Improving Adversarial Robustness via Promoting Ensemble Diversity

Tianyu Pang [1] **Kun Xu** [1] **Chao Du** [1] **Ning Chen** [1] **Jun Zhu** [1]

Abstract

Though deep neural networks have achieved significant progress on various tasks, often enhanced by model ensemble, existing high-performance models can be vulnerable to adversarial attacks. Many efforts have been devoted to enhancing the robustness of individual networks and then constructing a straightforward ensemble, e.g., by directly averaging the outputs, which ignores the interaction among networks. This paper presents a new method that explores the interaction among individual networks to improve robustness for ensemble models. Technically, we define a new notion of ensemble diversity in the adversarial setting as the diversity among *non-maximal* predictions of individual members, and present an adaptive diversity promoting (ADP) regularizer to encourage the diversity, which leads to globally better robustness for the ensemble by making adversarial examples difficult to transfer among individual members. Our method is computationally efficient and compatible with the defense methods acting on individual networks. Empirical results on various datasets verify that our method can improve adversarial robustness while maintaining state-of-the-art accuracy on normal examples.

1. Introduction

Deep neural networks (DNNs) have recently achieved significant progress in many tasks, such as image classification (Russakovsky et al., 2015) and speech recognition (Graves & Jaitly, 2014). However, they are not problemless. Recent work has shown that high-performance classifiers are vulnerable to adversarial attacks (Goodfellow et al., 2015; Papernot et al., 2016b; Carlini & Wagner, 2017b; Athalye et al., 2018), where images with human imperceptible perturbations (i.e., adversarial examples) can be generated to fool the classifiers. To improve the adversarial robustness of classifiers, various kinds of defenses have been proposed (Kurakin et al., 2017b; 2018a; Kannan et al., 2018), and one practically effective strategy is to construct ensembles of the enhanced networks to obtain stronger defenses (Liao et al., 2018; Kurakin et al., 2018b).

However, most of these defense strategies focus on enhancing an individual network, while ignoring the potentially very informative interactions among multiple models. As widely observed, adversarial examples can have strong transferability among models (Papernot et al., 2016a; Kurakin et al., 2018b). When training an ensemble model, the ignorance of interaction among individual members may lead them to return similar predictions or feature representations (Dauphin et al., 2014; Li et al., 2016). Since it is easier for adversarial examples to transfer among similar models, it is probable for the adversarial examples crafted for one individual member to also fool other members, or even fool the whole ensemble model (See experiments).

Previous work has shown that for a single network, promoting the diversity among learned features of different classes can improve adversarial robustness (Pang et al., 2018a;b). In this paper, we propose to improve adversarial robustness from a new perspective by promoting the diversity among the predictions returned by different members of an ensemble, called *ensemble diversity*. Our approach is orthogonal to other defenses acting on a single network and thus is compatible with most of the previous defense methods.

Technically, we first develop a new notion of ensemble diversity in the adversarial setting, which is significantly different from the previous attempts in the standard setting of ensembling weak classifiers in a non-adversarial environment. Namely, previous work has defined ensemble diversity by considering the prediction errors of individual classifiers in the ensemble (Liu & Yao, 1999a;b; Dietterich, 2000; Islam et al., 2003). However, this definition is inappropriate for our case since most DNNs can achieve high accuracy on a training set (Goodfellow et al., 2016) and promoting diversity of prediction errors will largely sacrifice accuracy. Instead, we define the diversity on the *non-maximal* predictions of each network (detailed in Eq. (3)). Geometrically, the diversity

[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, Tsinghua-Fuzhou Inst. for Data Tech., THBI Lab, Tsinghua University, Beijing, 100084, China. Correspondence to: Ning Chen <ningchen@tsinghua.edu.cn>, Jun Zhu <dcszj@tsinghua.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

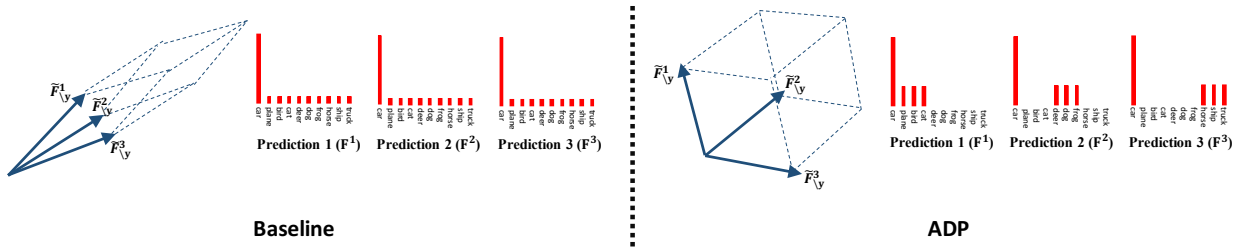


Figure 1. Illustration of the ensemble diversity. **Baseline:** Individually training each member of the ensemble. **ADP:** Simultaneously training all the members of the ensemble with the ADP regularizer. The left part of each panel is the normalized non-maximal predictions.

equals to the squared volume spanned by the normalized non-maximal predictions, as illustrated in Fig. 1. Note that promoting the diversity still allows the maximal prediction of each network to be consistent with the true label, and thus will not affect ensemble accuracy. Besides, since the non-maximal predictions correspond to all potentially wrong labels returned for the adversarial examples, a high diversity or inconsistency on the non-maximal predictions of individual networks can lower down the transferability of adversarial examples among them, and further lead to better robustness of the ensemble.

Under our definition of ensemble diversity, we propose to promote the diversity using an *adaptive diversity promoting* (ADP) regularizer. The ADP regularizer consists of two terms—a logarithm of ensemble diversity (LED) term and an ensemble entropy term. The ensemble entropy term contains the Shannon entropy of ensemble predictions. In Section 3.3, we prove that the ensemble entropy term is necessary such that the LED term can properly encourage the diversity. The ADP regularizer encourages the non-maximal predictions of each member in the ensemble to be mutually orthogonal, while keeping the maximal prediction be consistent with the true label, as shown in Fig. 1 and theoretically proved in Section 3.3. In the training procedure, we use the ADP regularizer as the penalty term and train all the networks in the ensemble simultaneously and interactively on the same dataset. This is called the ADP training procedure.

In experiments, we test our method under several widely studied adversarial attacks introduced in Section 2.2 on MNIST (LeCun et al., 1998), CIFAR-10, and CIFAR-100 (Krizhevsky & Hinton, 2009). The results show that our method can significantly improve adversarial robustness while maintaining state-of-the-art accuracy on normal examples. Although when performing back-propagation (Rumelhart et al., 1986) in the ADP training procedure, we need to separately perform the operations $\det(G)$ and G^{-1} with the computational complexity of $\mathcal{O}(K^3)$, where K is the number of members in the ensemble, K usually grows much slower with the scale of problem (Russakovsky et al., 2015). This enables our method to scale to most of the classification tasks and avoid the computational obstacle caused by these matrix operations as encountered in previous work (Kulesza et al., 2012; Kwok & Adams, 2012; Mariet & Sra, 2016).

2. Preliminary Knowledge

In this section, we first define the notations, then briefly introduce the adversarial attacks that we test in experiments.

2.1. Notations

We denote a deep neural network (DNN) classifier as $F(x, \theta)$, where x is the input variable, and θ denotes all the trainable parameters. Let L be the number of classes, the output vector $F(x, \theta) \in \mathbb{R}^L$ represents a probability distribution (Hereafter we will omit θ without ambiguity). When training a classifier, one common objective function is the cross-entropy (CE) loss defined as

$$\mathcal{L}_{\text{CE}}(x, y) = -1_y^\top \log F(x) = -\log F(x)_y,$$

for an input-label pair (x, y). Here, 1_y is the one-hot encoding of y and the logarithm of a vector is defined as taking logarithm of each element. Given a probability vector F, the Shannon entropy is defined as $\mathcal{H}(F) = -\sum_i F_i \log(F_i)$.

2.2. The Adversarial Setting

Recent work shows that high-performance classifiers could still be fooled by adversarial examples (Kurakin et al., 2018b; Athalye et al., 2018). A generated adversarial example x^* is usually indistinguishable from its normal counterpart x by a human observer. Here we introduce some of the most commonly used attack methods as below:

Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) crafts the adversarial example x^* as $x^* = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y))$, where $\mathcal{L}(x, y)$ is the adversarial loss.

Basic Iterative Method (BIM) (Kurakin et al., 2017a) iteratively crafts an adversarial example as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* + \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_{i-1}^*} \mathcal{L}(x_{i-1}^*, y)))$. Here $x_0^* = x$, r is the number of iteration steps and $\text{clip}_{x,\epsilon}(\cdot)$ is the clipping function.

Projected Gradient Descent (PGD) (Madry et al., 2018) has the same generation process as BIM, except for x_0^* is a randomly perturbed image in the neighborhood $\mathring{U}(x, \epsilon)$.

Momentum Iterative Method (MIM) (Dong et al., 2018) won the NeurIPS 2017 Adversarial Competition (Kurakin et al., 2018b), which is a variant of BIM. In each iteration, MIM updates the gradient g_i with momentum and crafts the

adversarial example as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* + \frac{\epsilon}{r} \cdot \text{sign}(g_i))$

Jacobian-based Saliency Map Attack (JSMA) (Papernot et al., 2016b) perturbs one feature x_i by ϵ in each iteration step that maximizes the saliency map

$$S(x,y)[i] = \begin{cases} 0, \text{ if } \frac{\partial F(x)_y}{\partial x_i} < 0 \text{ or } \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} > 0, \\ (\frac{\partial F(x)_y}{\partial x_i}) \left| \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} \right| \text{,otherwise.} \end{cases}$$

The adversarial noise crafted by JSMA is often sparser, i.e., JSMA perturbs fewer pixels compared to other attacks.

Carlini & Wagner (C&W) (Carlini & Wagner, 2017b) define $x^*(\omega) = \frac{1}{2}(\tanh(\omega) + 1)$ in terms of ω, and minimize the objective function $\min_\omega \|x^*(\omega) - x\|_2^2 + c \cdot f(x^*(\omega))$, where c is a constant chosen by a modified binary search. Let $\mathbb{S}_{pre}(x)$ be the input vector of the softmax function in a classifier, then $f(\cdot)$ is the objective function defined as

$$f(x) = \max(\max\{\mathbb{S}_{pre}(x)_i : i \neq y\} - \mathbb{S}_{pre}(x)_i, -\kappa),$$

where κ controls the confidence on adversarial examples.

Elastic-net Attack (EAD) (Chen et al., 2018) is a variant of the C&W method, where the objective function becomes $\min_\omega \|x^*(\omega) - x\|_2^2 + \beta \|x^*(\omega) - x\|_1 + c \cdot f(x^*(\omega))$. EAD includes C&W as a special case when $\beta = 0$.

3. Methodology

In this section, we first introduce the training strategies for ensemble models. Then we propose the adaptive diversity promoting (ADP) training method, and provide theoretical analyses on the optimal solutions of the ADP method.

3.1. Training Strategies for Ensemble Models

In real-world problems, ensembling several individual classifiers is usually effective to improve generalization performance (Russakovsky et al., 2015). For clarity, we denote the k-th member in the ensemble as $F^k \in \mathbb{R}^L$, which represents the prediction of classifier k over L categories. Then a common construction of the ensemble prediction $\mathcal{F}$ is a simple average of all individual predictions as

$$\mathcal{F} = \frac{1}{K} \sum_{k \in [K]} F^k, \tag{1}$$

where $[K] := \{1, \cdots, K\}$. There are three major strategies for training ensembles, i.e., independent training, simultaneous training, and sequential training (Islam et al., 2003).

In independent training, each classifier F^k is trained separately, usually with the corresponding CE loss $\mathcal{L}_{\text{CE}}^k(x,y) := -\log F^k(x)_y$. Because of its simplicity, most of the deep learning systems apply this strategy (Russakovsky et al., 2015; Liao et al., 2018). However, one major disadvantage of independent training is that it ignores the interaction among the members in the ensemble. This ignorance may lead to similar predictions or shared feature representations among these members (Dauphin et al., 2014; Li et al., 2016), which would not be useful to improve the performance of the ensemble (Krogh & Vedelsby, 1995), and result in less improvement on robustness as shown in our experiments.

In simultaneous training, all the classifiers are trained on the same mini-batch of data in each training iteration. A naive objective function for simultaneous training is the ensemble cross-entropy (ECE) loss defined as

$$\mathcal{L}_{\text{ECE}} = \sum_{k \in [K]} \mathcal{L}_{\text{CE}}^k, \tag{2}$$

which is a simple sum of all individual cross-entropy losses. Since the parameters for different networks are typically independent, simultaneously training all K networks with the ECE loss is equivalent to independently training each classifier with the corresponding CE loss. The learning rate for each network could be different. If some networks converge faster, then we can freeze the parameters of those converged networks and continue training others. Simultaneous training scheme allows us to incorporate the interaction among different members in the ensemble. Indeed, for large-scale tasks, simultaneous training may require more GPU memory or workers compared to independent training. However, it does not require extra computation. When the computing resource is limited, sequential training can be applied instead, which trains one network after another in each iteration.

In our method, we apply the simultaneous training scheme. Namely, we combine a novel regularizer (introduced in Eq. (5)) with the ECE loss term as the objective function for training. By doing this, the members in the ensemble can separately keep high accuracy since the ECE loss term, while interacting with each other via our regularizer.

3.2. Adaptive Diversity Promoting Training

Previous work has shown that for a single network, promoting the diversity among learned features of different classes can improve adversarial robustness (Pang et al., 2018a;b). In this paper, we promote diversity from another level, i.e., the diversity among learned predictions of different members in an ensemble, to improve robustness. This kind of diversity is called *ensemble diversity*. The previous analysis states that there is no strict definition of what is intuitively perceived as ensemble diversity (Kuncheva & Whitaker, 2003) and most of the previous work defines the ensemble diversity concerning the prediction errors of individual classifiers in the ensemble (Liu & Yao, 1999a;b; Dietterich, 2000; Islam et al., 2003). This definition partly bases on the assumption that each member in the ensemble is a weak classifier (Friedman et al., 2001). However, this assumption hardly holds for the ensemble of DNNs nowadays, thus encouraging the prediction errors of DNNs to be diverse is inappropriate and will largely sacrifice the accuracy of them.

In order to maintain the accuracy of each member in the ensemble, the maximal prediction returned by each member should be consistent with the true label. Therefore, in our method, we define the ensemble diversity with respect to the *non-maximal* predictions. For a single input pair (x, y) in the training set, the maximal prediction should be consistent with the true label y, and the non-maximal predictions correspond to other labels except for y. Inspired by the theory of the determinant point process (DPP) (Kulesza et al., 2012), we define the ensemble diversity as

$$\mathbb{ED} = \det(\tilde{M}_{\backslash y}^{\top}\tilde{M}_{\backslash y}). \tag{3}$$

Here $\tilde{M}_{\backslash y} = (\tilde{F}_{\backslash y}^{1}, \cdots, \tilde{F}_{\backslash y}^{K}) \in \mathbb{R}^{(L-1)\times K}$, each column vector $\tilde{F}_{\backslash y}^{k} \in \mathbb{R}^{L-1}$ is obtained by normalizing $F_{\backslash y}^{k}$ under L_2-norm, where $F_{\backslash y}^{k}$ is the order preserving prediction of the k-th classifier on x without the y-th element. According to the matrix theory (Bernstein, 2005), we have:

$$\det(\tilde{M}_{\backslash y}^{\top}\tilde{M}_{\backslash y}) = \mathrm{Vol}^2(\{\tilde{F}_{\backslash y}^{k}\}_{k\in[K]}), \tag{4}$$

where $\mathrm{Vol}(\cdot)$ denotes the volume spanned by the vectors of the input set. This provides an elegant geometric interpretation for the ensemble diversity, as intuitively shown in Fig. 1. Since $\tilde{F}_{\backslash y}^{k}$ is normalized, i.e., $\|\tilde{F}_{\backslash y}^{k}\|_2 = 1$, the ensemble diversity $\mathbb{ED}$ achieves its maximal value 1 if and only if the column vectors of $\tilde{M}_{\backslash y}$ are mutually orthogonal.

Note that most of the previous definitions of diversity, e.g., those defined on prediction errors, include the predictions F_y^k on the true label y (Liu & Yao, 1999a;b; Islam et al., 2003; Kuncheva & Whitaker, 2003). Since DNNs are no longer weak classifiers in the most cases (Goodfellow et al., 2016), encouraging F_y^k to be diverse may make the loss on accuracy outweighs the gain for DNNs, or render the convergence point of the training process uncontrollable. In contrast, we define the diversity among the normalized non-maximal predictions $\tilde{F}_{\backslash y}^{k}$, which allows the maximal prediction of each network to be consistent with the true label, and thus will not affect ensemble accuracy. What's more, promoting this diversity could improve robustness. Because, in the adversarial setting, when the maximal prediction corresponds to the true label y, the non-maximal predictions correspond to the labels $[L]\backslash\{y\}$, which include all potentially wrong labels returned for the adversarial examples. Thus a high diversity or inconsistency on the non-maximal predictions $F_{\backslash y}^{k}$ can lower down the transferability of adversarial examples among the networks, and further lead to better robustness of the ensemble.

To promote ensemble diversity, we propose the **adaptive diversity promoting (ADP)** regularizer as

$$\mathrm{ADP}_{\alpha,\beta}(x, y) = \alpha \cdot \mathcal{H}(\mathcal{F}) + \beta \cdot \log(\mathbb{ED}) \tag{5}$$

for a single input pair (x, y), where $\alpha, \beta \geq 0$ are two hyperparameters. Note that the ADP regularizer consists of two parts. The first part is the ensemble Shannon entropy $\mathcal{H}(\mathcal{F})$, and the second part is the logarithm of the ensemble diversity (LED), which we call as the LED part. More technical details about the effects of these two parts and why the ensemble entropy part is necessary will be deferred to Section 3.3. The training procedure with the ADP regularizer is called the ADP training method, as described in Algorithm 1. Further analysis on other potential definition of ensemble diversity is provided in Appendix B.1.

Algorithm 1 The ADP training procedure

Input: The K individual models $\{F^k(x, \theta^k)\}_{k\in[K]}$; the training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i\in[N]}$.
Initialization: Initialize each θ^k as θ_0^k, the training step counters as $c_k = 0$, ε_k be the learning rate variables, $I = [K]$ be the indicator set, where $k \in [K]$.
while $I \neq \varnothing$ **do**
  Calculate the objective on a mini-batch of data $\mathcal{D}_m$
  $$\mathcal{L}_{\mathrm{ADP}}^m = \frac{1}{|\mathcal{D}_m|} \sum_{(x_i,y_i)\in\mathcal{D}_m} [\mathcal{L}_{\mathrm{ECE}} - \mathrm{ADP}_{\alpha,\beta}](x_i, y_i).$$
  for $k^{'}$ in I **do**
    Update $\theta_{c+1}^{k} \leftarrow \theta_{c}^{k^{'}} - \varepsilon_{k^{'}} \nabla_{\theta^{k^{'}}} \mathcal{L}_{\mathrm{ADP}}^m\big|_{\{\theta_{c_k}^k\}_{k\in[K]}}$.
    Update $c_{k^{'}} \leftarrow c_{k^{'}} + 1$, where $c = c_{k^{'}}$.
    if $\theta^{k^{'}}$ converges **then** Update $I = I\backslash\{k^{'}\}$.
  end for
end while
Return: The parameters $\theta^k = \theta_{c_k}^k$, where $k \in [K]$.

When performing back-propagation (Rumelhart et al., 1986) in the ADP training procedure, we need to separately perform two matrix operations $\det(G)$ and G^{-1}. Without loss of generality, let $G \in \mathbb{R}^{m\times m}$, where m denote the dimension of the matrix, these two operations both have computational complexities of $\mathcal{O}(m^3)$ under common algorithms (Kulesza et al., 2012), which may cause computational obstacle if m scales comparably with the size of the problem. For example, in previous DPP-based methods used to sample diverse subsets (Kwok & Adams, 2012; Mariet & Sra, 2016), they need to calculate the values of $\det(G)$ to obtain the sampling probability for each subset, where m usually scales with the number of classes or number of hidden units in networks. However, in our method, we have $G = \tilde{M}_{\backslash y}^{\top}\tilde{M}_{\backslash y}$ and $m = K$, where K grows much slower with the scale of problem (Russakovsky et al., 2015). This property enables our method to scale to most of the modern machine learning tasks and avoid similar computational obstacle encountered in previous work. As experimentally shown in Section 4, our method is also compatible with other defenses acting on individual networks, e.g., adversarial training (Goodfellow et al., 2015; Madry et al., 2018), and thus our method could be an orthogonal approach to further improve robustness for ensemble models. These results verify that ensemble diversity is indeed an effective

component to consider when designing stronger defenses.

3.3. Theoretical Analyses on ADP Training

The minimization problem in the ADP training on a single input-label pair (x, y) can be formally denoted as

$$\min_{\theta} \mathcal{L}_{\text{ECE}} - \text{ADP}_{\alpha,\beta} \quad \text{s.t. } 0 \le F_j^k \le 1, \sum_{j\in[L]} F_j^k = 1, \tag{6}$$

where $\theta = \{\theta^k\}_{k\in[K]}$ denotes all the trainable parameters in the ensemble. For multiple input-label pairs, the objective function is simply a summation of each single function as in Algorithm 1. However, to simplify the analysis, we focus on the optimal solution in the space of output predictions $\mathbb{F} = \{F^k\}_{k\in[K]}$, and assume that the representation capacity of each network is unlimited (Hornik et al., 1989). Therefore, we ignore the specific mapping from x to each F^k, and redefine the above minimization problem with respect to $\mathbb{F}$, rather than θ. Below we will theoretically analyze how the values of the two hyperparameters α and β influence the solution for the minimization problem. If $\alpha = 0$, the ADP regularizer degenerates to the LED part. In this case, the solution will be one-hot vectors 1_y just as there is no regularizer, as shown in the following theorem:

Theorem 1. *(Proof in Appendix A) If $\alpha = 0$, then $\forall \beta \ge 0$, the optimal solution of the minimization problem (6) satisfies the equations $F^k = 1_y$, where $k \in [K]$.*

The conclusion means that the LED part in the ADP regularizer cannot work alone without the ensemble entropy part, indicating that the ensemble entropy part is necessary. In a nutshell, this is because we define the LED part on the *normalized* predictions $\tilde{F}^k_{\setminus y}$ corresponding to untrue labels. Thus the LED part can only affect the mutual angles of these untrue predictions. The summation of all unnormalized untrue predictions is $1 - F_y^k$, and in this case, the summation will only be influenced by the ECE loss part, which leads to one-hot optimal solution 1_y. On the other hand, if $\beta = 0$, the ADP regularizer degenerates to the ensemble entropy, which results in similar optimal solutions on ensemble prediction $\mathcal{F}$ as label smoothing (Szegedy et al., 2016). The equivalent smoothing factor is $1 - \mathcal{F}_y$, and the conclusions are formally stated in the following theorem:

Theorem 2. *(Proof in Appendix A) When $\alpha > 0$ and $\beta = 0$, the optimal solution of the minimization problem (6) satisfies the equations $F_y^k = \mathcal{F}_y$, $\mathcal{F}_j = \frac{1-\mathcal{F}_y}{L-1}$ and*

$$\frac{1}{\mathcal{F}_y} = \frac{\alpha}{K} \log \frac{\mathcal{F}_y(L-1)}{1-\mathcal{F}_y}, \tag{7}$$

where $k \in [K]$ and $j \in [L]\backslash\{y\}$.

Here Eq. (7) can assist us in selecting proper values for the hyperparameter α. For example, if we want to train an ensemble of 5 individual networks ($K = 5$) on ImageNet ($L = 1000$) with equivalent label smoothing factor 0.1 ($\mathcal{F}_y = 0.9$), then we can calculate from Eq. (7) that $\alpha \approx 0.61$. The difference of the solutions in Theorem 2 from those of label smoothing is that there is an extra constraint for each individual prediction: $F_y^k = \mathcal{F}_y$. This guarantees the consistency of all maximal predictions with the true label. Besides, it is worthy to note that for any $j \in [L]\backslash\{y\}$, the ensemble entropy part does not directly regularize on the non-maximal predictions F_j^k, but regularize on their mean, i.e., the ensemble prediction $\mathcal{F}_j$. This leaves degrees of freedom on the optimal solutions of F_j^k, and the effect of the LED part is to further regularize on them: if $L - 1$ is divisible by K, i.e., $K \mid (L-1)$, the non-maximal predictions of each network are mutually orthogonal, as intuitively shown in the right panel of Fig. 1, where $K = 3$ and $L = 10$. We mathematically denote the optimal solution in this case as in the following corollary:

Corollary 1. *If there is $K \mid (L-1)$, then $\forall \alpha, \beta > 0$, the optimal solution of the minimization problem (6) satisfies the Eq. (7). Besides, let $S = \{s_1, , \cdots, s_K\}$ be any partition of the index set $[L]\backslash\{y\}$, where $\forall k \in [K]$, $|s_k| = \frac{L-1}{K}$. Then the optimal solution further satisfies:*

$$F_j^k = \begin{cases} \frac{K(1-\mathcal{F}_y)}{L-1}, & j \in s_k, \\ \mathcal{F}_y, & j = y, \\ 0, & otherwise. \end{cases} \tag{8}$$

The index subset s_k includes all the indexes except for y that the k-th classifier predicts non-zero probability. Note that the partition S is adaptive concerning the input, which means that for two inputs x_1 and x_2 of the same label y, the partition S_1 and S_2 could be different. For example, for dog image #1 and dog image #2, an individual network may both assign 0.7 probability to label dog, and separately assign 0.1 probability to labels car, frog, ship for image #1 and plane, truck, cat for image #2. This is why the ADP regularizer is named as *adaptive*, and this adaptivity can avoid imposing an artificial bias on the relationship among different classes. When L is large, the optimal solution will still approximately satisfy the conclusion if $K \nmid (L-1)$. As shown in Table 5 of our experiments, even when L is not large ($L = 10$), the ADP method can still work well in the case of $K \nmid (L-1)$. After training by the ADP method, the non-maximal predictions of each network will tend to be mutually orthogonal, and further leads to different feature distributions and decision domains as shown in Fig. 2, where more technical details can be found in Section 4.2.

4. Experiments

We now provide the experimental results to demonstrate the effectiveness of the ADP training on improving adversarial robustness under different attack methods, while maintaining state-of-the-art accuracy on normal examples.

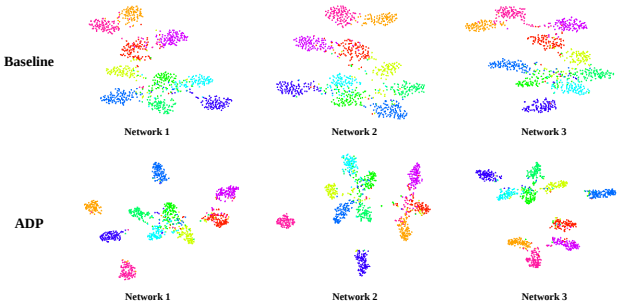


Figure 2. t-SNE visualization results on the final hidden features of each individual network. The input is the test set of CIFAR-10.

4.1. Setup

We choose three widely studied datasets—MNIST, CIFAR-10 and CIFAR-100 (LeCun et al., 1998; Krizhevsky & Hinton, 2009). MNIST consists of grey images of handwritten digits in classes 0 to 9, and CIFAR-10 consists of color images in 10 different classes. CIFAR-100 is similar to CIFAR-10 except that it has 100 classes. Each dataset has 50,000 training images and 10,000 test images. The pixel values of images are scaled to be in the interval $[0, 1]$. The normal examples in our experiments refer to all the ones in the training and test sets. The code is available at https://github.com/P2333/Adaptive-Diversity-Promoting.

4.2. Performance on Normal Examples

We first evaluate the performance in the normal setting. On each dataset, we implement the ensemble model consisting of three Resnet-20 networks (He et al., 2016), where $K = 3$ satisfies the condition $K \mid (L-1)$ for $L = 10$ and $L = 100$. In Section 4.6, we will investigate on the case when $K \nmid (L-1)$. Our baseline training method is to train the ensemble with the ECE loss, which is equivalent to ADP training with $\alpha = \beta = 0$. As analyzed in Section 3.1, simultaneously training all networks with the ECE loss is also equivalent to independently training each one with the corresponding CE loss, which is the most commonly applied strategy on training DNN ensembles (Russakovsky et al., 2015; Goodfellow et al., 2016). About our method, we test two hyperparameter settings: **ADP**$_{2,0}$ sets $\alpha = 2$ and $\beta = 0$ corresponding to the case in Theorem 2, where the value of α is chosen according to Eq. (7). This is a reference setting to demonstrate the effect of the ensemble entropy part alone in the ADP regularizer. **ADP**$_{2,0.5}$ sets $\alpha = 2$ and $\beta = 0.5$ corresponding to the case in Corollary 1, which is a full version of our method. Later we may omit the subscript as **ADP** to denote this setting without ambiguity.

In Table 1, we show the test error rates on each dataset. For all the three training settings, we use the same training hyperparameters, e.g., learning rate, training epoch, etc. The weight initialization is different for each network. We apply Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.001. Following similar setting as in He et al. (2016), we separately run the training process for 40 epochs on MNIST, 180 epochs on CIFAR-10 and CIFAR-100 with the mini-batch size of 64 on a Tesla P100 GPU worker. As Table 1 shows, although the ADP training causes higher error rates on individual networks, it leads to lower error rates on the ensembles. This verifies that promoting the ensemble diversity as defined in Eq. (3) is also effective to improve ensemble performance in the normal setting (Krogh & Vedelsby, 1995; Tsymbal et al., 2005). To investigate the influence of the ADP training on the feature distributions, we use the t-SNE method (Maaten & Hinton, 2008) to visualize the final hidden features of each network in Fig. 2. We can see that when trained by the baseline method, the three individual networks have learned similar feature distributions. In contrast, when trained by the ADP method, the learned feature distributions are much more different, which makes it more difficult for adversarial examples to transfer among individual networks as demonstrated in Section 4.4.

Table 1. Classification error rates (%) on the test set of each dataset. The ensemble model consists of three Resnet-20 networks.

Dataset	Classifier	Baseline	ADP$_{2,0}$	ADP$_{2,0.5}$
MNIST	Net 1	0.44	0.43	0.36
	Net 2	0.39	0.45	0.47
	Net 3	0.43	0.37	0.47
	Ensemble	0.32	0.32	**0.28**
CIFAR-10	Net 1	8.30	9.01	9.34
	Net 2	8.52	9.15	9.34
	Net 3	8.67	9.14	9.92
	Ensemble	6.78	6.74	**6.56**
CIFAR-100	Net 1	35.25	-	39.04
	Net 2	35.91	-	40.86
	Net 3	36.03	-	39.00
	Ensemble	30.35	-	**29.80**

Furthermore, to demonstrate that the ADP method only results in a marginal increase in training time, we test the training time per epoch with the mini-batch size of 32 on CIFAR-10. When $K = 5$, it takes 204s/epoch for the baseline method, and 227s/epoch for the ADP methods; when $K = 20$, it takes 692s/epoch for baseline, and 744s/epoch for ADP. Thus the ADP methods only increase around 10% training time compared to baseline even when $K = 20$.

4.3. Performance under White-box Attacks

In the adversarial setting, there are mainly two threat models (Carlini & Wagner, 2017a): *white-box adversaries* know all the information about the classifier models, including training data, model architectures and parameters; *black-box adversaries* know the classification tasks, but have no access to the information about the classifier models.

In this sub-section, we test the performance of ensemble models defending white-box attacks. We apply the attack methods introduced in Section 2.2. The results are reported in Table 2 and Table 3, where we test each attack on two or three representative values of parameters. The iteration

Table 2. Classification accuracy (%) on adversarial examples. Ensemble models consist of three Resnet-20. For JSMA, the perturbation $\epsilon = 0.2$ on MNIST, and $\epsilon = 0.1$ on CIFAR-10. For EAD, the factor of L_1-norm $\beta = 0.01$ on both datasets.

	MNIST				CIFAR-10			
Attacks	Para.	Baseline	$\text{ADP}_{2,0}$	$\text{ADP}_{2,0.5}$	Para.	Baseline	$\text{ADP}_{2,0}$	$\text{ADP}_{2,0.5}$
FGSM	$\epsilon = 0.1$	78.3	95.5	**96.3**	$\epsilon = 0.02$	36.5	57.4	**61.7**
	$\epsilon = 0.2$	21.5	50.6	**52.8**	$\epsilon = 0.04$	19.4	41.9	**46.2**
BIM	$\epsilon = 0.1$	52.3	86.4	**88.5**	$\epsilon = 0.01$	18.5	44.0	**46.6**
	$\epsilon = 0.15$	12.2	69.5	**73.6**	$\epsilon = 0.02$	6.1	28.2	**31.0**
PGD	$\epsilon = 0.1$	50.7	73.4	**82.8**	$\epsilon = 0.01$	23.4	43.2	**48.4**
	$\epsilon = 0.15$	6.3	36.2	**41.0**	$\epsilon = 0.02$	6.6	26.8	**30.4**
MIM	$\epsilon = 0.1$	58.3	89.7	**92.0**	$\epsilon = 0.01$	23.8	49.6	**52.1**
	$\epsilon = 0.15$	16.1	73.3	**77.5**	$\epsilon = 0.02$	7.4	32.3	**35.9**
JSMA	$\gamma = 0.3$	84.0	88.0	**95.0**	$\gamma = 0.05$	29.5	33.0	**43.5**
	$\gamma = 0.6$	74.0	85.0	**91.0**	$\gamma = 0.1$	27.5	32.0	**37.0**
C&W	$c = 0.1$	91.6	95.9	**97.3**	$c = 0.001$	71.3	76.3	**80.6**
	$c = 1.0$	30.6	75.0	**78.1**	$c = 0.01$	45.2	50.3	**54.9**
	$c = 10.0$	5.9	20.2	**23.8**	$c = 0.1$	18.8	19.2	**25.6**
EAD	$c = 5.0$	29.8	91.3	**93.4**	$c = 1.0$	17.5	64.5	**67.3**
	$c = 10.0$	7.3	87.4	**89.5**	$c = 5.0$	2.4	23.4	**29.6**

Table 3. Classification accuracy (%) on adversarial examples. Ensemble models consist of three Resnet-20.

	CIFAR-100		
Attacks	Para.	Baseline	$\text{ADP}_{2,0.5}$
BIM	$\epsilon = 0.005$	21.6	**26.1**
	$\epsilon = 0.01$	10.1	**14.8**
PGD	$\epsilon = 0.005$	26.6	**32.1**
	$\epsilon = 0.01$	11.7	**18.3**
MIM	$\epsilon = 0.005$	24.2	**29.4**
	$\epsilon = 0.01$	11.2	**17.1**

steps are set to be 10 for BIM, PGD, and MIM, with the step size equals to $\epsilon/10$. The iteration steps are set to be $1,000$ for C&W and EAD, with the learning rate of 0.01. We separately test the ensemble models trained by the baseline, $\text{ADP}_{2,0}$ and $\text{ADP}_{2,0.5}$ methods. As expected, the results verify that our method significantly improves adversarial robustness. However, the improvements are more prominent on MNIST and CIFAR-10 than those on CIFAR-100. This can be explained by Eq. (8): the non-maximal predictions scale with the factor $\frac{1}{L-1}$, which could result in a numerical obstacle in the training phase when L is large. A potential way to solve this problem is to use the temperature scaling method (Guo et al., 2017), as discussed in Appendix B.2. An interesting phenomenon is that when the LED part is inactive as in the $\text{ADP}_{2,0}$ setting, the learned ensemble diversity is still much larger than the baseline (detailed in Appendix B.3). This is because the ensemble entropy part expands the feasible space of the optimal solution, while leaving degrees of freedom on the optimal non-maximal predictions F^k_j, as shown in Theorem 2. Then the ensemble diversity is unlikely to be small in this case. However, the existence of the LED part further explicitly encourages the ensemble diversity, which leads to better robustness.

To show that our method is an orthogonal approach concerning other defenses acting on a single network, we test the compatibility of our method with adversarial training (AdvT), which is the most widely studied defense method (Kurakin et al., 2018b). AdvT augments the training data with adversarial examples in each mini-batch, where the two quite common attacks used to craft these adversarial examples are FGSM (Goodfellow et al., 2015) and PGD (Madry et al., 2018). In Table 4, we demonstrate the classification accuracy of enhanced ensemble models on CIFAR-10. We uniformly sample the perturbation ϵ from the interval $[0.01, 0.05]$ when performing AdvT as in Kurakin et al. (2017b). The ratio of adversarial examples and normal ones in each mini-batch is 1:1, with the batch size of 128. The results demonstrate that our method can further improve robustness for ensemble models, with little extra computation and is universal to different attacks. Besides, when comparing the results in Table 2 and Table 4, we can find that AdvT can also further improve the performance of our method, which means they are indeed complementary.

Table 4. Classification accuracy (%): $\textbf{AdvT}_{\textbf{FGSM}}$ denotes adversarial training (AdvT) on FGSM, $\textbf{AdvT}_{\textbf{PGD}}$ denotes AdvT on PGD. $\epsilon = 0.04$ for FGSM; $\epsilon = 0.02$ for BIM, PGD and MIM.

	CIFAR-10			
Defense Methods	FGSM	BIM	PGD	MIM
$\text{AdvT}_{\text{FGSM}}$	39.3	19.9	24.2	24.5
$\text{AdvT}_{\text{FGSM}} + \text{ADP}_{2,0.5}$	**56.1**	**25.7**	**26.7**	**30.6**
AdvT_{PGD}	43.2	27.8	32.8	32.7
$\text{AdvT}_{\text{PGD}} + \text{ADP}_{2,0.5}$	**52.8**	**34.0**	**36.2**	**38.8**

4.4. Transferability among Individual Models

Due to the transferability of adversarial examples among models (Papernot et al., 2016b), black-box adversaries can craft adversarial examples based on substitute models and then feed these examples to original models to perform the attack. In this sub-section, we show that the ADP training can mitigate the transferability of adversarial examples among the members in the ensemble, as demonstrated in Fig. 3. For each matrix, the (i, j) element is the result of using i-th network as the substitute model to craft adversarial examples and feeding to j-th network as the original model.

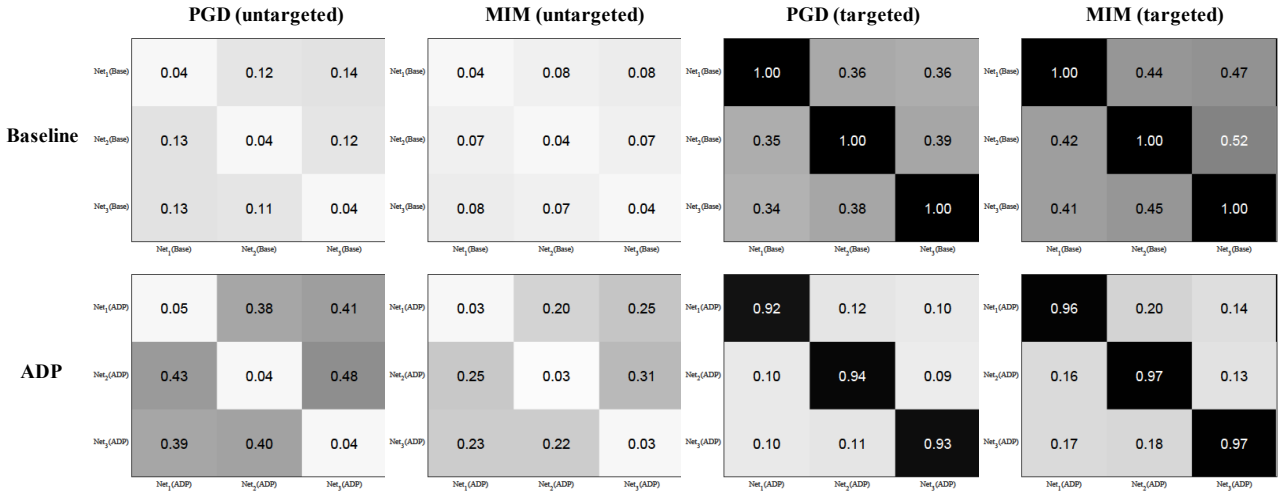
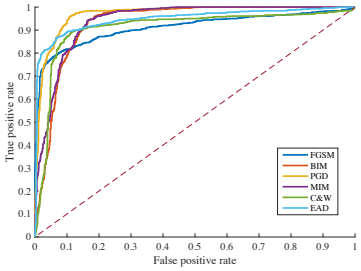


Figure 3. Adversarial transferability among individual models on CIFAR-10. For untargeted attacks, the values are the classification accuracy; For targeted attacks, those are the success rate of adversarial examples fooling classifiers to predict target classes.



Attacks	Para.	AUC-scores (10^{-2})
FGSM	$\epsilon = 0.1$	91.19
BIM	$\epsilon = 0.1$	93.14
PGD	$\epsilon = 0.1$	97.03
MIM	$\epsilon = 0.1$	94.09
C&W	$c = 1.0$	90.98
EAD	$c = 20.0$	94.84

Figure 4. The ROC-curves on 1,000 test examples and 1,000 adversarial examples of CIFAR-10. The model is trained by $\text{ADP}_{2,0.5}$.

Attacks	Para.	CIFAR-10 Baseline	CIFAR-10 $\text{ADP}_{2,0.5}$
Normal	-	93.6	**93.8**
FGSM	$\epsilon = 0.02$	42.0	**58.4**
BIM	$\epsilon = 0.01$	31.6	**41.8**
PGD	$\epsilon = 0.01$	37.4	**44.2**
MIM	$\epsilon = 0.01$	37.1	**47.5**
C&W	$c = 0.01$	52.3	**56.5**
EAD	$c = 1.0$	20.4	**65.3**

Table 5. Classification accuracy (%) on adversarial examples. Ensemble models consist of *five* Resnet-20 such that $5 \nmid 9$.

We apply PGD and MIM as the attack methods, which are the two most commonly used attacks in the black-box setting (Kurakin et al., 2018b). The perturbation parameters for both attacks are set to be $\epsilon = 0.05$. For a complete analysis, we test untargeted mode and targeted mode. For untargeted mode, adversaries only aim to make the models predict wrong labels, and the evaluation criterion is the classification accuracy of the models on adversarial examples. For targeted mode, the evaluation criterion becomes the success rate of fooling the models to predict specific target label. As the results indicate, the ADP training can significantly alleviate the transferability among the members of the ensembles in both untargeted and targeted modes, which can further lead to better robustness for the ensemble models.

4.5. Ensemble Diversity as the Detection Metric

When the adversarial perturbation becomes larger, neither existing defenses nor our method can avoid misclassifying on white-box adversarial examples (Athalye et al., 2018). Then another possible solution is detecting and filtering out adversarial examples instead (Pang et al., 2018a). In Fig. 4, we show the ROC-curves and corresponding AUC-scores when applying ensemble diversity as the detection metric to filter out adversarial examples. The models are trained by $\text{ADP}_{2,0.5}$. Here the parameters used for all attacks are much larger than those in Table 2, which can make trained models misclassify with nearly 100% rate. The results show the effectiveness of ensemble diversity as the detection metric on ADP-trained models, which generalizes our method to a more extensible defense in the adversarial setting.

4.6. The Cases of $K \nmid (L-1)$

To further investigate the flexibility of our method, we consider the cases of $K \nmid (L-1)$. We test on CIFAR-10 with $L = 10$ and construct an ensemble consisting of five individual members, i.e., $K = 5$. It is obvious that $5 \nmid 9$. The results are demonstrated in Table 5. We can find that even if the condition $K \mid (L-1)$ in Corollary 1 does not hold, the ADP training can still largely outperform the baseline method, which suggests that our method is practically insensitive to the relationship of K and L, even if L is not large, and thus can be applied in more general cases.

5. Conclusion

In this paper, we propose the ADP training method. Compared to previous efforts that focus on enhancing a single network, we provide an orthogonal approach to further improve a practical ensemble defense, which implies that ensemble diversity could be an important consideration to promote when designing robust ensemble systems.

Acknowledgements

The authors would like to thank Chongxuan Li for helpful comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFA0700904), NSFC Projects (Nos. 61620106010, 61621136008, 61571261), Beijing NSF Project (No. L172037), DITD Program JCKY2017204B064, Tiangong Institute for Intelligent Computing, NVIDIA NVAIL Program, and the projects from Siemens and Intel.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *International Conference on Machine Learning (ICML)*, 2018.

Bernstein, D. S. *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*, volume 41. Princeton university press Princeton, 2005.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. *ACM Workshop on Artificial Intelligence and Security*, 2017a.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pp. 39–57. IEEE, 2017b.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Ead: elastic-net attacks to deep neural networks via adversarial examples. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2933–2941, 2014.

Dietterich, T. G. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pp. 1–15. Springer, 2000.

Dong, Y., Liao, F., Pang, T., Su, H., Hu, X., Li, J., and Zhu, J. Boosting adversarial attacks with momentum. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*, 2015.

Graves, A. and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pp. 1764–1772, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *International Conference on Machine Learning (ICML)*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Islam, M. M., Yao, X., and Murase, K. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, 2003.

Kannan, H., Kurakin, A., and Goodfellow, I. Adversarial logit pairing. *Advances in Neural Information processing Systems (NeurIPS)*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, 2009.

Krogh, A. and Vedelsby, J. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 231–238, 1995.

Kulesza, A., Taskar, B., et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Kuncheva, L. I. and Whitaker, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *International Conference on Learning Representations (ICLR) Workshop*, 2017a.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *International Conference on Learning Representations (ICLR)*, 2017b.

Kurakin, A., Boneh, D., Tramr, F., Goodfellow, I., Papernot, N., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *International Conference on Learning Representations (ICLR)*, 2018a.

Kurakin, A., Goodfellow, I., Bengio, S., Dong, Y., Liao, F., Liang, M., Pang, T., Zhu, J., Hu, X., Xie, C., et al. Adversarial attacks and defences competition. *arXiv preprint arXiv:1804.00097*, 2018b.

Kwok, J. T. and Adams, R. P. Priors for diversity in generative latent variable models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2996–3004, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. E. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representations (ICLR)*, 2016.

Liao, F., Liang, M., Dong, Y., Pang, T., Zhu, J., and Hu, X. Defense against adversarial attacks using high-level representation guided denoiser. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Liu, Y. and Yao, X. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999a.

Liu, Y. and Yao, X. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):716–725, 1999b.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov): 2579–2605, 2008.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations (ICLR)*, 2018.

Mariet, Z. and Sra, S. Diversity networks. *International Conference on Learning Representations (ICLR)*, 2016.

Pang, T., Du, C., Dong, Y., and Zhu, J. Towards robust detection of adversarial examples. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018a.

Pang, T., Du, C., and Zhu, J. Max-mahalanobis linear discriminant analysis networks. *International Conference on Machine Learning (ICML)*, 2018b.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016a.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016b.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Tsymbal, A., Pechenizkiy, M., and Cunningham, P. Diversity in search strategies for ensemble feature selection. *Information fusion*, 6(1):83–98, 2005.