DVERGE: Diversifying Vulnerabilities for Enhanced Robust Generation of Ensembles

Huanrui Yang[1]*, **Jingyang Zhang**[1]*, **Hongliang Dong**[1]*, **Nathan Inkawhich**[1], **Andrew Gardner**[2], **Andrew Touchet**[2], **Wesley Wilkes**[2], **Heath Berry**[2], **Hai Li**[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Radiance Technologies
[1]{huanrui.yang, jz288, hongliang.dong, nai2, hai.li}@duke.edu,
[2]{andrew.gardner, atouchet, Wesley.Wilkes, Heath.Berry}@radiancetech.com

Abstract

Recent research finds CNN models for image classification demonstrate overlapped adversarial vulnerabilities: adversarial attacks can mislead CNN models with small perturbations, which can effectively transfer between different models trained on the same dataset. Adversarial training, as a general robustness improvement technique, eliminates the vulnerability in a single model by forcing it to learn robust features. The process is hard, often requires models with large capacity, and suffers from significant loss on clean data accuracy. Alternatively, ensemble methods are proposed to induce sub-models with diverse outputs against a transfer adversarial example, making the ensemble robust against transfer attacks even if each sub-model is individually non-robust. Only small clean accuracy drop is observed in the process. However, previous ensemble training methods are not efficacious in inducing such diversity and thus ineffective on reaching robust ensemble. We propose DVERGE, which isolates the adversarial vulnerability in each sub-model by distilling non-robust features, and diversifies the adversarial vulnerability to induce diverse outputs against a transfer attack. The novel diversity metric and training procedure enables DVERGE to achieve higher robustness against transfer attacks comparing to previous ensemble methods, and enables the improved robustness when more sub-models are added to the ensemble. The code of this work is available at `https://github.com/zjysteven/DVERGE`.

1 Introduction

Recent discoveries of *adversarial attacks* cast doubt on the inherent robustness of convolutional neural networks (CNNs) [1, 2, 3]. These attacks, commonly referred to as *adversarial examples*, comprise precisely crafted input perturbations that are often imperceptible to humans yet consistently induce misclassification in CNN models. Moreover, previous research has demonstrated widespread *transferability* of adversarial examples, wherein adversarial examples generated against an arbitrary model can reliably mislead other unspecified deep learning models trained with the same dataset [4, 5, 6]. Ilyas et al. [5] conjecture the existence of robust and non-robust features within standard image classification datasets. Whereas humans may understand an image via "human-meaningful" robust features, which usually are insensitive to small additive noise, deep learning models are more prone to learning non-robust features. Non-robust features are highly correlated with output labels and help improve clean accuracy but are not visually meaningful and are sensitive to noise. Such dependency on non-robust features leads to *adversarial vulnerability* that is exploited by adversarial examples to mislead CNN models. Moreover, Ilyas et al. empirically show that CNN models independently

*Equal contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

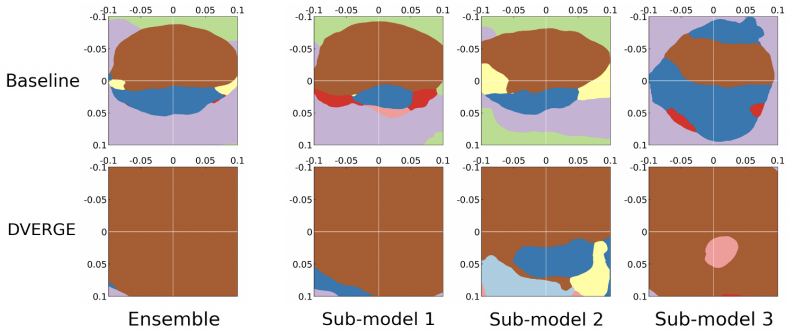

Figure 1: Decision regions in the ℓ_∞ ball around the same testing image learned by ensembles of 3 ResNet-20 models trained on CIFAR-10 dataset. Same color indicates the same predicted label. The vertical axis is along the adversarial direction of a surrogate benign ensemble, and the horizontal axis is along a random Rademacher vector. The same axes are used for each subplot. Adversarial vulnerability can be inferred from the closest decision boundary and corresponding class. The baseline ensemble is achieved via standard training on clean data while the bottom ensemble is trained with DVERGE. More plots of this nature can be seen in **Appendix C.1**.

trained on the same dataset tend to capture similar non-robust features, demonstrating overlapping vulnerability [5]. This property can be observed from the example in the upper row of Figure 1, where an ensemble is trained on clean data and each of its sub-models are vulnerable along the same axis of a transfer attack. This similarity is key to the high transferability of adversarial attacks [5, 7].

Extensive research has been conducted to improve the robustness of CNN models against adversarial attacks, most notably *adversarial training* [3]. Adversarial training minimizes the loss of a CNN model on online-generated adversarial examples against itself at each training step. This process forces the model to prefer robust to non-robust features and thereby largely eliminates the model's vulnerability. Nevertheless, learning robust features is hard, so adversarial training often leads to a significant increase in the generalization error on clean testing data [8].

Similar to traditional ensemble methods like bagging [9] and boosting [10], which train an ensemble of weak learners with diverse predictions to improve overall accuracy, a recent line of research proposes to train an ensemble of individually non-robust sub-models that produce diverse outputs against transferred adversarial examples [11, 12, 13]. Intuitively, the approach can defend against black-box transfer attacks as an attack can succeed only when multiple sub-models converge towards the same wrong prediction [13]. Such an ensemble could also hypothetically achieve high clean accuracy since the training process doesn't exclude non-robust features. Various ensemble training methods have been explored, such as diversifying output logits' distributions [11, 12] or minimizing the cosine similarity between the input gradient direction of each sub-model [13]. Yet empirical results show that these diversity metrics are not very effective at inducing output diversity among sub-models, and thus the corresponding ensemble can hardly attain the desired robustness [14].

We note that black-box transfer attacks are prevalent in real-world applications where model parameters are not exposed to end users [6, 13]. Moreover, high clean accuracy is always desirable. We therefore seek an effective training method that mitigates attack transferability while maintaining high clean accuracy. Based on a close investigation of the cause of adversarial vulnerability in sub-models, we propose to *distill* the features learned by each sub-model corresponding to its vulnerability to adversarial examples and use the overlap between the distilled features to measure the diversity between sub-models. As adversarial examples exploit the vulnerability of sub-models, a small overlap between sub-models indicates that a successful adversarial example on one sub-model is unlikely to fool the other sub-model. Consequently, our method impedes attack transferability between sub-models and leads to diverse outputs against a transferred adversarial example. Based on this diversity metric, we propose ***D**iversifying **V**ulnerabilities for **E**nhanced **R**obust **G**eneration of **E**nsembles* (DVERGE), which uses a round-robin training procedure to distill and diversify the features corresponding to each sub-model's vulnerability. The proposed ensemble training method makes the following contributions:

- DVERGE can successfully isolate and diversify the vulnerability in each sub-model such that within-ensemble attack transferability is nearly eliminated;

- DVERGE can significantly improve the overall robustness of the ensemble against black-box transfer attacks without significantly impacting the clean accuracy;
- The diversity induced by DVERGE consistently improves robustness as the number of ensemble sub-models increases under equivalent evaluation conditions.

As shown in the bottom row of Figure 1, diverse vulnerabilities allowed to persist in each sub-model for high clean accuracy by DVERGE combine to yield an ensemble robust to transfer attacks. Our method can also be augmented with the adversarial training objective to yield an ensemble with both satisfying white-box robustness and higher clean accuracy compared to exclusively adversarial training. To the best of our knowledge, this work is the first to utilize distilled features for training diverse ensembles and quantitatively relate it to the robustness against adversarial attacks.

2 Related work

Adversarial attack and defense. The pervasiveness of adversarial examples highlights the vulnerability of modern CNN systems to malicious inputs. An adversarial attack usually applies an additive perturbation δ subject to some constraint S to an original input x to form the adversarial example $x_{adv} = x + \delta$. The goal of the attack is to find δ so that x_{adv} can maximize the loss $\mathcal{L}_\theta$ of some CNN model with parameters θ w.r.t. x's true label y. The attacker's objective can be formulated as $x_{adv} = x + \text{argmax}_{\delta \in S} \mathcal{L}_\theta(x + \delta, y)$. The constraint S typically ensures adversarial examples are visually indistinguishable from original inputs, which is often defined as $||\delta||_p \leq \epsilon$ for some perturbation strength ϵ and ℓ_p-norm, *e.g.* $p = 0$ [15], $p = 2$ [2], or $p = \infty$ [1, 3]. In this work, we focus on the attack bounded by the ℓ_∞ norm, which has become increasingly common in recent attack and defense research studies. Madry et al. [3] show that the attacker's objective can be effectively optimized in a multi-step projected gradient descent (PGD) manner, where in each step of the gradient update the achieved adversarial example is projected back into the constraint set S to make sure it complies with the ℓ_∞ norm constraint. The attack can be further strengthened via application of a random starting point [3] or consideration of the gradient's momentum information during the optimization [16, 17].

Various empirical methods have been investigated for improving model robustness. Among theses methods, adversarial training [3] has gained prominence for its reliability and effectiveness. Adversarial training generates adversarial examples while concurrently training CNN model(s) to minimize the loss on these adversarial examples. The objective of adversarial training is formulated as a min-max optimization: $\min_\theta \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} \mathcal{L}_\theta(x + \delta, y)]$, where the inner maximization is often conducted with PGD attacks for greater robustness [3]. Although recent research shows that PGD adversarial training encourages a model to capture robust features within datasets [5], the process is difficult and costly. The learning of robust feature detrimentally and significantly affects the accuracy of the model on clean data [8], and the model architecture needs to be much larger in order to compensate for the added complexity of the objective [3].

Ensemble for improved accuracy and uncertainty measurement. Traditionally, ensemble learning methods have been extensively studied to improve the performance of the model and tackle out-of-distribution uncertainty and generalization. With some early success on performance improvement with neural network ensemble [18], bagging [9] and boosting [10], Kuncheva et al. [19] make a thorough evaluation on the relationship between sub-model diversity and ensemble accuracy, and find that as a higher diversity generally improves ensemble accuracy, specially designed diversity metric and training algorithm are needed to induce a stronger relationship between the two. Later on it is also observed that deep neural network ensembles can be used for uncertainty estimation [20], where the average predicted probability estimated from the ensemble outputs can lead to a well-calibrated uncertainty estimation. Recent advances in the field include diversifying the sub-models with varitional information bottleneck and diversity-inducing adversarial loss to further improve the ensemble accuracy [21], and resolving the scalability issue of deep ensembles or Bayesian neural networks by reducing the memory and computation cost with shared-weight across sub-models and rank-1 parameterization [22, 23]. As the contribution of our work is mainly addressing the adversarial robustness issues, our training algorithm is orthogonal to these previous works. Yet we believe it would be a promising direction to combine our algorithm with other newly-proposed ensemble training methods, leading to a robust ensemble with higher accuracy and less memory and computation cost.

Ensemble of diverse sub-models for robustness. Given the success of ensemble methods, a recent line of work investigates improving the robustness of an ensemble of small sub-models (especially against transfer adversarial attacks). Such robust ensembles can be obtained not only by combining individually robust sub-models but also by eliminating adversarial vulnerabilities shared by different sub-models, be they robust or non-robust, so that attacks cannot transfer between the sub-models within the ensemble. Several works attempt to promote diversity in internal representations or outputs across sub-models to serve as a mechanism to limit adversarial transferability and improve ensemble robustness. Pang et al. [12] propose the ADP regularizer, which forces different sub-models to have high diversity in the non-maximal predictions. Kariyappa et al. [13] reduce the overlap between "Adversarial Subspaces" [24] of different sub-models by maximizing the cosine distance between each sub-model's gradient w.r.t. the input. Although the ideas behind these methods are intuitive for improving sub-model diversity, these diversity metrics do not in practice align well with diversifying the adversarial vulnerability shared by different sub-models. Thus training the ensemble with these diversity metrics does not lead to satisfying robustness against transferability between sub-models, and consequently the resulted ensemble is still highly non-robust [14]. An ensemble diversity metric that can effectively lead to low attack transferability and high overall robustness is still lacking.

3 Method

3.1 Vulnerability diversity metric

A recent study [5] reveals that non-robust features captured by deep learning models are highly sensitive to additive noise, which is the main cause of adversarial vulnerability in CNN models. Based on this observation, we propose to isolate the vulnerability of CNN models based on their distilled non-robust features. Let us take a CNN model f_i trained on dataset D as an example. We consider a *target* input-label pair $(x, y) \in D$ and another randomly-chosen independent *source* pair $(x_s, y_s) \in D$. Corresponding to the source image x_s, the distilled feature of the input image x by the l-th layer of f_i can be approximated with the *feature distillation* objective as [5]:

$$x'_{f_i^l}(x, x_s) = \underset{z}{\operatorname{argmin}} \left\| f_i^l(z) - f_i^l(x) \right\|_2^2, \; s.t. \; ||z - x_s||_\infty \leq \epsilon, \tag{1}$$

where $f_i^l(\cdot)$ denotes the output before the activation (e.g. ReLU) of the l-th hidden layer. This constrained optimization objective can then be optimized with PGD [3]. The distilled feature is expected to be visually similar to x_s rather than x but classified as the target class y since the same feature will be extracted from $x'_{f_i^l}$ and x by f_i. Such misalignment between the visual similarity and the classification result shows that $x'_{f_i^l}$ reflects the adversarial vulnerability of f_i when classifying x. Therefore, we define the *vulnerability diversity* between two models f_i and f_j as:

$$d(f_i, f_j) := \frac{1}{2} \mathbb{E}_{(x,y),(x_s,y_s),l} \left[\mathcal{L}_{f_i} \left(x'_{f_j^l}(x, x_s), y \right) + \mathcal{L}_{f_j} \left(x'_{f_i^l}(x, x_s), y \right) \right]. \tag{2}$$

Here $\mathcal{L}_f(x, y)$ denotes the cross-entropy loss of model f for an input-label pair (x, y). The expectation is taken over the independent uniformly random choices of $(x, y) \in D$, $(x_s, y_s) \in D$, and layer l of models f_i and f_j. Since the distilled feature has the same dimension as input images, this formulation can be evaluated on models with arbitrary architectures trained on the same dataset. As $x'_{f_i^l}(x, x_s)$ is visually uncorrelated with y, the cross entropy loss $\mathcal{L}_{f_j}(x'_{f_i^l}(x, x_s), y)$ is small only if f_j's vulnerability on x's non-robust features overlaps with that of f_i, and vice versa. So the formulation in Equation (2) effectively measures the vulnerability overlap between the two models. Note that the feature distillation process in Equation (1) can be considered a special case of generating an adversarial example from source image x_s with target label y. The diversity defined in Equation (2) therefore corresponds to the attack success rate when transferring adversarial examples between the two models in the same way as training cross-entropy loss corresponds to training accuracy.

3.2 Vulnerability diversification objective

As adversarial attacks are less likely to transfer between models with high vulnerability diversity, we propose to apply the metric defined in Equation (2) as an objective during the ensemble training to induce diverse sub-models and block transfer attacks. Equation (3) shows a straightforward way to incorporate the diversity metric into the training objective, where for each sub-model f_i, the

diversity between itself and all other sub-models f_j in the ensemble is maximized when minimizing the original cross-entropy loss:

$$\min_{f_i} \mathbb{E}_{(x,y)}[\mathcal{L}_{f_i}(x,y)] - \alpha \sum_{j \neq i} d(f_i, f_j). \tag{3}$$

As the formulation of $d(f_i, f_j)$ has no upper bound, directly maximizing it may ultimately lead to divergence. Thus, we revise the training objective as

$$\min_{f_i} \mathbb{E}_{(x,y)} \left[\mathcal{L}_{f_i}(x,y) + \alpha \sum_{j \neq i} \mathbb{E}_{(x_s,y_s),l} \left[\mathcal{L}_{f_i} \left(x'_{f_j^l}(x, x_s), y_s \right) \right] \right], \tag{4}$$

which has a stronger bound than Equation (3). The new objective not only encourages the increase of vulnerability diversity but also facilitates the correct classification of the distilled image as y_s. As such, the objective is well-posed and can be effectively optimized.

Furthermore, it should be noted that minimizing $\mathcal{L}_{f_i}(x'_{f_j^l}(x, x_s), y_s)$ can effectively contribute to the minimization of $\mathcal{L}_{f_i}(x_s, y_s)$ as the distilled image $x'_{f_j^l}(x, x_s)$ is close to the clean image x_s. Previous adversarial training research [3, 25] also show that it is not necessary to include the clean data loss in the objective. So we further simplify Equation (4) to

$$\min_{f_i} \mathbb{E}_{(x,y),(x_s,y_s),l} \sum_{j \neq i} \mathcal{L}_{f_i}(x'_{f_j^l}(x, x_s), y_s), \tag{5}$$

which is adopted for training individual sub-models in DVERGE. The objective in Equation (5) can be understood as training sub-model f_i with the adversarial examples generated for other sub-models. However, DVERGE is fundamentally different from adversarial training. Adversarial training process constantly trains a model on white-box attacks against itself and forces the model to capture the robust feature of the dataset. In DVERGE, Equation (5) can be minimized if f_i utilizes a different set of features from other sub-models, including non-robust features. As non-robust features are distributed more commonly in dataset than robust features [5], capturing and integrating some non-robust features allows DVERGE to reach higher clean accuracy compared to adversarial training. Our training process should also be distinguished from that of Tramer et al. [26], which trains a single model with adversarial examples transferred from an ensemble of static pretrained sub-models for improving robustness. In DVERGE, all the sub-models in the ensemble are being optimized with Equation (5) in a round-robin fashion. The procedure dynamically maximizes the diversity of every pair of sub-models, rather than forcing only a single model away from static pretrained sub-models. The entire training process of DVERGE is elaborated in Section 3.3.

3.3 DVERGE training routine

Algorithm 1 shows the pseudo-code for training an ensemble of N sub-models. We first randomly initialize and pre-train all the sub-models based on the clean dataset so that their feature spaces will be useful and we do not waste time diversifying irrelevant features. Though we show in **Appendix C.6** that training DVERGE from scratch can also lead to better results than other methods. Then for each batch of training data during the diverse training phase, we randomly sample another batch of source data and use them to distill the non-robust feature following the objective of Equation (1). A PGD optimization scheme is applied in the feature distillation process. Round-robin training is then employed wherein a single stochastic gradient descent step is performed on each sub-model with the distilled images from all other sub-models and their source labels, as stated in the objective in Equation (5). This training process is performed on all B batches of training data and repeated for E epochs. The layer l for the feature distillation is randomly chosen in each epoch to avoid overfitting to the features of a particular layer. An ablation study along with discussion on this choice is given in **Appendix C.5**. This training routine can effectively increase the vulnerability diversity between each pair of sub-models within the ensemble and block within-ensemble transfer attacks. Consequently, the overall black-box robustness of the ensemble improves.

DVERGE induces a similar training complexity as adversarial training does. Both of these methods need extra back propagations to either distill non-robust features or find adversarial examples. However, DVERGE uses only intermediate features rather than final outputs for distillation so it is marginally faster than adversarial training. Detailed comparison of the training complexity of DVERGE vs. previous methods can be found in **Appendix A**.

Algorithm 1 DVERGE training routine for a N-sub-model ensemble.

```
# initialization and pretraining
for i = 1, ..., N do
    Randomly initialize sub-model f_i
    Pretrain f_i with clean dataset
# round-robin feature diversification
for e = 1, ..., E do
    Uniformly randomly choose layer l for feature distillation
    for b = 1, ..., B do
        (X, Y) ← get batched input-label pairs
        (X_s, Y_s) ← uniformly sample batched source input-label pairs
        # get distilled batch for each model
        for i = 1, ..., N do
            X'_i := x'_{f_i^l}(X, X_s) ← non-robust feature distillation with Equation (1)
        # calculate loss and perform SGD update for all sub-models
        for i = 1, ..., N do
            ∇_{f_i} ← ∇[Σ_{j≠i} L_{f_i}(f_i(X'_j), Y_s)]
            f_i ← f_i − lr · ∇_{f_i}
```

4 Experimental results

4.1 Setup

We compare DVERGE with various counterparts, including *Baseline* which trains an ensemble in a standard way and two previous robust ensemble training methods: *ADP* [12] and *GAL* [13]. For a fair comparison, we use ResNet-20 [27] as sub-models and average the output probabilities after the soft-max layer of each sub-model to yield the final predictions of ensembles. All the evaluations are performed on the CIFAR-10 dataset [28]. Training configuration details can be found in **Appendix A**. For DVERGE, we use PGD with momentum [29] to perform the feature distillation in Equation (1). We conduct 10 steps of gradient descent during feature distillation with a step size of $\epsilon/10$. The ϵ used for each ensemble size to achieve the results in this section was empirically chosen for the highest diversity and lowest transferability, such that $\epsilon = 0.07, 0.05, 0.05$ for ensembles with 3, 5, and 8 sub-models, respectively. Analysis on the effect of ϵ is given in **Appendix B**. Codes for reproducing the experiments are available at `https://github.com/zjysteven/DVERGE`.

4.2 Diversity and transferability within the ensemble

The objective of DVERGE is to guide sub-models to capture diverse non-robust features and minimize the vulnerability overlap between sub-models, thereby reducing the attack transferability within the ensemble. To validate our method, we measure diversity and transferability by randomly picking 1,000 test samples on which all sub-models initially give correct predictions. We compute the pair-wise diversity as the expected cross-entropy loss formulated in Equation (2), which is further averaged across all pairs of sub-models to obtain the diversity measurement of the whole ensemble. When measuring the transferability, we generate untargeted adversarial examples using 50-step PGD with a step size of $\epsilon/5$ and five random starts. The transferability is measured by the attack success rate which counts any misclassification as a success. Similar to diversity, the averaged pair-wise attack success rate is used to indicate the level of transferability within the ensemble.

First, let's take a look at how the diversity and transferability within the ensemble changes during the training process of DVERGE. Figure 2 shows the result of an ensemble of three sub-models. The diversity is evaluated using the same ϵ of 0.07 as during the training, and the transferability is measured using the standard ϵ of 0.03 ($\approx 8/255$) for adversarial attacks on CIFAR-10 [3]. The figure clearly shows that the diversity increases while the transferability decreases as the training proceeds. This trend empirically validates that minimizing the DVERGE objective can effectively lead to a higher diversity and a lower adversarial transferability within an ensemble.

Figure 3 presents the pair-wise transferability of an ensemble with three sub-models tested under the same ϵ as aforementioned. Results for the ensembles composed of more sub-models and for other testing ϵ are reported in **Appendix B** and **Appendix C.2**, respectively. The number at the

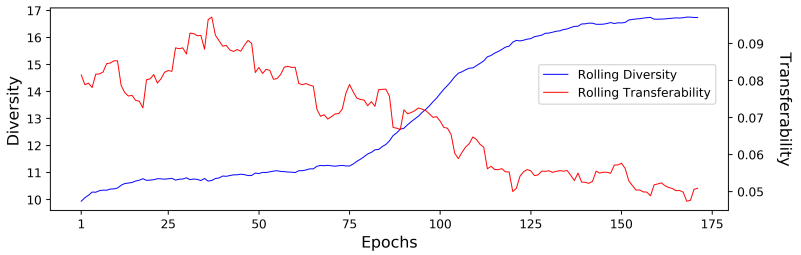

Figure 2: The trend of diversity and transferability during the training of DVERGE. The results are rolling averaged with a window size of 30.

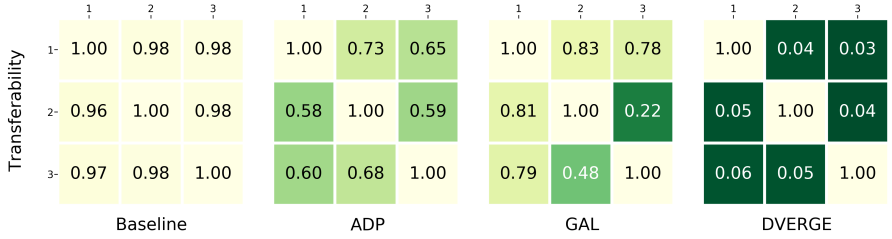

Figure 3: Pair-wise transferability (in the form of attack success rate) among sub-models for different ensemble methods.

intersection of the i-th row and j-th column represents the transfer success rate of the adversarial examples generated from the i-th sub-model and tested on the j-th sub-model. When $i = j$, the number becomes the white-box attack success rate. Larger off-diagonal numbers indicate greater transferability across sub-models. Compared with other ensemble methods, DVERGE suppresses the transferability to a much lower level; among all adversarial examples that successfully break one sub-model, only 3-6% of them could lead to misclassification on other sub-models. Although ADP and GAL also strive to improve the diversity for better robustness, they cannot effectively block adversarial transfer. ADP exhibits transference of 60% to 70% of attacks between sub-models. When it comes to GAL, two out of the three sub-models are still extremely vulnerable to each other, where more than 80% of adversarial examples can successfully transfer between the first and second sub-models. Our evaluation demonstrates that stopping attack transfers is not trivial and applying an appropriate diversification metric is crucial. Therefore, we advocate the use of DVERGE as a more effective means for mitigating the attack transferability within an ensemble.

4.3 Robustness of the ensemble

We evaluate the robustness of ensembles under two threat models: black-box transfer adversary, where the attackers cannot access the model parameters and rely on surrogate models to generate transferable adversarial examples, and white-box adversary, where the attackers have the full access of everything of the model. Under the black-box scenario, we use hold-out baseline ensembles with

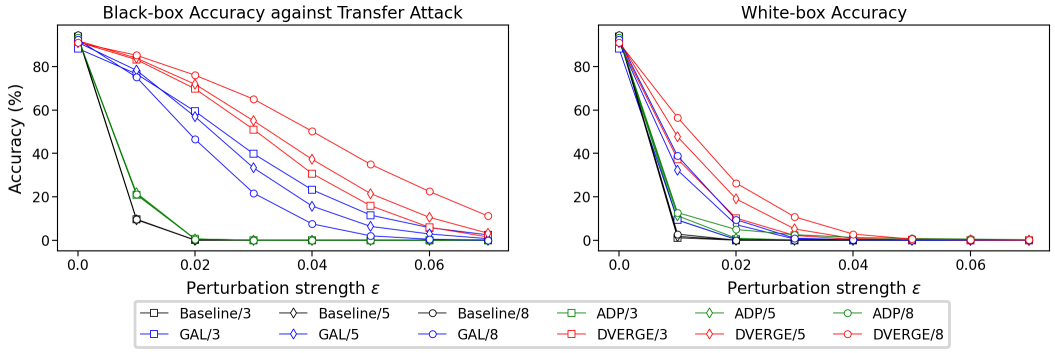

Figure 4: Robustness results for different ensemble methods. The number after the slash stands for the number of sub-models.

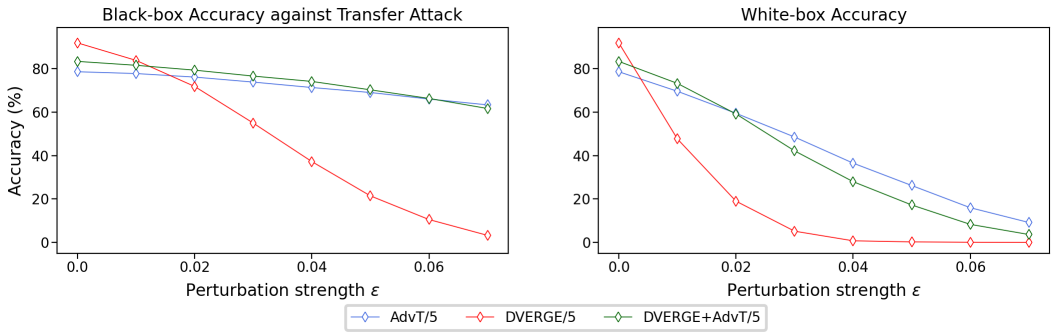

Figure 5: Results for DVERGE combined with adversarial training.

3, 5, and 8 ResNet-20 sub-models as the surrogate models. A more challenging setting considers an attacker fully aware of the defense such that the surrogate ensemble is trained with the exact technique. The results under this setting can be seen in **Appendix C.3**. We use three attack methodologies: (1) PGD with momentum [29] with three random starts. (2) M-DI^2- FGSM [30], which randomly resizes and pads the image in each step of attack generation. (3) SGM [31], which adds weight to the gradient through the skip connections of ResNets. The latter two attacks are essentially two stronger black-box transfer attacks that can better expose the attack transferability between models. For more details, we refer the reader to the attacks' respective papers. We run each attack for 100 iterations with the step size of $\epsilon/5$. Other than using the cross-entropy loss, we also generate adversarial examples with CW loss [2] since it can also help with the transfer. As a result, in total, each sample will have 3 (surrogate models)$\times$5 (PGD with 3 random starts plus 2 other attacks)$\times$2 (loss functions) = 30 adversarial counterparts. The black-box accuracy is reported in a *all-or-nothing* fashion: We say the model is accurate on one sample only if all of its 30 adversarial versions are correctly classified by the model. We adopt such a powerful adversary and a strict criteria to give a tighter upper bound of the robustness against black-box transfer attacks. Under the white-box scenario, we use 50-step PGD with five random starts and the step size of $\epsilon/5$ to attack ensembles. Our results in **Appendix D** prove that we have applied sufficient steps for attacks to converge.

Evaluated on 1,000 randomly selected test samples, Figure 4 shows the black-box and white-box robustness of ensembles with various number of sub-models across a wide range of attack budget ϵ. Here we show the averaged results over three independent runs. We refer the reader to **Appendix C.3** for the plots with error bars and numerical results. DVERGE, even with the least sub-models, outperforms each case of the other methods with higher accuracy in both black-box and white-box settings and achieves comparable clean accuracy. In addition, robustness improvement can be easily obtained by adding more sub-models into the ensemble when using our method, while such a satisfying trend is less obvious in other methods. GAL, as the second best performing approach among the four methods, actually shares the same high-level concept as the proposed DVERGE algorithm. They both aim at diversifying the vulnerabilities shared by the sub-models. The difference lies in the fact that GAL considers using the adversarial gradient directions to evaluate the vulnerability of CNN models whereas DVERGE identifies the vulnerability in a model by distilling the learnt non-robust features. Results from both Figure 3 and Figure 4 suggest that our approach is a more effective realization of the intuition of identifying and diversifying adversarial vulnerability.

4.4 DVERGE with adversarial training

Although DVERGE achieves the highest robustness among ensemble methods, its robustness against white-box attacks and transfer attacks with a large perturbation strength is still quite low. This result is expected because the objective of DVERGE is to diversify the adversarial vulnerability rather than completely eliminate it. In other words, vulnerability inevitably exists within sub-models and can be captured by attacks with larger ϵ. One straightforward way to improve the robustness of ensembles is to augment DVERGE with adversarial training [3]. We describe implementation details regarding adversarial training in **Appendix A** and the amended objective in **Appendix C.4**.

Figure 5 presents the black-box and white-box accuracy for adversarial training (*AdvT*), DVERGE only (*DVERGE*) and the combination of the two (*DVERGE+AdvT*) using the same evaluation setting as in Section 4.3. Ensembles with 5 sub-models are used here, and the results are averaged over three

independent runs. More results under different ensemble sizes and plots with error bars can be found in **Appendix C.4**. The DVERGE+AdvT objective favors the capture of more robust features by the ensemble. Compared to AdvT, DVERGE+AdvT encourages the ensemble to learn diverse non-robust features alongside robust features, leading to a higher clean accuracy and higher robustness against transfer attacks. In the meantime, no matter which objective is applied, the overall learning capacity of the ensemble remains the same. That is, learning more robust features will leave less capacity in the ensemble to capture diverse non-robust features, and vice versa. Forcing the inclusion of robust features causes DVERGE+AdvT to sacrifice the accuracy on clean examples comparing to performing DVERGE only. Learning diverse non-robust features harms DVERGE+AdvT's robustness against white-box attacks with larger perturbations compared to AdvT alone. These results can be seen as an evidence for the recent findings in [5, 8] regarding the tradeoff between clean accuracy and robustness. DVERGE+AdvT can effectively explore such tradeoff by changing the ratio between the two objectives, which is further illustrated in **Appendix C.4**.

5 Conclusions

In this work we propose DVERGE, a CNN ensemble training method that isolates and diversifies the adversarial vulnerability in each sub-model to improve the overall robustness against transfer attacks without significantly reducing clean accuracy. We show that adversarial diversity in a CNN model can be successfully characterized by distilled non-robust features, from which we can measure the vulnerability diversity between two models. The diversity metric is further developed into the vulnerability diversification objective used for DVERGE ensemble training. We empirically show that training with DVERGE objective can effectively increase the vulnerability diversity between sub-models, thereby blocking attack transferability within the ensemble. In this way DVERGE reduces the success rate of transfer attacks between sub-models from more than 60% achieved by previous ensemble training methods to less than 6%, which enables ensembles trained with DVERGE to achieve significantly higher robustness against both black-box transfer attacks and white-box attacks compared to previous ensemble training methods. The robustness can be further improved with additional sub-models in the ensemble. We further demonstrate that DVERGE can be augmented with an adversarial training objective, which enables the ensemble to achieve higher clean accuracy and higher transfer attack robustness compared to adversarial training. In conclusion, the vulnerability diversity induced by DVERGE training objective can effectively contribute to enhancing the robustness of CNN ensembles while maintaining desirable clean accuracy.

Broader Impact

DVERGE hypothetically addresses some black-box adversarial vulnerabilities pervasive across machine learning applications while increasing compute requirements to training models. As such methods presented herein suggest potential impacts on the reliability, security, and carbon-footprint of deep-neural-network-based systems. The reliability and robustness of machine learning systems are not just a concern for practitioners but also policy makers [32].

A net increase in carbon production would be considered a negative impact by many researchers in climate-related fields. This problem is common to many techniques that modify model training to achieve robustness, including DVERGE. While yet to be examined in the case of DVERGE, the possibility to mitigate or reduce excessive training burdens through informed hyperparameter selection exists. Sometimes, though modified training may increase the required computation per model parameter update, the modified method may nevertheless require fewer steps or epochs to achieve desirable results. Recent work provides actionable recommendations, such as performing cost-benefit analysis, to determine if efficient downstream adoption is desirable [33].

In both industrial and military applications, practical solutions to vulnerabilities, such as relying on human-AI teaming [34], are effective but do not address the underlying source of vulnerability and may limit the adoption of machine learning elsewhere. Addressing vulnerabilities at the training stage, then, is a desirable capability for positive-impact applications. By orthogonally improving only black-box robustness, though, we leave machine learning systems vulnerable to other types of attacks. Previous work has shown that white-box knowledge can still be leaked in black-box scenarios [35, 36]. As such, DVERGE is reliant on adversarial training to defend against white-box attacks and on traditional computer security to maintain system integrity. The ultimate interpretation of impact

due to improved model reliability and security is not clear-cut, however, as it is highly dependent on the application space. This uncertainty is symptomatic of the fact that machine learning is often fundamental by nature and that there is no machine learning technique for improving robustness that can be applied only to positive-impact applications, whatever one's subjective interpretation of "positive" may be.

Acknowledgments and Disclosure of Funding

This work is supported by the DARPA HR00111990079 (QED for RML) program.

References

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[3] 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.

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

[5] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[6] Nathan Inkawhich, Kevin Liang, Lawrence Carin, and Yiran Chen. Transferable perturbations of deep feature distributions. In *International Conference on Learning Representations*, 2020.

[7] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pages 196–212, 2015.

[8] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[11] Alexander Bagnall, Razvan Bunescu, and Gordon Stewart. Training ensembles to detect adversarial examples. *arXiv preprint arXiv:1712.04006*, 2017.

[12] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. *arXiv preprint arXiv:1901.08846*, 2019.

[13] Sanjay Kariyappa and Moinuddin K Qureshi. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981*, 2019.

[14] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

[15] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Discovering adversarial examples with momentum. *arXiv preprint arXiv:1710.06081*, 2017.

[17] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2253–2260, 2019.

[18] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

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

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[21] Samarth Sinha, Homanga Bharadhwaj, Anirudh Goyal, Hugo Larochelle, Animesh Garg, and Florian Shkurti. Dibs: Diversity inducing information bottleneck in model ensembles. *arXiv preprint arXiv:2003.04514*, 2020.

[22] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

[23] Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. *arXiv preprint arXiv:2005.07186*, 2020.

[24] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[25] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020.

[26] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

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

[28] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[29] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[30] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[31] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv preprint arXiv:2002.05990*, 2020.

[32] Ronan Hamon, Henrik Junklewitz, and Ignacio Sanchez. Robustness and explainability of artificial intelligence. 2020.

[33] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*, 2018.

[34] David Danks. How adversarial attacks could destabilize military ai systems. `https://spectrum.ieee.org/automaton/artificial-intelligence/embedded-ai/adversarial-attacks-and-ai-systems`, 2020. [Online; accessed 2-June-2020].

[35] Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 121–144. Springer, 2019.

[36] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

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

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[39] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[40] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

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