet. Table 9 reports the SAR-All metric of transferred adversarial examples, where a higher number indicates a higher success rate of attacking a target model, therefore, also implies a higher transferability of adversarial examples. The first column (heading RME) shows SAR-All when adversarial examples attack the source model (i.e., the whitebox attack setting).

**The Uniform strategy achieves the lowest transferability.** It can be observed from Table 9 that the Uniform strategy achieves the lowest SAR in the whitebox attack setting. This strategy also has the lowest transferability in attacking other ensembles (except an ensemble RVW).

**MinMax's transferability drops on dissimilar target models.** While MinMax achieves the second-best performance in the whitebox attack setting, its adversarial examples have a low transferability when target models are different from the source model. For example, in the target model RVW where there is only one member of the target model from the source model (RME) (i.e., R or ResNet18), MinMax achieves a 23.75% success rate which is lower than the Uniform strategy by 1.28%. Similar observation can be observed on target models EVW and MVW, where MinMax outperforms the Uniform strategy by just 0.2% and 0.6%, respectively.

**TA-MOO achieves the highest transferability on a diverse set of ensembles** . Our TA-MOO adversary achieves the highest attacking performance on the whitebox attack setting, with a huge gap of 9.24% success rate over the Uniform strategy. Our method also achieves the highest transferability regardless diversity of a target ensemble. More specifically, on target models such as REV, MEV, and RMEV, where members in the source ensemble (RME) are also in the target ensemble, our TA-MOO significantly outperforms the Uniform strategy, with the highest improvement is 5.19% observed on target model RMEV. On the target models EVW and MVW which are less similar to the source model, our method still outperforms the Uniform strategy by 1.46% and 1.65%. The superior performance of our adversary on the transferability shows another benefit of using multi-objective optimization in generating adversarial examples. By reaching the intersection of all members' adversarial regions, our adversary is capable to generate a common vulnerable pattern on an input image shared across architectures, therefore, increasing the transferability of adversarial examples.

### D.2 Adversarial Training with TA-MOO

**Setting.** We conduct adversarial training with adversarial examples generated by our MOO and TA-MOO attacks to verify the quality of these adversarial examples. We choose an ensemble of 3 MobileNet architectures (non-diverse set) and ensemble of 3 different architectures including ResNet18, MobileNet and EfficientNet (diverse set). To evaluate the adversarial robustness, we compare natural accuracy (NAT) and robust accuracy (ADV) against PGD-Linf attack of these adversarial training methods (the higher the better). We also measure the success attack rate (SAR) of adversarial examples generated by the same PGD-Linf attack on fooling each single member and all members of the ensemble (the lower the better). We use $k = 10, \epsilon = 8/255, \eta = 2/255$ for adversarial training and PGD-Linf with $k = 20, \epsilon = 8/255, \eta = 2/255$ for robustness evaluation.

Table 10: Robustness evaluation of Adversarial Training methods on the CIFAR10 dataset. RME represents an ensemble of ResNet18 (R), MobileNet (M) and EfficientNet E), while MobiX3 represents an ensemble of three MobileNets. NAT and ADV measure the natural accuracy and the robust accuracy against PGD-Linf attack (↑the higher the better). Other metrics measure the success attack rate (SAR) of adversarial examples generated by the same PGD-Linf attack on fooling each single member and all members of the ensemble (↓the lower the better). The highest/second highest **robustness** is highlighted in **Bold**/Underline. The most important metric is emphasized in blue.

| | Arch | NAT↑ | ADV↑ | A-All↓ | A-Avg↓ | R/M1↓ | M/M2↓ | E/M3↓ |
|---|---|---|---|---|---|---|---|---|
| PGD-AT | MobiX3 | **80.43** | 32.78 | 54.34 | 73.89 | 76.17 | 74.35 | 71.14 |
| MinMax-AT | MobiX3 | 79.01 | 37.28 | 50.28 | **66.77** | **65.27** | 70.27 | **64.78** |
| MOO-AT | MobiX3 | 79.38 | 33.04 | **46.28** | 74.36 | 71.25 | 74.53 | 77.29 |
| TA-MOO-AT | MobiX3 | 79.22 | **38.22** | 48.21 | 67.83 | 68.04 | **67.37** | 68.07 |
| PGD-AT | RME | **86.52** | 37.36 | 49.01 | 69.75 | 65.81 | 75.24 | 68.21 |
| MinMax-AT | RME | 83.16 | 40.40 | 46.91 | 65.73 | 65.22 | **68.28** | 63.70 |
| MOO -AT | RME | 82.04 | 37.48 | 45.24 | 70.11 | 69.00 | 75.43 | 65.90 |
| TA-MOO-AT | RME | 82.59 | **41.32** | **43.68** | **65.09** | **63.77** | 68.98 | **62.51** |

We use SGD optimizer with momentum 0.9 and weight decay 5e-4. Initial learning rate is 0.1 with Cosine Annealing scheduler and train on 100 epochs.

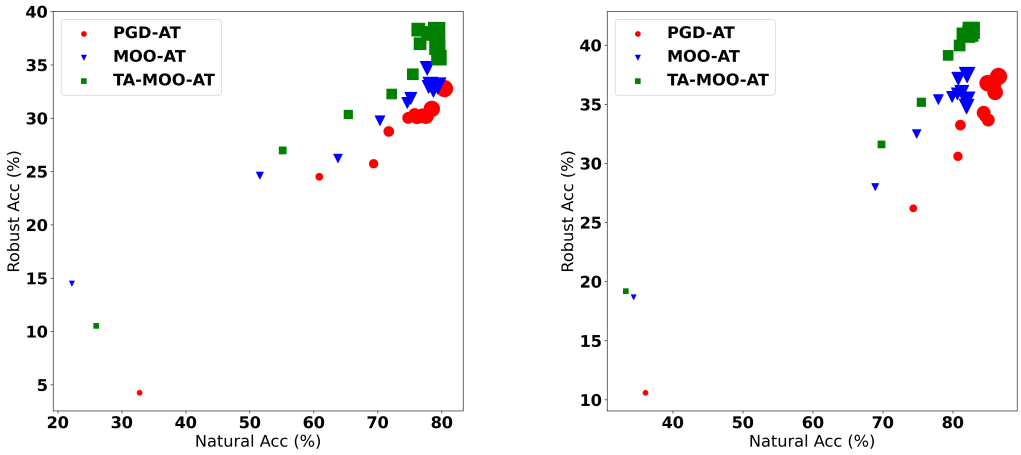
**Result 1. Reducing transferability.** It can be seen that the SAR-All of MOO-AT and TA-MOO-AT are much lower than that on other methods. More specifically, the gap of SAR-All between PGD-AT and TA-MOO-AT is (5.33%) 6.13% on the (non) diverse setting. The lower SAR-All indicating that adversarial examples are harder to transfer among ensemble members on the TA-MOO-AT model than on the PGD-AT model.

**Result 2. Producing more robust single members.** The comparison of average SAR shows that adversarial training with TA-MOO produces more robust single models than PGD-AT does. More specifically, the average robust accuracy (measured by 100% - *A-Avg*) of TA-MOO-AT is 32.17%, an improvement of 6.06% over PGD-AT in the non-diverse setting, while there is an improvement of 4.66% in the diverse setting.

**Result 3. Adversarial training with TA-MOO achieves the best robustness.** More specifically, on the non-divese setting, TA-MOO-AT achives 38.22% robust accuracy, an improvement of 1% over MinMax-AT and 5.44% over standard PGD-AT. On the diverse setting, the improvement over MinMax-AT and PGD-AT are 0.9% and 4%, respectively. The root of the improvement is the ability to generate stronger adversarial examples in the the sense that they can challenge not only the entire ensemble model but also all single members. These adversarial examples lie in the joint insecure region of members (i.e., the low confidence region of multiple classes), therefore, making the decision boundaries more separate. As a result, adversarial training with TA-MOO produces more robust single models (i.e., lower SAR-Avg) and significantly reduces the transferability of adversarial examples among members (i.e., lower SAR-All). These two conditions explain the best ensemble adversarial robustness achieved by TA-MOO.

### D.3 Universal Perturbation (UNI)

**Additional experimental results.** In addition to the experiments on ResNet18 as reported in Table 5, we would like to provide additional experimental results on two other adversarial trained models VGG16 and EfficientNet as shown in Table 11. It can be seen that, TA-MOO consistently achieves the best attacking performance on ResNet18 and VGG16, on both CIFAR10 and CIFAR100 datasets, with $K \geq 8$.




(a) MobiX3

(b) RME

Figure 2: Comparison progress of three adversarial training methods. The bigger marker size represents the later epoch. Each point represents the natural accuracy and robust accuracy against PGD-Linf attack on the testing set.

Table 11: Evaluation of generating Universal Perturbation on the CIFAR10 and CIFAR100 datasets. R: ResNet18, V: VGG16, E: EfficientNet.

| | | CIFAR10 | | | | | CIFAR100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | K=4 | K=8 | K=12 | K=16 | K=20 | K=4 | K=8 | K=12 | K=16 | K=20 |
| R | Uniform | 37.52 | 30.34 | 27.41 | 25.52 | 24.31 | 65.40 | 58.99 | 55.33 | 53.02 | 51.49 |
| | MinMax | **50.13** | 33.68 | 20.46 | 15.74 | 14.73 | **74.73** | 62.29 | 52.05 | 45.26 | 42.33 |
| | MOO | 43.80 | 35.92 | 31.41 | 28.75 | 26.83 | 69.35 | 62.72 | 57.72 | 54.12 | 52.25 |
| | TA-MOO | 48.00 | **39.31** | **34.96** | **31.84** | **30.12** | 72.74 | **68.06** | **62.33** | **57.48** | **54.12** |
| V | Uniform | 37.76 | 30.81 | 27.49 | 25.94 | 24.46 | 66.87 | 61.49 | 58.53 | 56.29 | 54.98 |
| | MinMax | **47.96** | 30.88 | 20.20 | 16.93 | 16.25 | **78.58** | 69.14 | 58.85 | 51.81 | 48.09 |
| | MOO | 43.04 | 34.56 | 30.07 | 27.43 | 25.42 | 73.46 | 66.51 | 61.28 | 57.88 | 56.09 |
| | TA-MOO | 46.58 | **38.33** | **32.32** | **29.16** | **26.56** | 75.57 | **71.86** | **67.22** | **62.99** | **59.19** |
| E | Uniform | 44.86 | 39.03 | 36.37 | 34.65 | 33.49 | 67.55 | 60.99 | 57.35 | **54.84** | **53.57** |
| | MinMax | 44.47 | 32.96 | 28.86 | 27.01 | 26.47 | 69.69 | 57.99 | 50.93 | 45.59 | 43.87 |
| | MOO | 45.31 | **39.28** | **36.44** | **34.72** | **33.51** | 66.68 | 59.69 | 54.95 | 53.20 | 51.43 |
| | TA-MOO | **46.74** | 37.95 | 33.95 | 31.71 | 30.41 | **70.40** | **63.78** | **58.17** | 53.26 | 50.66 |

Table 12: Evaluation of generating Universal Perturbation (K=8) on the CIFAR10 dataset with ResNet18 architecture and MOO method. $\{T_i\}_{i=1}^K$ represents value for each task (i.e., a sample in a group). $w_1/w_2$ represents the weight of the first/second group of K samples, while $w$ represents the the statistic of weight over all groups (mean$\pm$std). $\|\nabla_{\delta_1} f_i(\delta_1)\|$ / $\|\nabla_{\delta_2} f_i(\delta_2)\|$ represents the gradient norm of the first/second group of K samples, while $\|\nabla_\delta f_i(\delta)\|$ represents the statistic of gradient norm over all groups (mean$\pm$std). $\mathbb{I}_0/\mathbb{I}_1$ represents the indicator function for a successful (1) or unsuccessful (0) task, while $\mathbb{I}$ represents the the statistic of successful rate over all groups.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|
| $\|\nabla_{\delta_1} f_i(\delta_1)\|$ | 1.15e1 | 3.45e-5 | 1.97e-2 | 1.26e-4 | 1.27e0 | 1.04e-1 | 1.04e1 | 9.91e0 |
| $w_0$ | 0.0238 | 0.1861 | 0.1859 | 0.1861 | 0.1763 | 0.1862 | 0.0257 | 0.0299 |
| $\mathbb{I}_0$ | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $\|\nabla_{\delta_2} f_i(\delta_2)\|$ | 9.70e0 | 1.59e1 | 4.32e-4 | 4.27e-4 | 1.25e1 | 6.23e-5 | 2.91e-5 | 6.17e-6 |
| $w_1$ | 0.0341 | 0.0167 | 0.1854 | 0.1854 | 0.0222 | 0.1854 | 0.1854 | 0.1854 |
| $\mathbb{I}_1$ | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| $\|\nabla_\delta f_i(\delta)\|$ | 4.93$\pm$6.63 | 4.23$\pm$6.97 | 5.18$\pm$7.42 | 3.84$\pm$5.83 | 4.39$\pm$6.04 | 6.66$\pm$7.64 | 4.82$\pm$7.48 | 5.25$\pm$7.17 |
| $w$ | 0.12$\pm$0.08 | 0.14$\pm$0.09 | 0.12$\pm$0.08 | 0.13$\pm$0.08 | 0.12$\pm$0.09 | 0.10$\pm$0.08 | 0.14$\pm$0.10 | 0.11$\pm$0.08 |
| $\mathbb{I}$ | 0.38$\pm$0.49 | 0.28$\pm$0.46 | 0.36$\pm$0.48 | 0.32$\pm$0.48 | 0.38$\pm$0.48 | 0.48$\pm$0.50 | 0.32$\pm$0.47 | 0.40$\pm$0.49 |

**Why does MOO work?** As shown in Table 11, MOO consistently achieves better performance than the Uniform strategy (except for the setting with EfficientNet on the CIFAR100 dataset). To find out the reason for the improvement, we investigate the gradient norm $\|\nabla_\delta f(\delta)\|$ and weight $w$ for the first, and second groups (as an example) and the average over 100 groups of the testset as shown in Table 12. It can be seen that in the first and second groups, there are some tasks that have significantly low gradient strengths than other tasks. The gap of the strongest/weakest gradient strength can be a magnitude of $10^6$ indicating the domination of one task over others. While this issue can cause the failure as in the ENS setting, however, in the UNI setting, the lowest gradient strengths in each group correspond to unsuccessful tasks (unsuccessful adversarial examples) and vice versa. Recall that we use the multi-gradient descent algorithm to solve MOO, which in principle assigns a higher weight for a weaker gradient vector. Therefore, in the UNI setting, while the dominating issue still exists, fortunately, the result still fits our desired weighting strategy (i.e., higher weight for an unsuccessful task and vice versa). Moreover, when there are a large number of groups (i.e., 100 groups), the issue of dominating tasks is alleviated. The average gradient strength is more balanced as shown in Table 12. This explains the improvement of MOO over the Uniform strategy in the UNI setting.

### D.4 Robust Adversarial Examples against Transformations (EoT)

We observed that in EoT with the stochastic setting, adjusting gamma sometimes has the overflow issue resulting in an infinite gradient. Recall that our method using MGDA to solve MOO which relies on the stability of gradient strengths. Therefore, in the case of having infinite gradients, learning weight $w$ is unstable, resulting to lower performance in both MOO and TA-MOO.

To overcome the overflow issue, we allocate memory to cache the valid gradient of each task in the previous iteration and replace the infinite value in the current iteration with the valid one in the memory. The storage only requires a tensor with the same shape as the gradient (i.e., as the exact size of the input), therefore, it does not increase the computation resource significantly. As shown in Table 13, this simple technique helps to improve performance of TA-MOO by 5.3% on both the CIFAR10 and CIFAR100 datasets. It also helps to improve performance of MOO by 0.8% and 4.8%, respectively. Finally, after overcoming the gradient issue, the TA-MOO achieves the best performance on the CIFAR100 dataset and the second best performance on the CIFAR10 dataset (0.4% lower in ASR-All but 0.8% higher in ASR-Avg when comparing to MinMax). This result provides additional evidence of the advantage of our method.

Table 13: Robust adversarial examples against transformations evaluation. The highest/second highest performance is highlighted in **Bold**/Underline. MOO$^\star$ and TA-MOO$^\star$ represent version with memory to overcome the infinite gradient issue in the stochastic setting.

| | | Deterministic | | Stochastic | |
|---|---|---|---|---|---|
| | | A-All | A-Avg | A-All | A-Avg |
| C10 | Uniform | 25.98 | **55.33** | 31.47 | **50.55** |
| | MinMax | 30.54 | 52.20 | **33.35** | 49.44 |
| | MOO | 21.25 | 49.81 | 26.97 | 43.84 |
| | TA-MOO | **31.10** | 55.26 | 28.26 | 45.67 |
| | MOO$^\star$ | - | - | 27.79 | 45.91 |
| | TA-MOO$^\star$ | - | - | 32.96 | 50.27 |
| C100 | Uniform | 56.19 | 76.23 | 59.89 | 73.73 |
| | MinMax | 59.75 | 75.72 | 61.30 | 73.59 |
| | MOO | 53.17 | 74.21 | 54.96 | 69.26 |
| | TA-MOO | **60.88** | **76.71** | 56.23 | 69.91 |
| | MOO$^\star$ | - | - | 58.79 | 72.81 |
| | TA-MOO$^\star$ | - | - | **61.54** | **74.07** |

Table 14: Robust adversarial examples against transformations evaluation. The highest/second highest performance is highlighted in **Bold**/Underline. The most important metric is emphasized in blue color. MOO$^\star$ and TA-MOO$^\star$ represent version with memory to overcome the infinite gradient issue in the stochastic setting. I: Identity, H: Horizontal flip, V: Vertical flip, C: Center crop, G: Adjust gamma, B: Adjust brightness, R: Rotation.

| | | A-All | A-Avg | I | H | V | C | G | B | R |
|---|---|---|---|---|---|---|---|---|---|---|
| D-C10 | Uniform | 25.98 | **55.33** | **44.85** | 41.58 | 82.90 | **72.56** | **45.92** | **49.59** | **49.93** |
| | MinMax | 30.54 | 52.20 | 43.31 | 41.59 | 78.80 | 64.83 | 44.38 | 46.53 | 45.97 |
| | MOO | 21.25 | 49.81 | 36.23 | 33.93 | **87.47** | 71.05 | 37.68 | 40.21 | 42.12 |
| | TA-MOO | **31.10** | 55.26 | 44.15 | **41.86** | 85.19 | 71.86 | 45.53 | 48.70 | 49.54 |
| D-C100 | Uniform | 56.19 | 76.23 | **70.43** | 69.01 | 87.66 | 87.36 | 71.40 | 74.25 | 73.47 |
| | MinMax | 59.75 | 75.72 | 70.13 | 69.26 | 87.45 | 86.03 | 71.54 | 73.30 | 72.32 |
| | MOO | 53.17 | 74.21 | 66.96 | 65.68 | **89.16** | 87.03 | 68.49 | 71.11 | 71.06 |
| | TA-MOO | **60.88** | **76.71** | **70.43** | **69.37** | 89.11 | **87.95** | **71.70** | **74.73** | **73.69** |
| S-C10 | Uniform | 31.47 | **50.55** | **48.58** | 44.70 | **65.52** | 51.14 | **47.43** | **48.76** | 47.70 |
| | MinMax | **33.35** | 49.44 | 47.35 | 44.45 | 62.78 | 51.75 | 46.32 | 47.13 | 46.34 |
| | MOO | 26.97 | 43.84 | 40.62 | 38.45 | 57.65 | 48.55 | 40.41 | 40.71 | 40.47 |
| | TA-MOO | 28.26 | 45.67 | 42.80 | 39.66 | 61.98 | 47.92 | 41.80 | 43.01 | 42.54 |
| | MOO$^\star$ | 27.79 | 45.91 | 42.43 | 39.65 | 62.11 | 51.44 | 41.62 | 42.21 | 41.92 |
| | TA-MOO$^\star$ | 32.96 | 50.27 | 48.18 | **45.26** | 62.97 | **52.49** | 47.03 | 48.22 | **47.76** |
| S-C100 | Uniform | 59.89 | 73.73 | **73.19** | **71.15** | 79.73 | 74.81 | 72.05 | 73.10 | **72.10** |
| | MinMax | 61.30 | 73.59 | 72.44 | 70.55 | 80.04 | 75.55 | 71.99 | 72.49 | **72.10** |
| | MOO | 54.96 | 69.26 | 67.62 | 66.11 | 75.88 | 72.72 | 66.87 | 68.11 | 67.49 |
| | TA-MOO | 56.23 | 69.91 | 68.52 | 66.92 | 76.70 | 72.71 | 67.57 | 68.97 | 67.97 |
| | MOO$^\star$ | 58.79 | 72.81 | 71.58 | 69.08 | 80.17 | 75.01 | 70.78 | 71.71 | 71.33 |
| | TA-MOO$^\star$ | **61.54** | **74.07** | 72.95 | 70.95 | **80.94** | **76.22** | **72.22** | **73.21** | 72.00 |

Table 15: Average time per sample for generating adversarial example. All experiments are measured on the CIFAR10 dataset, EoT and Uni are with ResNet18 architecture.

| | Ensemble (K=4) | EoT (K=7) | Uni@K=12 | Uni@K=20 |
|---|---|---|---|---|
| Uniform | 640ms | 350ms | 1850ms | 3030ms |
| MinMax | 1540ms | 610ms | 1210ms | 2080ms |
| MOO | 1770ms | 1130ms | 5600ms | 9280ms |
| TA-MOO | 1960ms | 1200ms | 5870ms | 9500ms |

Table 16: Stability of experiments' evaluation on different random seeds. Experiment on the ENS setting, with an ensemble of 4 models: Resnet18, VGG16, GoogleNet and EfficientNet.

| | A-All | A-Avg | R | V | G | E |
|---|---|---|---|---|---|---|
| Uniform | $28.12 \pm 0.09$ | $48.29 \pm 0.05$ | $48.81 \pm 0.08$ | $49.06 \pm 0.08$ | $48.27 \pm 0.10$ | $47.06 \pm 0.03$ |
| MOO | $25.61 \pm 0.36$ | $45.13 \pm 0.30$ | $39.84 \pm 0.62$ | $47.29 \pm 0.36$ | $37.51 \pm 0.36$ | $55.90 \pm 0.17$ |
| TA-MOO | $37.56 \pm 0.32$ | $51.15 \pm 0.21$ | $49.37 \pm 0.15$ | $52.80 \pm 0.45$ | $48.98 \pm 0.25$ | $53.24 \pm 0.13$ |

### D.5 Generating Speed Comparison and Experiments' Stability

**Generating Speed Comparison.** Table 15 shows the average time to generate one adversarial example in each setting. The results are measured on the CIFAR10 dataset with ResNet18 architecture in the Ensemble of Transformations (EoT) and Universal Perturbation (Uni) settings. We use 1 Titan RTX 24GB for the EoT experiment and 4 Tesla V100 16GB each for the other experiments. It is worth mentioning that our primary focus in this paper is showing the advantage of MOO and the Task-Oriented regularization in generating adversarial examples. Therefore, we did not try to optimize our implementation in terms of generating time.

**Experiments' Stability.** We conduct an experiment with 5 different random seeds to generate adversarial examples for the ENS setting to evaluate the stability of experimental results on choosing of random seed. The experiment is on the CIFAR10 dataset, with an ensemble of 4 architectures including ResNet18, VGG16, GoogLeNet, and EfficientNet. We report mean and variation values in Table 16. It can be observed that there is a slight variation in attack performances across methods. The variation is small enough compared to the gap between methods (i.e., the biggest variation is 0.32% in SAR-All while the smallest gap is 2.51% between MOO and the Uniform approach), therefore, making the comparison still reliable.

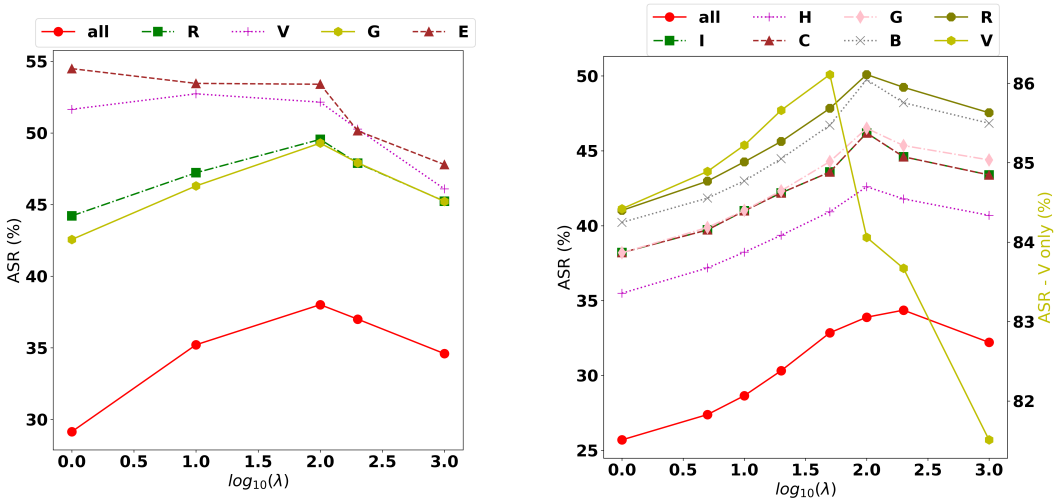
### D.6 Sensitivity to Hyper-parameters

In this section we provide an analytical experiment on the sensitivity of our TA-MOO method to the tradeoff $\lambda$. The study has been conducted with the ENS setting with CE loss and the EoT setting with deterministic transformations using ResNet18 architecture. All experiments are on the CIFAR10 dataset. The value of $\lambda$ is changed from 1 to 1000. It can be observed from Figure 3a (the ENS setting) that (i) increasing $\lambda$ reduces the performance of dominated task (i.e., ASR on the EfficientNet decreases from 54.49% at $\lambda = 1$ to 53.40% at $\lambda = 100$) while increases performances of other tasks. In overall, it significantly increases the ASR-All performance of the entire ensemble from 29.14% at $\lambda = 1$ to 38.01% at $\lambda = 100$. (ii) However, over-high $\lambda$ (i.e., $\lambda > 200$) leads to the drop of performance in all tasks, resulting in a lower overall performance.

A similar observation can be seen in the EoT setting in Figure 3b. The attack performance on the dominated task (V-Vertical flipping) decreases from 86.11% at $\lambda = 50$ to 83.67% at $\lambda = 200$. In contract, in the same range of $\lambda$ the overall performance increases from 32.85% to 34.36%. The performances of all tasks decrease when using too large $\lambda$ (i.e., $\lambda > 200$). Based on the result of this study, we choose $\lambda = 100$ in all the other experiments.

### D.7 Comparison with Standard Attacks

We conducted an additional comparison on the ENS setting to further confirm the effectiveness of our method over standard adversarial attacks (which consider an entire ensemble as a single model). More specifically, we compare with AutoAttack (Croce & Hein, 2020), Brendel-Bethge attack (BB)




(a) ENS setting

(b) EoT setting

Figure 3: Sensitivity to the parameter $\lambda$.

Table 17: Attacking Ensemble model with a diverse set D={R-ResNet18, V-VGG16, G-GoogLeNet, E-EfficientNet} and non-diverse set ND={4 ResNets}. Experiment on the CIFAR10 dataset with cross-entropy objective loss. The most important metric is emphasized in blue.

| | | A-All | A-Avg | R/R1 | V/R2 | G/R3 | E/R4 |
|---|---|---|---|---|---|---|---|
| D | PGD | 28.21 | 48.34 | 48.89 | 49.08 | 48.38 | 47.03 |
| | CW | 6.10 | 16.63 | 13.53 | 15.76 | 11.74 | 25.47 |
| | B&B | 6.67 | 38.03 | 37.95 | 38.92 | 35.58 | 39.68 |
| | AutoAttack | 30.71 | 45.49 | 48.32 | 45.83 | 47.25 | 40.56 |
| | MOO | 25.16 | 44.76 | 39.06 | 46.83 | 37.05 | **56.11** |
| | TA-MOO | **38.01** | **51.10** | **49.55** | **52.15** | **49.29** | 53.40 |
| ND | PGD | 28.17 | 48.75 | 51.94 | 45.55 | 54.15 | 43.34 |
| | CW | 4.71 | 13.86 | 14.92 | 12.71 | 17.51 | 10.31 |
| | B&B | 5.29 | 40.51 | 49.06 | 35.19 | 48.63 | 29.16 |
| | AutoAttack | 37.00 | 49.32 | 51.07 | 48.58 | 51.08 | 46.55 |
| | MOO | 32.50 | 52.21 | 53.25 | 49.05 | 56.80 | 49.76 |
| | TA-MOO | **41.01** | **57.33** | **58.88** | **55.32** | **60.81** | **54.29** |

(Brendel et al., 2019), Carlini-Wagner attack (CW) (Carlini & Wagner, 2017), and PGD attack (Madry et al., 2018). For AutoAttack, we use the standard version which includes 4 different attacks. For BB attack, we initialized with the PGD attack with 20 steps. For CW attack, we set the confidence factor to 1.0. We evaluate these attacks on 2 ensemble settings, a diverse (D) ensemble set with 4 different architectures (ResNet18, VGG16, GoogLeNet, and EfficientNet) and a non-diverse (ND) ensemble set with 4 ResNet18 architectures.

It can be seen from the Table 17 that our TA-MOO attack consistently achieves the best attack performance, with a significant gap compared to the best standard attack. More specifically, our TA-MOO method achieves 38.01% (SAR-All metric) on the diverse ensemble set, while the second best attack is AutoAttack with 30.71% (a gap of 7.3%). On the non-diverse set, the gap between our TA-MOO and AutoAttack is still notably large at 4%. These standard attacks consider an entire ensemble as a single model, i.e., aim to optimize a single objective given a single ensemble output. Therefore, they cannot guarantee a successful attack on each member.

Table 18: Evaluation attacking performance on the ImageNet dataset. RE/LE/TAR/UNTAR represents Robust Ensemble/Less-Robust Ensemble/Targeted Attack/Untargeted Attack, respectively. R18/R50/STD represents robust ResNet18, robust ResNet50 and standard ResNet50 pre-trained model, respectively. The most important metric is emphasized in blue.

| | | A-All | A-Avg | R18/R18 | R50/R50 | R50/STD |
|---|---|---|---|---|---|---|
| RE-TAR | Uniform | 29.58 | 39.38 | 42.50 | 32.22 | 43.42 |
| | MOO | 29.66 | **39.73** | 42.86 | **32.32** | 44.00 |
| | TA-MOO | **29.68** | **39.73** | **42.90** | 32.26 | **44.02** |
| LE-TAR | Uniform | 30.30 | 58.14 | 42.36 | 32.06 | **100.0** |
| | MOO | 30.66 | **58.37** | **42.70** | **32.48** | 99.94 |
| | TA-MOO | **30.68** | 58.25 | 42.54 | 32.36 | 99.86 |
| RE-UNTAR | Uniform | 48.58 | 60.11 | 64.22 | 51.72 | 64.38 |
| | MOO | 48.68 | **60.20** | **64.30** | 51.82 | **64.48** |
| | TA-MOO | **49.80** | 59.71 | 63.80 | **52.38** | 62.94 |
| LE-UNTAR | Uniform | 34.24 | 61.01 | 46.98 | 36.28 | 99.78 |
| | MOO | 44.76 | 68.29 | 58.42 | 46.64 | 99.80 |
| | TA-MOO | **49.46** | **70.74** | **61.26** | **51.14** | **99.82** |

### D.8 Attacking the ImageNet dataset

**Experimental Setting.** We conduct experiments on the ENS setting using the adversarial pre-trained models on the RobustBench (Croce et al., 2021). We use two sets of an ensemble to verify the importance of our task-oriented strategy. The first set is the robust ensemble (RE) set including 3 robust models: ResNet18 (model ID: Salman2020Do_R18 (Salman et al., 2020), robust accuracy 25.32%), ResNet50 (model ID: Salman2020Do_R50 (Salman et al., 2020), robust accuracy 34.96%) and ResNet50 (model ID: Wong2020Fast (Wong et al., 2019), robust accuracy 26.24%). The second set is the less-robust ensemble (LE) which includes 3 models: ResNet18 (model ID: Salman2020Do_R18), ResNet50 (model ID: Salman2020Do_R50) and the standard training ResNet50 (model ID: Standard_R50, robust accuracy 0%). We use both targeted attack and untargeted attack settings, with $\epsilon = 4/255$ , and $\eta = 1/255$ with 20 steps. We use 5000 images of the validation set to evaluate.

**Experimental Results.** We report experimental results with different settings in Table 18, where RE/LE/TAR/UNTAR represents Robust Ensemble/Less-Robust Ensemble/Targeted Attack/Untargeted Attack, respectively. It can be seen that, in the robust ensemble setting (RE-TAR and RE-UNTAR), our MOO achieves a similar performance compared to the baseline, while TA-MOO has a further improvement over MOO. The gap of SAR-All between TA-MOO and the uniform weighting strategy is 0.1% in the targeted attack setting (RE-TAR), while that in the untargeted attack setting is 1.2%. In the less-robust ensemble setting (LE-TAR and LE-UNTAR), the improvement of our methods over the baseline is higher than in the robust ensemble setting. With the gap of SAR-All between TA-MOO and the uniform strategy is 0.38% with the targeted attack setting (LE-TAR), while the gap in the untargeted setting (LE-UNTAR) is 15.22% a significantly higher. While it is acknowledged that the targeted attack is a more common protocol in attacking the ImageNet dataset (Athalye et al., 2018), however, we believe that our significant improvement on the untargeted attack is still worth noting.

We conduct an additional experiment on the EoT setting with the ImageNet dataset and report result in Table 19. In this experiment, we use the robust pretrained ResNet18 model (model ID: Salman2020Do_R18) as the victim model. We use the standard attack setting, i.e., targeted attack with $\epsilon = 4/255, \eta = 1/255$ with 20 steps. It can be seen that both MOO and TA-MOO could obtain a better attack performance than the uniform strategy. It is a worth noting that, in the experiment on the CIFAR10/CIFAR100 datasets (i.e., Table 6 in the main paper) the dominating issue of the vertical filliping exists and prevents MOO to obtain a better performance. In the ImageNet dataset, the dominating issue is less serious, therefore, explains the improvement of MOO and corroborates our hypothesis on the issue of dominating task.

Table 19: Evaluation on the EoT setting with the ImageNet dataset. The most important metric is emphasized in blue.

| | A-All | A-Avg | I | H | V | C | G | B | R |
|---|---|---|---|---|---|---|---|---|---|
| Uniform | 31.52 | 46.59 | 41.12 | **40.98** | 67.42 | 41.60 | 43.26 | 41.82 | 49.96 |
| MOO | 31.92 | 47.19 | 41.92 | 41.78 | 67.64 | **42.10** | 43.66 | 42.74 | 50.48 |
| TA-MOO | **32.00** | **47.21** | **41.94** | 41.80 | **67.66** | 42.06 | **43.70** | **42.80** | **50.52** |

Table 20: Evaluation of Attacking Ensemble model on the CIFAR10 (C10) and CIFAR100 (C100) datasets. The highest/second highest performance is highlighted in **Bold**/Underline. The table is copied from Table 1 in the main paper for reading comprehension purpose.

| | | CW | | CE | | KL | |
|---|---|---|---|---|---|---|---|
| | | A-All | A-Avg | A-All | A-Avg | A-All | A-Avg |
| C10 | Uniform | 26.37 | **41.13** | 28.21 | 48.34 | 17.44 | 32.85 |
| | MinMax | 27.53 | 41.20 | 35.75 | **51.56** | 19.97 | **33.13** |
| | MOO | 18.87 | 34.24 | 25.16 | 44.76 | 15.69 | 29.54 |
| | TA-MOO | **30.65** | 40.41 | **38.01** | 51.10 | **20.56** | 31.42 |
| C100 | Uniform | 52.82 | **67.39** | 55.86 | 72.62 | 38.57 | **54.88** |
| | MinMax | 54.96 | 66.92 | 63.70 | 75.44 | 40.67 | 53.83 |
| | MOO | 51.16 | 65.87 | 58.17 | 73.19 | 39.18 | 53.44 |
| | TA-MOO | **55.73** | 67.02 | **64.89** | **75.85** | **41.97** | 53.76 |

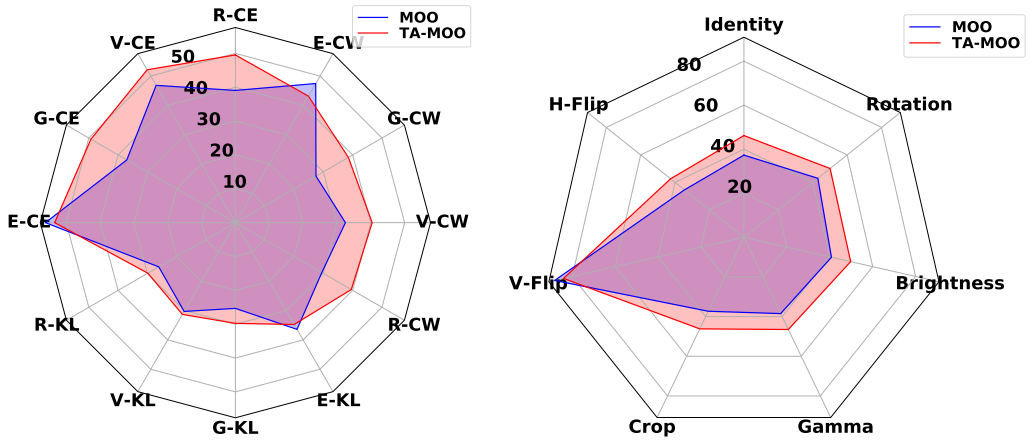
# E Additional Discussions

## E.1 When does MOO Work?

**The dominating issue.** On one hand, there is the dominating issue that happens in all the three settings. The issue can be recognized by the gap of attack performance among tasks. For example, in Table 21 (i.e., the ENS setting with the diverse ensemble and MOO method), the gap between highest ASR (at EfficientNet) and lowest ASR (at GoogLeNet) is 19%. In the EoT setting, the problem is even worse: The largest gap observed is 53.6% as shown in Table 14 (the highest ASR is 88.19% with Vertical flipping and the lowest ASR is 34.54% with Horizontal flipping in with MOO - D-C10 setting). The dominating issue is also be recognized by the observation that a significant small gradient strength of one task on comparison with other tasks' strength. For example, in Table 21 it can be seen that the gradient strength corresponding to the EfficientNet architecture (mean value

Table 21: Attacking Ensemble model with a diverse set D={R-ResNet18, V-VGG16, G-GoogLeNet, E-EfficientNet} and non-diverse set ND={4 ResNets}. $w$ represents the final $w$ of MOO (mean $\pm$ std). $\|\nabla_\delta f_i(\delta)\|$ represents the gradient norm of each model (mean $\pm$ std). The table is copied from Table 2 in the main paper for reading comprehension purpose.

| | | A-All | A-Avg | R/R1 | V/R2 | G/R3 | E/R4 |
|---|---|---|---|---|---|---|---|
| D | $\|\nabla_\delta f_i(\delta)\|$ | - | - | $7.15 \pm 6.87$ | $4.29 \pm 4.64$ | $7.35 \pm 7.21$ | $0.98 \pm 0.72$ |
| | $w$ | - | - | $0.15 \pm 0.14$ | $0.17 \pm 0.13$ | $0.15 \pm 0.14$ | $0.53 \pm 0.29$ |
| | Uniform | 28.21 | 48.34 | 48.89 | 49.08 | 48.38 | 47.03 |
| | MOO | 25.16 | 44.76 | 39.06 | 46.83 | 37.05 | 56.11 |
| | TA-MOO | 38.01 | 51.10 | 49.55 | 52.15 | 49.29 | 53.40 |
| ND | $\|\nabla_\delta f_i(\delta)\|$ | - | - | $8.41 \pm 8.22$ | $6.68 \pm 6.95$ | $7.36 \pm 6.03$ | $5.67 \pm 6.09$ |
| | $w$ | - | - | $0.23 \pm 0.21$ | $0.24 \pm 0.17$ | $0.23 \pm 0.19$ | $0.30 \pm 0.21$ |
| | Uniform | 28.17 | 48.75 | 51.94 | 45.55 | 54.15 | 43.34 |
| | MOO | 32.50 | 52.21 | 53.25 | 49.05 | 56.80 | 49.76 |
| | TA-MOO | 41.01 | 57.33 | 58.88 | 55.32 | 60.81 | 54.29 |




(a) ENS setting (b) EoT setting

Figure 4: Comparison on the ASR of each individual task. R: ResNet18, V: VGG16, G: GoogLeNet, E: EfficientNet. CE: Cross-entropy loss, KL: Kullback-Leibler divergence, CW: Carnili-Wagner loss

is 0.98) is much lower than those of other architectures (mean values are at least 4.29). As the result, the weight corresponding to the EfficientNet architecture is much higher than those of others.

The root of the dominating issue can be the natural of the setting (i.e., as shown in Table 14 with the EoT setting, when the domination of the Vertical flipping task can be observed in all methods) or because of the MOO solver which is discussed in Section E.3

**Overcoming the dominating issue.** On the other hand, if overcoming this issue, MOO can outperform the Uniform strategy. For example, on attacking the non-diverse ensemble model (i.e., 4 ResNets) MOO surpasses the Uniform strategy by 4.3% and 3.5% in the ASR-All and ASR-Avg metrics, respectively. On generating universal perturbations, MOO outperforms the Uniform strategy in most of the settings. As discussed in Section D.4, a simple memory caching trick can helps to overcome the infinite gradient issue and significantly boosts the performance of MOO or TA-MOO. Therefore, we believe that developing a technique to lessen the dominating issue might be a potential extension to further improve the performance.

**Balancing among goal-unachived tasks.** We observed in the EoT setting, the dominating issue is strictly serious when gradients of some tasks are much weaker/stronger than others. It is because of the natural of the transformation operations, therefore, this issue happens regardless status of the tasks. In the set of goal-unachieved tasks' gradients can exist a dominated one, resulting to a much higher weight of the dominated task. Therefore, in order to strike a more balance among goal-unachieved tasks, we apply an additional regularization which minimizes the entropy of goal-unachieved weights $\mathcal{H}(w) = \sum_{i=s+1}^{m} -w_i \log w_i$. If all tasks have been achieved (i.e., $s = m$) then the additional regularization will be ignored. This additional regularization helps to improve further 2% in the EoT setting.

### E.2 Importance of the Task-Oriented Regularization.

In this discussion, we would like to provide more experimental results in the ENS and EoT settings to further emphasize the contribution of the Task-Oriented regularization. Figure 4 shows the ASR of each individual task in the ENS setting with three losses and the EoT setting with ResNet18 architecture and deterministic transformations. As shown in Figure 4a, in the ENS setting, the MOO adversary produces a much higher ASR on the EfficientNet architecture than other architectures with any losses. In contrast, the TA-MOO adversary has a lower ASR on the EfficientNet architecture but a much higher ASR on other architectures. Similar observation can be seen in Figure 4b such that the ASR corresponding to the V-flipping of MOO is slightly higher than that of TA-MOO, however, the ASR on other transformations of MOO is much lower than those of TA-MOO.

### E.3 More Efficient MOO Solvers

**Discussions on the weighted-sum method.** One of the most common approaches to solve the MOO problem is the scalarizing method, which formulates a single-objective optimization (SOO) such that the optimal solutions to the SOO problem are Pareto optimal solutions to the MOO problem. While this line of approach (e.g., weighted-sum method) is suitable for end-to-end learning such as deep learning, there are several acknowledged weaknesses: (i) the choice of utility function has a large impact on the computational complexity of the resulted SOO problem (Bjornson et al., 2014; Björnson & Jorswieck, 2013); (ii) a small change in weights may results in big changes in the combined objective (Caballero et al., 1997), and vice versa, a huge different weights may produce nearly similar result (Coello Coello, 1999); (iii) it does not work well in the case of a non-convex objective space (Deb, 2011).

One of the most common replacement for the weighted-sum method is the $\epsilon$ constraint method which is applicable to either convex or non-convex problem. Applying a more efficient MOO solver might be one of the potential extensions of this work.

**Discussions on the gradient descent solver.** Inspired by Sener & Koltun (2018), in this paper we use multi-gradient descent algorithm (Deb, 2011) as an MOO solver which casts the multi-objective problem to a single-objective problem. While Sener & Koltun (2018) used Frank-Wolfe algorithm to project the weight into the desired simplex, we use parameterization with softmax instead. Although this technique is much faster than Frank-Wolfe algorithm, it has some weaknesses that will be addressed in our future work. More specifically, the GD solver with softmax parameterization cannot handle well the edge case which is the root of the dominating issue. The snippet code E.3 provides a minimal example of quadratic optimization problem as similar in MGDA, where the goal is to find $w^* = \underset{w \in \Delta_w}{\text{argmin}} \sum_{i=1}^{5} \|w_i \mathbf{g}_i\|_2^2$. The solver is the Gradient Solver with softmax parameterization. With $\text{input}_1$ where none of elements dominates others, the solver works quite reasonable with the weights corresponding to 4 first elements are equal and less than the last one (corresponding to bigger strength). With $\text{input}_2$ where $g_5 \gg g_1$, the solver still works well where $w_1 = 1$ corresponding to the minimal strength $g_1 = 0.1$. However, with $\text{input}_3$, the solver fails to find a good solution (which should be $w = [1, 0, 0, 0]$ given that input). It is a worth noting that the main goal of this paper is to show the application of Multi-objective Optimization for generating adversarial examples and the impact of the Task-Oriented regularization. Therefore, while the issue of the gradient descent solver is well recognized, we did not take effort to try with a better solver.

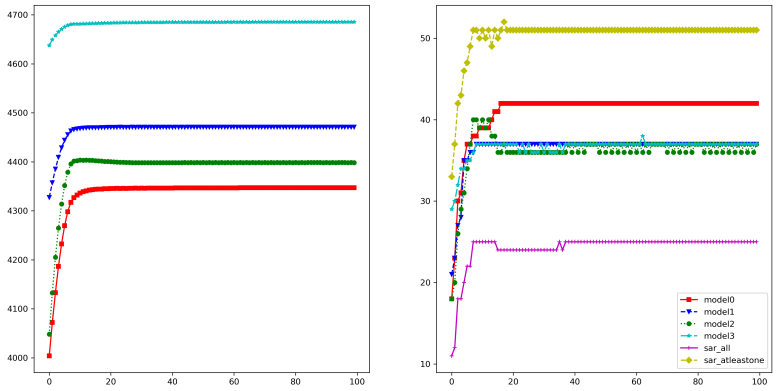
```python
import torch
import torch.nn.functional as F
import torch.optim as optim

input_1 = [0.1, 0.1, 0.1, 0.1, 0.2] # normal case
input_2 = [0.01, 0.1, 0.1, 0.1, 2e3] # normal case
input_3 = [0.001, 0.002, 0.002, 0.002, 2e3] # dominating issue

init_alpha = [0.2, 0.2, 0.2, 0.2, 0.2]
g = torch.tensor(input_3)
alpha = torch.tensor(init_alpha, requires_grad=True)
opt = optim.SGD([alpha], lr=1.0)

for step in range(20):
    w = F.softmax(alpha, dim=0)
    loss = torch.square(torch.sum(w * g))
    opt.zero_grad()
    loss.backward()
    opt.step()
    print('step={}, w={}'.format(step, w.detach().numpy()))

# Result with input_1
# step=19, w=[0.20344244 0.20344244 0.20344244 0.20344244 0.18623024]
# Result with input_2
# step=19, w=[9.999982e-01 5.582609e-07 5.582609e-07 5.582609e-07 0.]
# Result with input_3
```

```
# step=19, w=[0.28042343 0.23985887 0.23985887 0.23985887 0.]
```

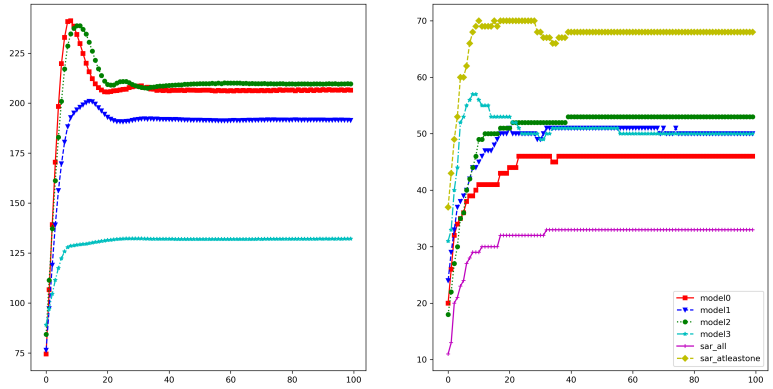
Listing 1: Python example of the Gradient Solver with softmax parameterization

### E.4 Correlation between the Objective Loss and Attack Performance.

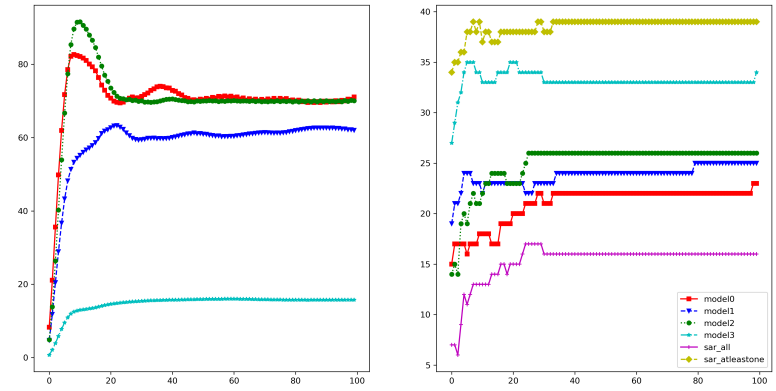
It is broadly accepted that to fool a model, a feasible approach is maximizing the objective loss (i.e., CE, KL, or CW loss), and the higher the loss, the higher the attack success rate. While it is true with the same architecture, we found that it does not hold when comparing different architectures. Figure 5 shows the adversarial loss and the attack success rate for each model in the ENS setting. With the CW loss as the adversarial objective, it can be observed that there is a positive correlation between the loss value and the ASR, i.e., the higher the loss, the higher the ASR. For example, with the same adversarial examples, the adversarial loss on EfficientNet is the highest and so is ASR. However, there is no clear correlation observed when using CE and KL losses. Therefore, the higher weighted loss does not directly imply a higher success rate for attacking an ensemble of different architectures. The MinMax method (Wang et al., 2021) which solely weighs the tasks' losses, therefore, does not always achieve a good performance in all the tasks.




(a) CW




(b) CE




(c) KL

Figure 5: Loss (left fig) and ASR (right fig) of each task over all attack iterations with the MinMax method. model0/1/2/3 represents R/V/G/E architecture, respectively.