Ensemble-in-One: Ensemble Learning within Random Gated Networks for Enhanced Adversarial Robustness

Yi Cai, Xuefei Ning*, Huazhong Yang, Yu Wang*

Department of Electronic Engineering, Tsinghua University
yicai.cy@gmail.com, foxdoraame@gmail.com, yanghz@tsinghua.edu.cn,
yu-wang@tsinghua.edu.cn

Abstract

Adversarial attacks have threatened modern deep learning systems by crafting adversarial examples with small perturbations to fool the convolutional neural networks (CNNs). To alleviate that, ensemble training methods are proposed to facilitate better adversarial robustness by diversifying the vulnerabilities among the sub-models, simultaneously maintaining comparable natural accuracy as standard training. Previous practices also demonstrate that enlarging the ensemble can improve the robustness. However, conventional ensemble methods are with poor scalability, owing to the rapidly increasing complexity when containing more sub-models in the ensemble. Moreover, it is usually infeasible to train or deploy an ensemble with substantial sub-models, owing to the tight hardware resource budget and latency requirement. In this work, we propose *Ensemble-in-One* (EIO), a simple but effective method to efficiently enlarge the ensemble with a random gated network (RGN). EIO augments a candidate model by replacing the parametrized layers with multi-path random gated blocks (RGBs) to construct an RGN. The scalability is significantly boosted because the number of paths exponentially increases with the RGN depth. Then by learning from the vulnerabilities of numerous other paths within the RGN, every path obtains better adversarial robustness. Our experiments demonstrate that EIO consistently outperforms previous ensemble training methods with smaller computational overheads, simultaneously achieving better accuracy-robustness trade-offs than adversarial training methods under black-box transfer attacks. Code is available at https://github.com/cai-y13/Ensemble-in-One.git

Introduction

With the convolutional neural networks (CNNs) becoming ubiquitous, the security and robustness of neural network systems are attracting increasing focuses. Recent studies find that CNN models are inherently vulnerable to adversarial attacks (Goodfellow, Shlens, and Szegedy 2014), which add imperceptible perturbations or patches on the natural images, referred to as adversarial examples, to mislead the neural network models. Even without accessing the target model, an adversary can still generate adversarial examples on other surrogate models to attack the target model by exploiting the adversarial transferability among them.

*Corresponding authors.

Such vulnerability of CNN models has spurred extensive researches on improving the robustness against adversarial attacks. One stream of approaches target on learning robust features for an *individual* model (Madry et al. 2017; Brendel et al. 2020). Informally, robust features are defined as the features that are less sensitive to the adversarial perturbations on the inputs. A representative approach, referred to as adversarial training (Madry et al. 2017), on-line generates adversarial examples on which the model minimizes the training loss. As a result, adversarial training encourages the model to capture the robust features that are less sensitive to the adversarial input perturbations, thereby alleviating the model's vulnerability. However, such adversarial training methods often have to sacrifice the clean accuracy for enhanced robustness (Zhang et al. 2019) even with a bag of tricks (Pang et al. 2020), since they exclude the non-robust features and become less discriminative for the examples with high similarity in the feature space.

Besides pursuing improved robustness for an individual model, another active stream of researches focus on forming strong *ensemble* models to improve the robustness (Yang et al. 2020; Bagnall, Bunescu, and Stewart 2017; Pang et al. 2019; Kariyappa and Qureshi 2019; Liu et al. 2019). Generally speaking, an ensemble model is constructed by aggregating multiple sub-models. Intuitively, an ensemble model is more likely to exhibit better robustness than an individual model, because a successful attack needs to mislead the majority of the sub-models rather than just one model. While the robustness of an ensemble highly relies on the diversity of its sub-models, recent study finds that CNN models trained independently have highly-overlapped adversarial subspaces (Tramèr et al. 2017). Therefore, many studies propose ensemble training methods to diversify the sub-models. For example, DVERGE (Yang et al. 2020) proposes to distill non-robust features corresponding to each sub-model's vulnerability, then isolates the vulnerabilities of sub-models by mutual learning so as to impede the adversarial transferability among them.

There is another recognized insight that the ensembles composed by more sub-models tend to have greater robustness improvement. Table 1 shows the robustness trend of the ensembles trained with various ensemble training methods. It drives us to further explore whether the trend will continue when keeping increasing the sub-models until infinity.

$\#$sub-model	Baseline	ADP	GAL	DVERGE
3	0.0%/1.5%	0.0%/9.6%	39.7%/11.4%	53.2%/40.0%
5	0.0%/2.1%	0.0%/11.8%	32.4%/31.7%	57.2%/48.9%
8	0.0%/3.2%	0.0%/12.0%	22.4%/37.0%	63.6%/57.9%

Table 1: Accuracy of the ensembles trained by different methods under adversarial attack, with 3, 5, and 8 sub-models respectively (Yang et al. 2020). The number before and after the slash represent black-box and white-box adversarial accuracy under perturbation strength 0.03 (8/255) and 0.01 (2.5/255), respectively.

However, the conventional ensemble construction methods are with poor scalability because of the rapidly increasing overhead. Especially when using mutual learning that trains the sub-models in a round-robin manner, the training cost increases at a speed of $\mathcal{O}(n^2)$ where n is the number of sub-models. And the computational cost of inference also increases to $n\times$ of an individual model.

To address the scalability issue, we propose *Ensemble-in-One* (EIO), a novel ensemble construction and training paradigm, to simultaneously obtain enhanced adversarial robustness, natural (clean) accuracy and higher computational efficiency. For a dedicated CNN model, we conduct a *Random Gated Network* (RGN) by substituting each parameterized layer with a *Random Gated Block* (RGB). In this way, the network can instantiate numerous sub-models by controlling the gates in the blocks. Ensemble-in-One substantially reduces the complexity when scaling up the ensemble. In summary, the contributions are summarized as below:

- Ensemble-in-One is a simple but effective method that learns robust ensembles within one over-parametrized random gated network. The **scalable** construction of EIO enables us to build substantial larger training-time ensembles and employ diversification learning algorithms to **win both higher clean accuracy and better adversarial robustness**.
- Ensemble-in-One is a **plug-in** method that can be easily applied on any CNN model (as the basic network).
- Ensemble-in-One **incurs no extra inference-time cost**, as opposed to conventional ensemble methods. Specifically, we only sample one path from the RGN, whose inference cost is exactly the same as the basic network.
- Extensive experiments demonstrate the effectiveness of Ensemble-in-One on a wide range of models and datasets. EIO consistently outperforms the previous ensemble training methods with a much smaller computational overhead. Moreover, EIO can achieve better accuracy-robustness trade-offs than adversarial training methods, especially under black-box attack scenarios, which is more prevalent in the real world.

Related Work

Adversarial Attacks

The inherent vulnerability of CNN models raises high risk on the security of deep learning systems. An adversary can apply additive perturbations on a natural instance to generate an adversarial example to induce wrong predictions of CNN models (Goodfellow, Shlens, and Szegedy 2014). Denoting an input as x, the goal of adversarial attacks is to find a perturbation δ such that $x_{adv} = x + \delta$ can mislead the model, where $||\delta||_p$ satisfies the intensity constraint $||\delta||_p \leq \epsilon$ to be human-invisible. To formulate that, the adversarial attack aims at maximizing the loss $\mathcal{L}$ for the model with parameters Θ on the input-label pair (x, y), i.e. $\delta = \mathrm{argmax}_\delta \mathcal{L}_\Theta(x + \delta, y)$, under the constraint that the ℓ_p norm of the perturbation should not exceed the bound ϵ. Usually, we use ℓ_∞ norm (Goodfellow, Shlens, and Szegedy 2014; Madry et al. 2017) of the perturbations to measure the attack's effectiveness or model's robustness. An attack that requires smaller perturbations to successfully deceive the model is regarded to be stronger. Correspondingly, a defense that enforces the attacks to enlarge perturbation intensity is regarded to be more effective.

A variety of adversarial attack methods have been investigated to strengthen the attack effectiveness. For example, the fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2014) exploits the gradient descent to generate the adversarial perturbations. As an improvement, many studies further show the attack can be strengthened through multi-step projected gradient descent (PGD) (Madry et al. 2017) generation, random-starting strategy, and momentum mechanism (Dong et al. 2017). Recently, SGM (Wu et al. 2020) further finds that adding weight to the gradients going through the skip connections can make the attacks more effective. Other prevalent attack approaches include C&W losses (Carlini and Wagner 2017b) , M-DI^2-FGSM (Xie et al. 2019), AutoAttack (Croce and Hein 2020), etc. These attacks provide strong and effective ways to generate adversarial examples, which greatly threatens the AI systems.

Adversarial Training

To improve the robustness of CNN models, there are also extensive countermeasures for adversarial attacks. One active research direction targets improving the robustness of individual models. Adversarial training (Madry et al. 2017) optimizes the model parameters Θ on the adversarial examples generated in every step of the training stage, that is, $\Theta = \mathrm{argmin}_\Theta \mathcal{L}_\Theta(x + \delta, y)$. Therefore, the optimized model will tend to drop non-robust features to converge better on the adversarial data. However, adversarial training encourages the model to fit better on the adversarial examples, thereby reducing the generalization on the clean data and causing significant degradation of the clean accuracy. The studies in (Zhang et al. 2019; Gowal et al. 2020; Wu et al. 2021) theoretically characterize the trade-offs between accuracy and robustness and explore the behind working mechanisms. Bag of tricks (Pang et al. 2020) have been developed to obtain higher accuracy, e.g. early stopping, warmup, learning rate scheduling, etc., while the accuracy loss is still non-negligible.

Test-time Randomness for Adversarial Defense

There also exist studies that introduce test-time randomness to improve the robustness. Feinman et. al. (Feinman et al. 2017) utilize the uncertainty measure in dropout networks to detect adversarial examples. Dhillon et. al. (Dhillon et al. 2018) and Xie et. al. (Xie et al. 2017) incorporate layer-wise weighted dropout and random input transformations during test time to improve the robustness. Test-time randomness is found to be effective in increasing the required distortion on the model, since test-time randomness makes generating white-box adversarial examples almost as difficult as generating transferable black-box ones (Carlini and Wagner 2017a). Nevertheless, test-time randomness increases the inference cost and can be circumvented to some extent with the expectation-over-transformation technique (Athalye, Carlini, and Wagner 2018). Instead of relying on test-time randomness, Ensemble-in-One exploits and only exploits randomness in the training time to ensure the stability and low cost of test-time predictions.

Ensemble Training for Adversarial Defense

Besides improving the robustness of individual models, another recent research direction is to investigate the robustness of model ensembles in which multiple sub-models work together. The basic idea is that multiple sub-models can provide diverse decisions. Ensemble methods can combine multiple weak models to jointly make decisions, thereby assembling as a stronger entirety. However, it is demonstrated that independent training of multiple models tends to capture similar features, which would not provide diversities among them (Kariyappa and Qureshi 2019).

Several studies propose ensemble training methods to diversify the sub-models to improve the ensemble robustness. For example, Pang et. al. treat the distribution of output predictions as a diversity criterion and they propose an adaptive diversity promoting (ADP) regularizer (Pang et al. 2019) to diversify the non-max predictions of sub-models. Sanjay et. al. regard the gradients w.r.t. the inputs as a discrimination of different models, thus they propose a gradient alignment loss (GAL) (Kariyappa and Qureshi 2019) which takes the cosine similarity of the gradients as a criterion to train the sub-models. The very recent work DVERGE (Yang et al. 2020) claims that the similar non-robust features captured by the sub-models cause high adversarial transferability among them. Therefore, the authors exploit non-robust feature distillation and adopt mutual learning to diversify and isolate the vulnerabilities among the sub-models, such that the within-ensemble transferability is highly impeded. However, as mentioned before, such ensemble methods are overwhelmed by the fast-increasing overhead when scaling up the ensemble. For example, DVERGE takes 11 hours to train an ensemble with three sub-models while needs approximately 50 hours when increasing to eight. Therefore, a more efficient ensemble construction method is demanded to tackle the scaling problem.

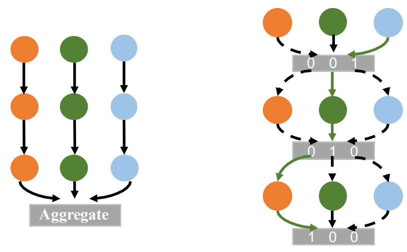


Figure 1: Normal ensemble training of multiple sub-models (Left) and the proposed ensemble-in-one training within a random gated network (Right). By selecting the paths along augmented layers, the ensemble-in-one network can instantiate n^L sub-models, where n represents the augmentation factor of the multi-gated block for each augmented layer and L represents the number of augmented layers in the network.

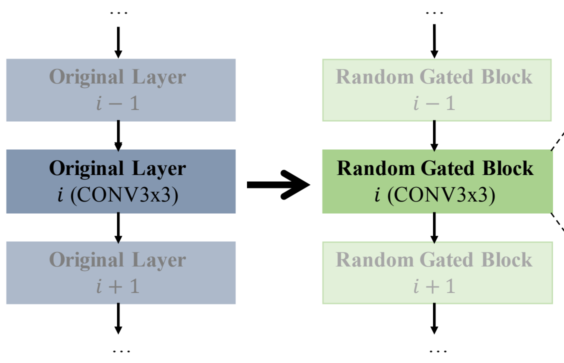
Ensemble-in-One

Basic Motivation

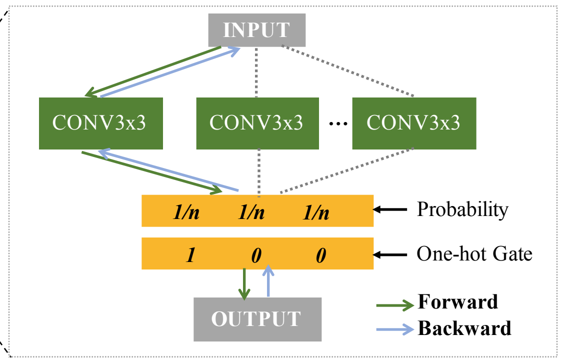
The conventional way to construct ensembles is to simply aggregate multiple sub-models by averaging their predictions, which is inefficient and hard to scale up. An intuitive way to improve the scalability of the ensemble construction is to introduce an ensemble for each layer in the network. As shown in Fig.1, we construct a random-gated network by augmenting each parametrized layer with an n-path gated block. Then by selecting the paths through the augmented layers, the dynamic network can instantiate n^L varied sub-models ideally. Taking ResNet-20 as an example, by replacing each convolution layer (ignoring the skip connection branch) with a two-path gated module, the overall path count could approach $2^{19} = 524,288$. This type of architecture augmentation provides an approximation to training a very large ensemble of sub-models. Then through vulnerability-diversification mutual learning, each path will obtain better robustness. Following this idea, we propose *Ensemble-in-One* to improve the robustness of both individual models and ensembles.

Construction of the Random Gated Network

We denote a candidate neural network as $\mathcal{N}(o_1, o_2, ..., o_m)$, where o_i represents an operator in the network. To transform the original network into a random gated network (RGN), we first extract the neural architecture to obtain the connection topology and layer types. Then, we replace each parameterized layer (mainly convolutional layer, optionally followed by a batch normalization layer) with a random gated block (RGB). As shown in Fig. 2, each RGB simply construct n copies of the original layer, and leverages one-hot gates with uniform probabilities to control the open or mute of corresponding sub-layers. Note that these copied sub-layers are with different weights. We denote the RGN as $\mathcal{N}(d_1, d_2, ..., d_m)$, where $d_i = (o_{i1}, ..., o_{in})$. Let g_i be the



(a) Construction of random gated network



(b) The schematic diagram of random gated block

Figure 2: The construction of random gated network based on random gated blocks. The forward propagation will select one path to allow the input pass. Correspondingly, the gradients also propagate backward along the same path.

Algorithm 1: Training process for Ensemble-in-One

```
Require: Path samples per iteration $p$
Require: Random Gated Network $\mathcal{N}$ with $L$ parameterized layers
Require: Pre-training epoch $E_w$, training epoch $E$, and data batch $B_d$
Require: Optimization loss $\mathcal{L}$, learning rate $lr$
Ensure: Trained Ensemble-in-One model
1:  # pre-training of $\mathcal{N}$
2:  for e = 1, 2, ..., $E_w$ do
3:      for b = 1, 2, ..., $B_d$ do
4:          Random Sample Path $\mathcal{P}_i$ from $\mathcal{N}$
5:          Train $\mathcal{P}_i$ in batched data
6:      end for
7:  end for
8:  # learning vulnerability diversity for $\mathcal{N}$
9:  for e = 1, 2, ..., $E$) do
10:     for b = 1, 2, ..., $B_d$) do
11:         Random sample $l \in [1, L]$
12:         # randomly sample $p$ paths
13:         $\mathcal{S}$=[$\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_p$], s.t. $\forall i, j, \exists k \in [1, l]$, s.t. $\mathcal{P}_i[k] \neq \mathcal{P}_j[k]$
14:         Get data $(X_t, Y_t), (X_s, Y_s) \leftarrow D$
15:         # Get distilled data
16:         for i = 1, 2, ..., $p$ do
17:             $X_i' = x'_{\mathcal{P}_i^l}(X_t, X_s)$
18:         end for
19:         $\nabla_{\mathcal{N}} \leftarrow 0$
20:         for i = 1, 2, ..., $p$ do
21:             $\nabla_{\mathcal{P}_i} = \nabla(\sum_{j \neq i} \mathcal{L}_{f_{\mathcal{P}_i}}(f_{\mathcal{P}_i}(X_j'), Y_s))$
22:             $\nabla_{\mathcal{N}} = \nabla_{\mathcal{N}} + \nabla_{\mathcal{P}_i}$
23:         end for
24:         $\mathcal{N} = \mathcal{N} - lr * \nabla_{\mathcal{N}}$
25:     end for
26: end for
```

gate in the i_{th} RGB, then a specific path derived from the RGN can be expressed as $\mathcal{P} = (g_1 \cdot d_1, g_2 \cdot d_2, ..., g_m \cdot d_m)$.

For each RGB, when performing the computation, only one of the n gates is opened at a time, and the others will be temporarily muted. Thus, only one path of activation is active in memory during both forward and backward passes, keeping the memory consumption of RGN training to the same level of training an individual model. Moreover, to ensure that all paths can be equally sampled and trained, each gate in a RGB is chosen with identical probability, i.e. $1/n$ if each RGB consists of n sub-operators. Therefore, the one-hot gate function can be expressed as:

$$g_i = \begin{cases} [1, 0, ..., 0] & \text{with probability } 1/n, \\ [0, 1, ..., 0] & \text{with probability } 1/n, \\ \quad ... & \\ [0, 0, ..., 1] & \text{with probability } 1/n. \end{cases} \tag{1}$$

An RGN is similar to the super network in parameter-sharing neural architecture search, and the forward process of an RGN is similar to evaluating a sub-architecture (Pham et al. 2018; Cai, Zhu, and Han 2018; Ning et al. 2021). Compared to conventional ensemble training methods, our method is easier to scale up the ensemble. It has the same memory requirement for activation as an individual model.

Learning Ensemble in One

The goal of learning ensemble-in-one is to encourage the diversity of vulnerabilities between paths within the RGN by mutually learning from each other. Let $\mathcal{P}_i$ and $\mathcal{P}_j$ be two different paths, where we define two paths as different when at least one of their gates is different. To diversify the vulnerabilities of paths, we train each path by distilling the adversarial features of other paths. We adopt the same adversarial feature distillation strategy as previous work (Ilyas et al. 2019; Yang et al. 2020). Specifically, consider two input-label pairs (x_t, y_t) and (x_s, y_s) independently sampled from the training set, we construct an adversarial sample, $x'_{\mathcal{P}_i^l}$, by adding perturbations on x_s, so that its feature in the l_{th} layer

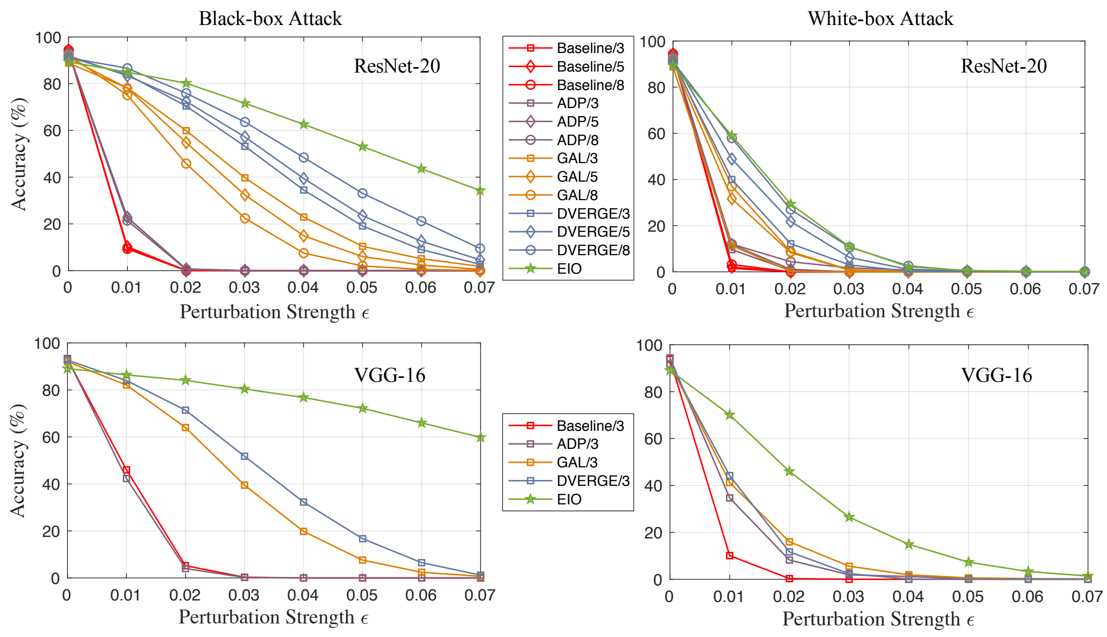


Figure 3: Robustness comparison of EIO with previous ensemble training methods. Left: adversarial accuracy under black-box transfer attack; Right: adversarial accuracy under white-box attack. The number after the slash (/3, /5, /8) stands for the number of sub-models within the ensemble. The evaluations include ResNet-20 and VGG-16 on CIFAR-10. The distillation perturbation strength of VGG-16-based EIO is set as $\epsilon_d = 0.03$.

in the path $\mathcal{P}_i$ is similar to that of x_t:

$$x'_{\mathcal{P}_i^l}(x_t, x_s) = \text{argmin}_z ||f^l_{\mathcal{P}_i}(z) - f^l_{\mathcal{P}_i}(x_t)||^2, \tag{2}$$

where $||z - x_s||_\infty \le \epsilon_d$. This adversarial sample exposes the vulnerability of path $\mathcal{P}_i$ on classifying x_s. Therefore, training another different path $\mathcal{P}_j$ using $x'_{\mathcal{P}_i^l}(x_t, x_s)$ encourages $\mathcal{P}_j$ to circumvent the vulnerability of $\mathcal{P}_i$. The optimization objective to minimize for path $\mathcal{P}_j$ is:

$$\mathbb{E}_{(x_t,y_t),(x_s,y_s),l}\mathcal{L}_{f_{\mathcal{P}_j}}(x'_{\mathcal{P}_i^l}(x_t, x_s), y_s). \tag{3}$$

We expect each path to learning from the vulnerabilities of a substantial number of other paths. Thus, the overall objective to minimize when training the ensemble-in-one RGN can be written as:

$$\sum_{\mathcal{P}_j \sim \mathcal{N}} \mathbb{E}_{(x_t,y_t),(x_s,y_s),l} \sum_{\mathcal{P}_i \sim \mathcal{N}, i \ne j} \mathcal{L}_{f_{\mathcal{P}_j}}(x'_{\mathcal{P}_i^l}(x_t, x_s), y_s), \tag{4}$$

where $\mathcal{N}$ is the set of all paths in the RGN. Since it is obviously impossible to involve all the paths in a training iteration, we randomly sample a certain number of paths by stochastically sample the one-hot gates according to Eq.1. We denote the number of paths sampled in each iteration as p, and refer to the set of selected paths as $\mathcal{S}$ (a subset of $\mathcal{N}$).

Algorithm 1 summarizes the training process of the RGN. Before starting vulnerability diversification training, we pretrain the RGN with standard training settings to quickly boost up the training process. During the training process, a random path is picked out in each iteration and trained on a batch of clean data. Then for each batched data-pairs (X_t, Y_t, X_s, Y_s), the process of vulnerability diversification contains three basic steps as follows. First, we randomly sample p different paths to be involved in the iteration. Second, we construct the adversarial samples for distillation-based training for the data-pair (x_t, y_t, x_s, y_s) and the sampled paths according to Eq. 2. The argmin optimization in Eq. 2 is conducted by applying a 10-step PGD. Third, we mutually train each path by distillation using the adversarial samples generated on other paths in a round-robin manner. As the paths share a proportion of weights owing to the weight sharing in the RGN, instead of instantaneously updating the weights when training each path, we sum up the gradients calculated using all paths in $\mathcal{S}$ and conduct a final update using the aggregated gradients.

Model Derivation and Deployment

Once the training of RGN is finished, we sample an individual path from the RGN and removing other modules for inference deployment. We randomly set the gates of all RGBs and extract the sub-model by tracing its execution path. Therefore, the architecture of the derived model for inference is exactly the same as the basic network before supernet augmentation. In this way, EIO has a smaller inference cost than conventional robust ensembles (Yang et al. 2020) as it doesn't introduce any additional overhead to the inference process. On the other hand, compared to previous studies that apply dropout-like strategy during testing (Feinman et al. 2017; Dhillon et al. 2018; Xie et al. 2017), EIO introduces only training-time randomness in ensemble training, while ensuring the test-time stability of the predictions by removing any test-time randomness.

Experiments and Analysis

Experiment Settings

Benchmark. We conduct experiments with the ResNet-20 (He et al. 2016), VGG-16 (Simonyan and Zisserman 2014), WideResNet-34-10 (Zagoruyko and Komodakis 2016), and ResNet-18 (He et al. 2016) networks on the CIFAR (Krizhevsky 2009) and Tiny-ImageNet (Le and Yang 2015) datasets. Specifically, we construct the RGNs based on these networks by substituting each convolution layer to a n-path RGB (by default, $n = 2$). Overall, there are 19 RGBs for ResNet-20, 13 RGBs for VGG-16, 32 RGBs for WideResNet-34-10, and 17 RGBs for ResNet-18. We compare with multiple competitive baseline methods, including the *Baseline* that trains the models in a standard way and three previous ensemble training methods: *ADP* (Pang et al. 2019), *GAL* (Kariyappa and Qureshi 2019), and *DVERGE* (Yang et al. 2020). In addition, we also include the adversarial training (*AdvT*) method (Madry et al. 2017), the improved adversarial training method *TRADES* (Zhang et al. 2019), TRADES with *a bag of tricks* (Pang et al. 2020), and an enhanced AdvT (Gowal et al. 2020) into the comparison.

Training Details. We follow the implementation of DVERGE (Yang et al. 2020)[1] to choose the training configurations of ADP, GAL, and DVERGE. The AdvT applies PGD-10 with different perturbation strengths. For TRADES, TRADES with a bag of tricks and AdvT (Gowal), we directly use their released codes and checkpoints[2,3,4]. We train the Ensemble-in-One networks for 200 epochs using SGD with momentum 0.9 and weight decay 0.0001. The initial learning rate is 0.1, and decayed by 10x at the 100-th and the 150-th epochs respectively. When deriving the individual models, we fine-tune the derived models for 0-20 epochs (by default 5) using SGD with a learning rate of 0.001. Note that the fine-tuning process is optional and one can adjust the number of epochs according to the need. By default, for an RGN training, we sample 3 paths to construct temporary sub-ensemble per iteration ($p = 3$). The augmentation factor n for each RGB is set to 2, and the PGD-based perturbation strengths ϵ_d is set to 0.07 for feature distillation with 10 iterative steps and each step size of $\epsilon_d/10$.

Threat Models. We focus more on black-box transfer attacks which is more prevalent in the real world, as the white-box attack assumes the adversary has full knowledge of the model parameters. For black-box transfer attacks on CIFAR-10, the involved attack methods include: (1) PGD with momentum and three random starts (Madry et al. 2017); (2) M-DI^2-FGSM (Xie et al. 2019); and (3) SGM (Wu et al. 2020). The attacks are with different perturbation strength and the iterative steps are set to 100 with the step size of ϵ/5. Besides the cross-entropy loss, we also incorporate the C&W loss with these attacks. Therefore, there will be 3 (surrogate models) $\times$ 5 (attack methods, PGD with three random starts, M-DI^2-FGSM, and SGM) $\times$ 2 (losses) = 30 adversarial versions. For CIFAR-100 and Tiny-ImageNet, we generate 10 (2 surrogate models $\times$ PGD with 5 random start) and 5 (1 surrogate model $\times$ PGD with 5 random start) adversarial versions respectively. We also report results under the white-box attack setting of a 50-step PGD with a step size of $\epsilon/5$ and 5 random starts. We choose this diverse set of attack methods to better quantify the model robustness.

We report both the black-box and white-box adversarial accuracy in an *all-or-nothing* fashion, i.e. a sample is judged to be correctly classified only if all of its adversarial samples using different attack methods are correctly classified. By default, we sample 1000 instances from the test set for evaluation.

[1]https://github.com/zjysteven/DVERGE

[2]https://github.com/yaodongyu/TRADES

[3]https://github.com/P2333/Bag-of-Tricks-for-AT

[4]https://github.com/deepmind/deepmind-research/tree/master/adversarial_robustness

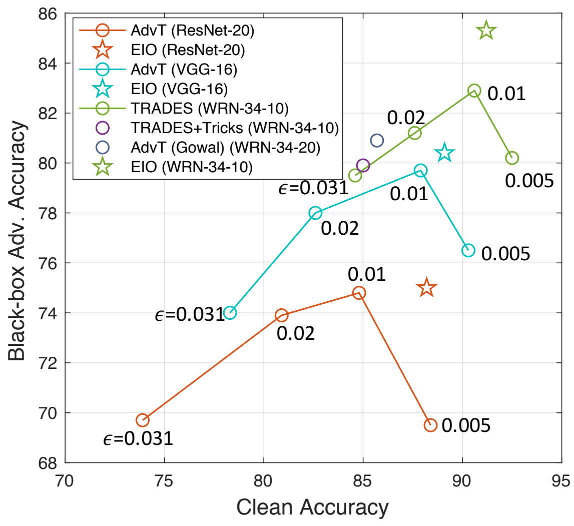


Figure 4: Comparison of EIO with state-of-the-art adversarial training methods with different training-time perturbation strength. X-axis: the clean accuracy; Y-axis: the black-box adversarial accuracy. The test-time perturbation strength is 0.031 (8/255).

Robustness Evaluation

Comparison with Ensemble Methods. Fig.3 shows the accuracy of the ensemble models trained by different methods with a wide range of attack perturbation strength. ResNet-20 and VGG-16 are selected as the networks to construct the ensembles and the EIO networks. The results show that through our Ensemble-in-One method, an individual model derived from the RGN, with the least computational cost, outperforms than each case of the heavy ensembles trained by previous ensemble training methods, simultaneously achieving comparable clean accuracy. The results suggest that EIO is a more effective realization of constructing large ensembles and brings significantly better robustness.

Comparison with Adversarial Training. AdvT has been demonstrated as a promising approach on enhancing the robustness. Prior work attributes the enhancement to the ex-

Model	Dataset	Method	Clean Acc.	Adv. Acc.	Inference Cost
ResNet-20	CIFAR-100	Baseline	67.1	0.7	1x
		AdvT($\epsilon = 0.03$)	49.6	42.4	1x
		AdvT($\epsilon = 0.02$)	56.1	47.8	1x
		AdvT($\epsilon = 0.01$)	61.6	49.7	1x
		EIO	**65.5**	**51.7**	1x
		GAL/3	68.6	42.9	3x
		DVERGE/3	**70.4**	46.4	3x
		EIO/3	69.4	**54.3**	3x
ResNet-18	Tiny-ImageNet	Baseline	50.3	4.3	1x
		AdvT($\epsilon = 0.03$)	31.0	28.4	1x
		AdvT($\epsilon = 0.02$)	35.1	31.4	1x
		AdvT($\epsilon = 0.01$)	39.5	31.4	1x
		EIO	**46.7**	**33.7**	1x
		GAL/3	51.1	9.6	3x
		DVERGE/3	**53.6**	13.1	3x
		EIO/3	50.5	**35.9**	3x

Table 2: Comparison of methods on CIFAR-100 and Tiny-ImageNet. The Adv. Acc. means the black-box adversarial accuracy with attack perturbation strength $\epsilon = 0.031$ (8/255). The number after the slash means the number of sub-models within corresponding ensembles during inference.

Method	Baseline	AdvT	DVERGE/3	DVERGE/5	DVERGE/8	EIO
Training Time	0.5h	4.2h	13.5h	26.6h	49.3h	10.9h

Table 3: The training cost of ResNet-20 on CIFAR-10 with different methods. The number after the slash means the number of sub-models within corresponding ensembles during inference.

Technique			Clean Acc	Adv. Acc
Distill	RGN	Mutual		
			91.2	0
✓			84.7 **(-6.5)**	69.8 **(+69.8)**
✓	✓		87.3 **(+3.6)**	70.3 **(+0.5)**
✓	✓	✓	88.3 **(+1.0)**	75.4 **(+5.1)**

Table 4: Ablation study on the involved techniques of EIO under the black-box setting with $\epsilon = 0.031$. "Distill" means training every sub-model (path) by its own adversarial samples. "RGN" means training within the super-net. "Mutual" means that every path learns the adversarial data distilled from other paths.

clusion of non-robust features during AdvT. However, these non-robust features might be *useful* to the classification accuracy, resulting in trade-offs between the clean accuracy and the robustness (Zhang et al. 2019). One can adjust the perturbation strength in the AdvT to acquire different combinations of clean accuracy and adversarial robustness, as shown in Fig.4. Three different networks are included. The Parato curves suggest that EIO consistently outperforms AdvT (including vanilla AdvT, TRADES, and TRADES with a bag of tricks) on both clean and adversarial accuracy under black-box transfer attacks.

Results on CIFAR-100 and Tiny-ImageNet. We further evaluate the methods on CIFAR-100 and Tiny-ImageNet, utilizing ResNet-20 and ResNet-18 as the basic networks. Table 2 shows that EIO consistently outperforms other ensemble training methods under black-box attack settings. Besides, we find that ensemble inference brings significant clean accuracy enhancements. We construct an EIO-based ensemble by sampling its sub-models from multiple independently trained RGNs. The simple aggregation of multiple EIO brings improvements, too. Overall, the results demonstrate that EIO is a more effective approach for achieving better trade-offs on adversarial robustness and clean accuracy.

Ablation Studies

Ablation Study on Techniques. As shown in Table 4, compared to the baseline, the adversarial feature distillation trades off helps clean accuracy for much higher adversarial robustness.

Then, the random gating mechanism helps achieve a significant increase on clean accuracy. This is because that the random gating prevents the network from over-fitting to the adversarial data, which works in a similar mechanism as dropout (Srivastava et al. 2014). Nevertheless, we emphasize that EIO has essential differences with dropout techniques from two aspects. The first is aim-aspect difference. EIO aims to construct many models to learn from each other. Solely because EIO shares their parameters for effi-

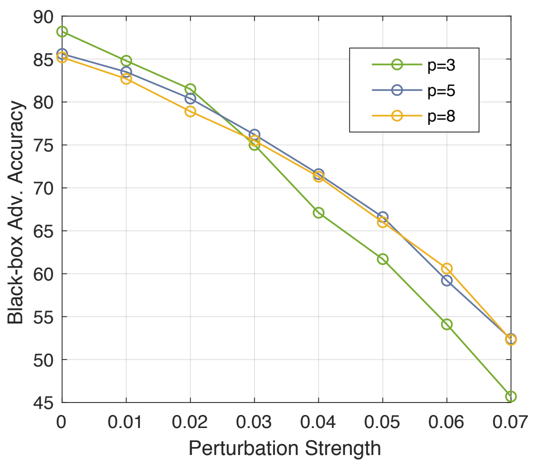


Figure 5: The black-box adversarial accuracy under different sample count p per iteration.

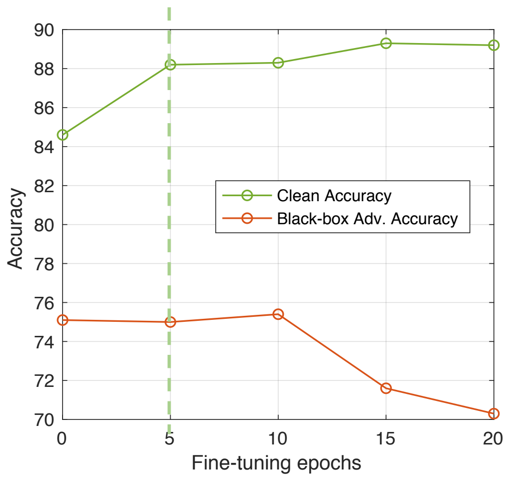


Figure 6: The black-box adversarial accuracy and clean accuracy under different epochs of fine-tuning.

ciency, it can be seen as dropping other paths when forwarding a path. The second is method-aspect difference. Vanilla dropout drops at parameter level from a single model, while EIO drops at module level from a supernet.

Finally, by incorporating mutual learning mechanism, the complete EIO solution can further improve the black-box adversarial accuracy as each path can learn from more distilled adversarial data generated from other paths.

Ablation Study on Sampling Path Count. Fig.5 shows the curves of black-box adversarial accuracy under different sampled path count p per training iteration. As is observed, when the sampled paths increase, the robustness of the derived individual model also improves. The underlying reason is that more samples of paths participating in each iteration allows more paths to be mutually trained, thereby each path is expected to learn from richer diversities. However, the clean accuracy drops with the increasing of path sample count, because a single operator has to adapt to more paths simultaneously. Moreover, the training time will also increase as the training complexity satisfies $\mathcal{O}(p^2)$. Overall, sampling 3 paths per iteration is the most practical choice.

Ablation Study on Fine-tuning. Recall that we apply a fine-tuning process after deriving a sub-model for deployment. The fine-tuning is normal training (i.e. training on clean examples in the train set) and optional while can effectively compensate for the under-convergence as the extracted sub-model was trained within a super-net. As shown in Fig.6, appropriate fine-tuning (e.g. fewer than 10 epochs) helps batch normalization layers capture more stable statistics and slightly adapts the weight parameters, thus helping the extracted model gain higher clean accuracy without sacrificing its robustness.

Training and Inference Cost

Table 3 summarizes the training time cost of different methods. As EIO applies a 10-step PGD-based adversarial feature distillation for training and sample 3 paths per training iteration, the training time cost of an EIO network is approximately 2.5× than AdvT. While the training time is substantially reduced compared to the DVERGE when scaling up the ensemble. For inference cost, EIO only sample one submodel for final deployment. Therefore, our method introduces no extra inference cost compared to AdvT or baseline methods, and incurs far smaller inference cost than the other ensemble training methods.

Model Derivation Stability

As in the deployment phase, an individual model will be derived from the RGN. Because the sampling is random, it is important to prove the stability of the derivation. Hence, we randomly extract 8 different sub-models from the same RGN and test their performance and robustness. Our experiments demonstrate the sampled sub-models are with slight fluctuations on the final performance. The standard derivations (stds) of the adversarial accuracy (under eps=0.031) are only 0.24% (for VGG-16) and 0.46% (for ResNet-20), which can prove that the final performance is not sensitive to the sampling randomness.

Conclusions

In this work, we propose Ensemble-in-One, an approach that constructs a random gated network (RGN) and conducts randomized ensemble training. Ensemble-in-One is inherently scalable, in which numerous sub-models can be instantiated by simply sampling different paths within the RGN. By diversifying the vulnerabilities of paths, Ensemble-in-One can efficiently obtain models with higher robustness, while keeping a small overhead of model training and deployment. The individual model derived from the RGN exhibits much better robustness than the ensemble model derived from previous ensemble training methods without sacrificing clean accuracy. Moreover, compared with the adversarial training methods, EIO achieves better trade-offs on the clean accuracy and the black-box robustness.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. U19B2019, 62104128, U21B2031, 61832007), National Key Research and Development Program of China (No. 2019YFF0301500), Tsinghua EE Xilinx AI Research Fund, and Beijing National Research Center for Information Science and Technology (BNRist). We thank Prof. Jianfei Chen for his valuable discussions. Finally, we thank all anonymous reviewers for their constructive suggestions.

References

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 274–283. PMLR.

Bagnall, A.; Bunescu, R.; and Stewart, G. 2017. Training ensembles to detect adversarial examples. *arXiv preprint arXiv:1712.04006*.

Brendel, W.; Rauber, J.; Kurakin, A.; Papernot, N.; Veliqi, B.; Mohanty, S. P.; Laurent, F.; Salathé, M.; Bethge, M.; Yu, Y.; et al. 2020. Adversarial vision challenge. In *The NeurIPS'18 Competition*, 129–153. Springer.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Carlini, N.; and Wagner, D. 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14.

Carlini, N.; and Wagner, D. 2017b. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

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

Dhillon, G. S.; Azizzadenesheli, K.; Lipton, Z. C.; Bernstein, J.; Kossaifi, J.; Khanna, A.; and Anandkumar, A. 2018. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*.

Dong, Y.; Liao, F.; Pang, T.; Hu, X.; and Zhu, J. 2017. Discovering adversarial examples with momentum. *arXiv preprint arXiv:1710.06081*.

Feinman, R.; Curtin, R. R.; Shintre, S.; and Gardner, A. B. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gowal, S.; Qin, C.; Uesato, J.; Mann, T.; and Kohli, P. 2020. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; and Madry, A. 2019. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*.

Kariyappa, S.; and Qureshi, M. K. 2019. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*, 32–33.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

Liu, L.; Wei, W.; Chow, K.-H.; Loper, M.; Gursoy, E.; Truex, S.; and Wu, Y. 2019. Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In *2019 IEEE 16th international conference on mobile ad hoc and sensor systems (MASS)*, 274–282. IEEE.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Ning, X.; Tang, C.; Li, W.; Zhou, Z.; Liang, S.; Yang, H.; and Wang, Y. 2021. Evaluating Efficient Performance Estimators of Neural Architectures. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Pang, T.; Xu, K.; Du, C.; Chen, N.; and Zhu, J. 2019. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, 4970–4979. PMLR.

Pang, T.; Yang, X.; Dong, Y.; Su, H.; and Zhu, J. 2020. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, 4095–4104. PMLR.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Tramèr, F.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*.

Wu, B.; Chen, J.; Cai, D.; He, X.; and Gu, Q. 2021. Do wider neural networks really help adversarial robustness? *Advances in Neural Information Processing Systems*, 34: 7054–7067.

Wu, D.; Wang, Y.; Xia, S.-T.; Bailey, J.; and Ma, X. 2020. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv preprint arXiv:2002.05990*.

Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; and Yuille, A. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*.

Xie, C.; Zhang, Z.; Zhou, Y.; Bai, S.; Wang, J.; Ren, Z.; and Yuille, A. L. 2019. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2730–2739.

Yang, H.; Zhang, J.; Dong, H.; Inkawhich, N.; Gardner, A.; Touchet, A.; Wilkes, W.; Berry, H.; and Li, H. 2020. DVERGE: diversifying vulnerabilities for enhanced robust generation of ensembles. *arXiv preprint arXiv:2009.14720*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 7472–7482. PMLR.