DENL: Diverse Ensemble and Noisy Logits for Improved Robustness of Neural Networks

Mina Yazdani MINA.YAZDANI@TORONTOMU.CA
Hamed Karimi HAMED.KARIMI@TORONTOMU.CA
Reza Samavi SAMAVI@TORONTOMU.CA
Department of Electrical, Computer, and Biomedical Engineering,
Toronto Metropolitan University, Toronto, ON, Canada
Vector Institute, Toronto, ON, Canada

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Neural Networks (NN) are increasingly used for image classification in medical, transportation, and security devices. However, recent studies have revealed neural networks' vulnerability against adversarial examples generated by adding small perturbations to images. These malicious samples are imperceptible by human eyes, but can give rise to misclassification by NN models. Defensive distillation is a defence mechanism in which the NN's output probabilities are scaled to a user-defined range and used as labels to train a new model less sensitive to input perturbations. Despite initial success, defensive distillation was defeated by state-of-the-art attacks. A proposed countermeasure was to add noise in the inference time to hamper the adversarial attack which also decreased the model accuracy. In this paper, we address this limitation by proposing a two-phase training methodology to defend against adversarial attacks. In the first phase, we train architecturally diversified models individually using the cross-entropy loss function. In the second phase, we train the ensemble using a diversity-promoting loss function. We also propose a robustness certification procedure for our method. Our experimental results show that our training methodology and noise addition in the inference time improved our ensemble's resistance against adversarial attacks, while maintaining reasonable accuracy, compared to the state-of-the-art methods.

Keywords: Deep Learning, Diversity, Ensemble Learning, Robustness, Adversarial Attacks, Trustworthy Machine Learning

1. Introduction

Recent advances in Neural Networks (NN) result in the widespread use of NN classifiers in daily human life applications such as natural language processing (Hinton et al., 2012) and image recognition (Zoph et al., 2018). However, NNs are vulnerable to adversarial attacks, which are intentionally crafted perturbations on images. The small but hardly detectable perturbations can impair the functionality of a model (Szegedy et al., 2014). For example, NNs embedded in image recognition systems of autonomous vehicles may misinterpret a stop sign as a yield sign or speed limit (Eykholt et al., 2018). Such malfunctions can cause serious safety problems. Therefore, machine learning researchers are motivated to seek a way to increase the robustness of DNN systems against adversarial attacks.

Robustness is one of the most critical properties of NN models that ensures the model's ability to handle distortions and variations in inputs and produce accurate results. An NN model is considered to be robust to adversarial examples when its output remains unaffected by slight modifications to its input (Goodfellow et al., 2014). The practicality of possible adversarial examples is shown by feeding adversarial images from a cell phone camera to an ImageNet inception classifier (Kurakin et al., 2018). By measuring the classification accuracy, it is shown that most adversarial examples are incorrectly classified in this real-world application.

Defence methods against adversarial perturbations have been proposed in the literature. The closest work to this proposal are (Liang and Samavi, 2023, 2021), where noises are added to the layers of a NN to improve the robustness of the model against Carlini and Wagner (C&W) attack (Carlini and Wagner, 2017). We refer to the work of (Liang and Samavi, 2023) as the state-of-the-art (SOTA) method throughout this paper. Since C&W attack is dependent on the logits of the neural network, the genuine logits are obscured by adding noise to the inputs and making the outputs of all the layers of the NN noisy. However, the noise addition method decreased the accuracy of the models. The SOTA method used an ensemble of networks to increase the accuracy. The ensemble method refers to the technique of combining multiple models to achieve better predictive performance than using an individual model. The authors were able to improve accuracy by using an ensemble where all networks shared the same architecture (similar ensemble). However, using an ensemble of networks brings about the concept of ensemble diversity. The impact of diversified ensembles on the robustness of NNs when the logits are noisy is an unknown phenomenon. Although (Li et al., 2021) proposed a diversity-promoting loss function to investigate the impact of diversity on the ensemble accuracy, the impact of diverse ensemble on the robustness has not been investigated.

The objective of this research is to investigate the compound impact of noisy logits and diversified ensembles on the robustness and accuracy of NNs. Our method is implemented using a two-phase training process. Phase 1 involves individual training of the models of the ensemble using the cross-entropy loss function. The architectures of the models are diversified by adding layers to the network or increasing the filter size (Strauss et al., 2017). Inspired by (Li et al., 2021), in Phase 2, we train the ensemble utilizing a loss function that encourages ensemble diversity. Similar to the SOTA method, to hinder the adversarial attack, we add noise to the inputs during inference time. We evaluate our approach by running experiments to compare our results with the state-of-the-art methods. The results of our experiments show that our approach improves the robustness of the ensemble against adversarial examples in comparison with the state-of-the-art defence method of noisy logits while maintaining accuracy. In this paper, we are making three contributions: (1) we develop a methodology to protect models against adversarial examples using diversified ensemble and noisy logits (Sections 3.1, 3.2, and 3.4), (2) propose an approach based on training the ensemble with a diversity-promoting loss function to evaluate the effect of increased ensemble diversity on the robustness of our models (Section 3.3), and (3) we define robustness guarantee and a certification procedure for our approach (Section 3.5). Our experimental evaluation confirms the effectiveness of our methodology (Section 4).

2. Related Work

The vulnerability of Neural Networks (NNs) to adversarial examples is attributed to model structures and learning techniques (Ren et al., 2020). Transferability is a key feature of adversarial examples, allowing attacks generated for one model to fool others, even with diverse architectures or training data (Papernot et al., 2016b). Various methods have been developed, including L-BFGS (Szegedy et al., 2014), Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), Jacobian-based Saliency Map Attack (JSMA) (Papernot et al., 2016a), Basic Iterative Method (BIM) (Kurakin et al., 2018), and Carlini and Wagner (C&W) attacks (Carlini and Wagner, 2017) to use the loss function that is dependent to the output of the second last layer (i.e. the logits). These methods aim to find minimal perturbations to deceive the model's predictions using techniques such as gradient updates, saliency maps, iterative noise application, and alternative loss functions based on logits.

Research into countermeasures against adversarial attacks has led to the development of defense mechanisms to mitigate their effects (Goodfellow et al., 2014; Kurakin et al., 2018). Adversarial training incorporates adversarial examples into the training data to increase a model's resistance (Szegedy et al., 2014; Goodfellow et al., 2014). (Athalye et al., 2018) presented an algorithm to synthesize adversarial examples (3D objects) over a specific distribution of transformations which are robust to noise, distortion, and affine transformation. Although adversarial training techniques have shown success in defending NNs against known adversarial examples, they do not provide a formal guarantee of the model's robustness (Raghunathan et al., 2018; Bastani et al., 2016). Certified robustness provides a formal guarantee of a model's ability to withstand adversarial attacks (Carlini and Wagner, 2017). As another defense method, Defensive Distillation smooths output probabilities using temperature scaling to decrease the effectiveness of attack algorithms (Papernot and McDaniel, 2016; Papernot et al., 2016b). However, it was later found that the C&W attack can defeat defensive distillation in black-box settings by targeting the logits of the network.

Ensemble methods enhance neural network accuracy and generalizability by combining multiple models (Li et al., 2021). Diversity within the ensemble compensates for one classifier's failures with others' successes, leading to increased robustness (Sharkey and Sharkey, 1997; Liu et al., 2019). Weighted strategies can also emphasize classifiers with better performance (Breiman, 1996). To overcome C&W attacks, Liang and Samavi (Liang and Samavi, 2023, 2021) (SOTA) impaired the functionality of the C&W attack algorithm by adding random noise to the logits and preserving model accuracy using an ensemble of models and a majority vote technique. They used an ensemble of architecturally similar models to maintain accuracy as a solution to the adverse effect of noise on the accuracy of the models. However, since diversifying the architectures of the ensemble models can enhance their resistance to adversarial examples (Strauss et al., 2017), we were inspired to diversify the architectures of the networks used by the SOTA method to improve the clean accuracy and robustness of their ensemble against adversarial attacks. Training an ensemble is a computationally expensive task. However, the diversity in an ensemble where the models' architectures are only slightly different does not impose significant additional expenses, but it results in the benefit of increasing both accuracy and robustness. Other approaches, such as "Specialists+1" ensembles (Abbasi and Gagné, 2017), ATHENA (Meng et al., 2020), Xensemble (Wei and Liu, 2020), gradient dispersion in ensemble models (Huang

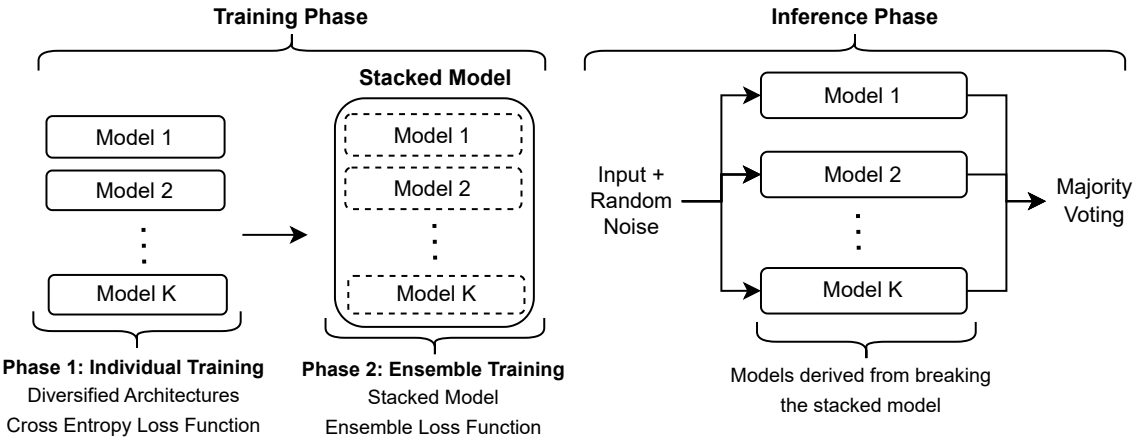


Figure 1: Model training and inference phases of DENL method

et al., 2021), and diversifying ensemble architectures (Strauss et al., 2017) have also been proposed to enhance ensemble performance and diversity for improved robustness against adversarial attacks.

3. Diverse Ensembles and Noisy Logits (DENL)

We are inspired by the general effect of ensemble diversity in increasing the accuracy of the NNs. We investigate using diversified ensembles to improve the robustness of NNs to adversarial examples, where the model is under C&W attack. To be more specific, we apply the combination of two methods to improve defensive distillation: 1) adding diversity to the ensemble in training time, and 2) adding random noise to the inputs during inference time. As model training and inference phases are shown in Figure 1, adding diversity to the ensemble is achieved through a two-phased training process. In Phase 1, we diversify the architectures of the models in the ensemble, and in Phase 2, we utilize a diversity-promoting loss function to improve the diversity among models of the ensemble. Then, a random noise is added to the inputs in inference time.

3.1. Diversity in Ensembles

In this paper, we quantify ensemble diversity using the correlation between the outputs of the models of the ensemble. This definition is inspired by the diversity measurement index R_{LL} proposed by (Li et al., 2021), which is the average correlation between the outputs of the pairs of models.

Our methodology is based on increasing the diversity among models. The training of our ensemble method is divided into two phases to add diversity to the ensemble. In the first phase, we diversify the architectures of the models in the ensemble and individually train the models of the ensemble while defensive distillation is applied. Therefore, by adding a temperature parameter to the softmax function, the output probabilities are scaled. The ensemble diversity initiates from the various architectures used for models in the ensemble. In the second phase, we utilize a diversity-promoting loss function to improve the diversity among models of the ensemble. During the ensemble training phase, the parameters of the

models are updated to minimize the loss function, which focuses on improving the ensemble diversity. An overview of model training in Phase 1 and Phase 2 is shown in Figure 1 which are described in Sections 3.2 and 3.3, respectively.

3.2. Injecting Diversity in Network Architectures

As shown in Figure 1, we add diversity to our ensemble by varying the architecture of the networks and call it Phase 1. The architecture of the first model in the ensemble is regarded as the base architecture. Architectures of the other models in the ensemble have slight variations from the basic architecture. The variations in the architectures are achieved by changing the number of layers or the size of convolutional filters. The underlying defence method that we are aiming to improve is defensive distillation (Papernot et al., 2016b) described in Section 2 to soften the output probabilities. In defensive distillation, the softmax function of the model is replaced with a version in which the logits are divided by a constant temperature T. Using $T > 1$, we smooth out the output probability distribution and reduce the confidence of the model's predictions. By increasing the parameter T, the probability distribution becomes smoother over the classes.

In creating adversarial examples, the attackers often choose pixels with a high gradient, because the gradient of the loss function with respect to the input image provides information about how much each pixel contributes to the output of the model. By manipulating the pixels with high gradients, an attacker has a better chance to cause misclassification. When the temperature parameter is applied to the softmax function, the gradients that are computed during backpropagation are affected since the softmax probabilities become more evenly distributed in which the value of the gradients decreases. In Phase 1, we train each individual model using cross-entropy loss function. We utilize the temperature-scaled softmax function to calculate the probabilities. In the next phase, we implement an ensemble learning phase to improve ensemble diversity using the diversity-promoting loss function.

3.3. Diversity-Promoting Ensemble Training

In this step which is called Phase 2, we perform a procedure similar to Phase 1, but with three important differences: (1) the initial values of the parameters for Phase 2 are the values calculated at the end of Phase 1, (2) in Phase 2, all the models in the ensemble share the same loss function. This is in contrast with Phase 1, where the cross-entropy loss function of each model is calculated independently, and (3) the outcomes of all the models in the ensemble are involved in calculating the shared loss function, as described below.

In Phase 2 (Figure 1), we construct a stacked model from our models in the ensemble to apply the diversity-promoting loss function proposed in (Li et al., 2021). The inputs are fed into each of the K models $F^1, \cdots, F^K$ with parameters of $\Theta^1, \cdots, \Theta^K$ and the outputs of each model are calculated. The loss function used in Phase 2 is defined as,

$$\mathcal{L}_{div} = -(R_{TL} - \lambda.R_{LL}) \ , \tag{1}$$

in which R_{TL} is the average correlation between models' outputs and ground-truth labels and defined as,

$$R_{TL} = \frac{1}{K}\sum_{k=1}^{K} r_{T,L_k} \quad \text{s.t.} \quad r_{T,L_k} := \frac{1}{M}\sum_{j=1}^{M} \text{corr}(\boldsymbol{y}_{:j}, \hat{\boldsymbol{y}}^k_{:j}) \ , \tag{2}$$

where K is the number of models in the ensemble and r_{T,L_k} is the average of the Pearson correlation coefficient between ground-truth labels and the predictions made by the k^{th} model in the ensemble denoted by $\text{corr}(\cdot,\cdot)$ defined in Eq. 3. Moreover, $\boldsymbol{y}_{:j}$ and $\hat{\boldsymbol{y}}^k_{:j}$ are the vectors associated with the class label j over all data samples in the matrices Y and $\hat{Y}^k$, respectively. For two arbitrary vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, the Pearson correlation coefficient is defined as,

$$\text{corr}(\boldsymbol{v},\boldsymbol{w}) = \frac{\sum_i (v_i - \bar{v}) \cdot (w_i - \bar{w})}{\sqrt{\sum_i (v_i - \bar{v})^2 \cdot \sum_i (w_i - \bar{w})^2}} \ , \tag{3}$$

where $v_i \in \boldsymbol{v}$, $w_i \in \boldsymbol{w}$, and $\bar{v}$ and $\bar{w}$ are the mean of the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$.

Assume that $Y = [y]_{N\times M}$ denotes a $N \times M$ matrix of M one-hot encoded ground-truth labels for N data samples, where y_{ij} is equal to one, if sample i belongs to class j and is equal to zero otherwise. Moreover, assume $\hat{Y}^k = [\hat{y}^k]_{N\times M}$ is a $N \times M$ matrix of class probabilities produced by the k^{th} model in the ensemble using N data samples in which $\hat{y}^k_{ij}$ is the prediction probability corresponding to the j^{th} class of the i^{th} sample. In Eq. 1, R_{LL} is the average correlation between the outputs of the pairs of models (i.e., inter-member correlation). In other words, R_{LL} measures the correlation between the predictions made by different pairs of models selected from the ensemble and is defined as,

$$R_{LL} = \frac{2}{K(K-1)} \sum_{k=1}^{K} \sum_{k'=k+1}^{K} r_{L_k,L_{k'}} \quad \text{s.t.} \quad r_{L_k,L_{k'}} = \frac{1}{M} \sum_{j=1}^{M} \text{corr}(\hat{\boldsymbol{y}}^k_{:j}, \hat{\boldsymbol{y}}^{k'}_{:j}) \ , \tag{4}$$

where $r_{L_k,L_{k'}}$ denotes the average of Pearson correlation coefficient between the predictions made by models k and k'. Note that R_{LL} and R_{TL} are capturing the ensemble diversity and the accuracy of the models, respectively. The subscripts LL and TL denote the computed correlation among pairs of learners (models) and the computed correlation among learners and true labels, respectively. The higher the value of the R_{LL} metric is, the more correlated the ensemble models are in terms of their predictions. An ensemble with a smaller value of R_{LL} can be regarded as being a more diverse ensemble.

In Eq. 1, the parameter λ controls the balance between the two components. Since the output of all the models is involved in the calculation of R_{LL} and R_{TL}, the loss function in each epoch is affected by the outputs of all the models. We call this newly developed model a *stacked model*, since the models are connected via a shared loss function. After calculating the shared loss function, in the backpropagation, the gradient of the loss function for each submodel F^k is calculated with respect to the parameters of k^{th} model Θ^k. It means the backpropagation process happens in each model individually. The gradients of the loss function $\mathcal{L}$, i.e., $\nabla(\lambda \cdot R_{LL} - R_{TL})$ with respect to Θ^k are used to update the values of Θ^k in each model. This is done by subtracting $\Theta^k - \alpha_2 \cdot \nabla(\lambda \cdot R_{LL} - R_{TL})$, where α_2 is the learning rate of Phase 2. The Phase 2 procedure is summarized in Algorithm 1. The inputs of the algorithm are K models as $F^1, \cdots, F^K$ with the updated parameters in Phase 1 as $\Theta^1, \cdots, \Theta^K$, N labelled samples $(x_1, y_1), \cdots, (x_N, y_N)$, number of classes M, learning rate α_2, and the trade-off parameter λ. Algorithm 1 shows the ensemble training for each epoch. Lines 5 and 8 show the process of calculating the loss functions based on two components of inter-member correlation (R_{LL}) and correlation between each model and ground truth (R_{TL}). Line 11 shows how the parameters of all the models are updated using the gradient

Algorithm 1: Phase 2: Ensemble Training

Data: A set of N data samples as X; A matrix of M one-hot encoded classes per data sample as $[y]_{N\times M}$; Models F^k with parameters Θ^k such that $k \in \{1, ..., K\}$ updated in Phase 1; Phase 2 learning rate α_2; Phase 2 trade-off parameter λ;

Result: A set of trained models F^k;

$R_{TL}, R_{LL} \leftarrow 0$

for *each class* j **do**

  for *each model* k **do**

    $\hat{Y}^k \leftarrow F^k(X; \Theta^k)$ `// $\hat{Y}^k$ is a $[N \times M]$ matrix of output probabilities`

    $R_{TL} \leftarrow R_{TL} + \frac{1}{MK} \operatorname{corr}\left(\boldsymbol{y}_{:j}, \hat{\boldsymbol{y}}^k_{:j}\right)$ `// $\hat{\boldsymbol{y}}^k_{:j}$ is the $j$th vector of $\hat{Y}^k$`

    for $k^{'} := k + 1$ *to* K **do**

      $\hat{Y}^{k'} \leftarrow F^{k'}(X; \Theta^{k'})$

      $R_{LL} \leftarrow R_{LL} + \frac{2}{MK(K-1)} \operatorname{corr}\left(\hat{\boldsymbol{y}}^k_{:j}, \hat{\boldsymbol{y}}^{k'}_{:j}\right)$

    end

    for *each* $\theta \in \Theta^k$ **do**

      $\theta \leftarrow \theta - \frac{\alpha_2}{n} \nabla(\lambda . R_{LL} - R_{TL})$ `// Update the parameters by gradients`

    end

  end

end

return $\{F^k\}$

of the shared loss function $\mathcal{L}$. The output of the algorithm are K models $F^1, \cdots, F^K$ with the updated parameters after each epoch.

3.4. Noisy Logits

Carlini and Wagner were able to break the defensive distillation technique by accessing the logits of the network, as defensive distillation was only hiding the softmax probabilities. To prevent C&W attack to succeed, similar to the SOTA method, we impair the performance of C&W attack by obfuscating the precise logits. Furthermore, the output of all the layers should be noisy, so the attacker cannot recover the logits by making queries of previous layers.

In the inference time of our method, random noise is added to each input before being fed to our ensemble. To achieve noisy outputs for all the layers, we add random Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ to the inputs of the network. Thus, for each input $x \in X$ after noise addition $x' = x + \varepsilon$ and we have,

$$x'_l = F_l \circ F_{l-1} \circ ... \circ F_1(x') \ . \tag{5}$$

Since the added noise is considered in the calculation of the outputs of all layers, the attacker's access to the genuine values of logits of the network is hindered. We add zero-mean Gaussian noise to our inputs, which has proved to be effective against C&W l_2 attack (SOTA). We use ensemble method as a solution to improve the accuracy, which was decreased because of the noise. After calculating the output of each individual model, the final output of the ensemble is calculated by majority voting among models.

3.5. Ensemble Voting and Robustness Guarantee

We follow a similar approach as the SOTA method and (Cohen et al., 2019) to aggregate the results of the ensemble and provide robustness guarantee for our approach. It is important to understand how much perturbation the model is able to withstand without changing its prediction, as it gives us confidence about the model's *robustness* to make predictions under adversarial attacks. We say a classifier F is robust to perturbations of a certain size if its original prediction on an input does not change after perturbations. Formally, let $F : D \to \mathcal{Y}$ and $x \in D$ be any input. F is robust at x up to perturbations of size R (under some norm function $||\cdot||$), if for any $\delta \in \mathbb{R}^m$ (m is the dimension of input x) such that $||\delta|| \leq R$ and $x + \delta \in D$, we have $F(x + \delta) = F(x)$.

Given an ensemble F consisting of K models $S = \{F^k : k = 1, 2, ..., K\}$, $p(S)$ is defined as the set of all partitions of the set S. For each partition h of the set $p(S)$, let L_h define the largest subset in h and $P(L_h)$ be the probability that all models in L_h successfully classify a data point. Then the probability of success by majority voting is $\sum_{h \in p(S)} P(L_h)$. It should be noted that when K is large, there are many alternative partitions, which increases the likelihood that the majority vote would succeed. We use the majority vote consensus method to aggregate the outputs of all the networks for the *winner* class.

To provide robustness guarantee, compared to the method in (Cohen et al., 2019), since we are using an ensemble with voting instead of a single network, we report the winner class along with the runner up. Let n_A and n_B correspond to the vote counts of the top two classes y_A and y_B, respectively, predicted by the ensemble for some given input x, where $n_A \geq n_B$. Let F^* denote the ensemble classifier where its output is decided from the most frequent output among K models as,

$$F^*(x) = \arg\max_{y \in \mathcal{Y}} \sum_{k=1}^{K} \mathbb{1}_{F^k(x)=y} \ . \tag{6}$$

After adding noise $\varepsilon \sim N(0, \sigma^2 I)$, we define $g(x) = F^*(x + \varepsilon)$. If, we follow Cohen et al. method and substitute their base classifier f, with our ensemble F^* and their smoother classifier g, with our noisy ensemble, we can adapt the same proof for the following theorem.

Theorem 1 (l_2-norm): If there exists $y_A \in \mathcal{Y}$ and $\underline{p_A}, \overline{p_B} \in [0, 1]$ such that

$$P(F^*(x + \varepsilon) = y_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y \neq y_A} P(F^*(x + \varepsilon) = y) \ , \tag{7}$$

then $g(x + \delta) = y_A$ for all $||\delta||_2 \leq R$, where $R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))$, $\underline{p_A}$ is the lower bound of p_A (the probability that the top class y_A is returned), $\overline{p_B}$ is the upper bound of p_B (the probability that the second top class y_B is returned), and Φ^{-1} is the inverse of the Gaussian CDF.

Note that the distribution of $F^*(x + \varepsilon)$ can be approximated using Monte Carlo simulations, so we can estimate $\underline{p_A}$ at a specified level of α. Then, the certification procedure follows the algorithm provided in the SOTA method (Liang and Samavi, 2023).

4. Experimental Evaluation

To investigate the impact of using a diversified ensemble and noisy logits on the robustness against adversarial examples, we implement our method on two different ensembles. First,

we implement our method on a similar-architecture ensemble (SAE). By comparing the results of SAE with a diverse-architecture ensemble (DAE), we aim to observe the impact of diversity on the robustness of the ensemble. Since our proposed method is implemented in two phases, we evaluate our ensembles (SAE and DAE) after each phase.

Test Environment. We implemented[1] and ran our experiments on Python 3.8 and TensorFlow v.2.8.0 using a machine with AMD EPYC 7402P (24 cores) CPU, a Tesla T4 GPU, and 220GB RAM.

Datasets. We evaluate our approach on two real-world datasets, CIFAR10 and MNIST. These two datasets are widely used in the literature as benchmarks. CIFAR10 consists of 60,000 colour images, with 32×32 pixels from 10 object class labels. MNIST is a dataset of 70,000 gray-scale images of digits (0-9) with 28×28 pixels.

Training Setup. The architectures used in SAE for MNIST and CIFAR10 have 32 and 64 kernels of size 3×3 in convolutional layers, max pooling of size 2×2, and 200 and 256 hidden neurons in fully-connected layers, respectively. This architecture is consistent with the architecture applied in the SOTA method for a fair experimental comparison. For DAE, we use 10 different models: the first model of the ensemble inherits the same architecture as the architecture used in SAE. The other models have different permutations in the number of convolutional kernels and hidden neurons in the architecture, specifically, additional convolution layers and extra filters are added to the models. To perform cross-validation evaluations for both datasets, we divide the training set into ten equal-sized subsets which in each fold, we dedicate one subset (5,000 images) for validation.

4.1. Evaluation Procedure

In addition to the model accuracy to assess our method, we also use adversarial accuracy as the proportion of the adversarial examples that the ensemble classifies correctly to the number of total generated adversarial examples. We evaluate ensemble diversity and individual model performance using the metrics R_{LL} and R_{TL}, respectively. In the following, we provide the evaluation procedures in both training phases.

Phase 1. To implement defensive distillation, we use the temperature-scaled version of the softmax function. The set of temperatures used to train the models is T={10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, for both SAE and DAE and consistent with (Papernot et al., 2016b)). Note that using different temperatures in the models of the ensemble helps to improve the ensemble diversity.

In order to be consistent with the SOTA method, in Phase 1, we train our models using SGD optimization algorithm with learning rate of 0.01, dropout rate of 0.5, and batch size of 128 for 150 and 250 epochs over CIFAR10 and MNIST datasets, respectively. We fix the number of models in the ensemble to ten and each model wass trained individually. For CIFAR10, we train the models on the entire training set (45,000 images). Each model was randomly initialized and trained for 150 epochs with a different temperature value. For MNIST, in Phase 1, we equally divide the training data (55,000 images) among the ten models of the ensemble. Each model was trained with random initialization of the parameters for 250 epochs with a different temperature value. When the temperature of the softmax function increases, the models need more training epochs to converge. The loss

1. Our source codes are available at: https://github.com/samavi/pubs/tree/main/DENL

function used in Phase 1 is cross-entropy, which solely stresses the accuracy of the ensemble and does not take into account ensemble diversity.

Phase 2. In this phase, we generate a stacked ensemble model and train it over the same training data as Phase 1. Note that in this phase, the MNIST training dataset, was not partitioned. We train the ensembles as a whole and update the weights and biases of the models simultaneously using the loss function described in Eq. 1. We use SGD optimization algorithm with learning rate of 0.1 and batch size of 512 for 120 and 20 epochs over CIFAR10 and MNIST datasets, respectively.

We use $\lambda = \{0.1, 0.2, ..., 1\}$, which was proposed by (Li et al., 2021). Our fine-tuning (reported in Appendix A) shows that $\lambda = 0.4$ produced the ensemble with the highest robustness for both datasets. We searched for the λ that yields an ensemble with higher robustness in terms of accuracy on adversarial examples. The number of epochs needed for this phase was 120 and 20 for CIFAR10 and MNIST, respectively. We continue ensemble training until R_{LL} reached its minima and started increasing. A set of experiments were implemented on validation data to tune the λ value and the number of epochs.

4.2. Generating Adversarial Examples

We generate adversarial examples after the implementation of Phase 1 and 2, to evaluate the effect of each phase on the resistance of our ensemble against adversarial examples. Adversarial examples are generated by two adversarial attack scenarios. The first is a single attack scenario, where we aim to observe the transferability feature of adversarial examples. Transferability can be interpreted as the likelihood that adversarial examples generated targeting one model is likely to fool other models of the ensemble (Papernot et al., 2016b). However, by increasing the diversity in an ensemble, we expect the risk of transferability decreases.

The second attack is superimposition attack, which is a perturbation aggregation type of attack. We aggregate the perturbations designed for examples targeting two models in the ensemble. Due to the combinatorial effect of possible subsets of superimposition attacks, in this experiment, we limit superimposition attacks to be consisting of two simultaneous attacks targeting two of the models in the ensemble. In this attack, we generate adversarial examples when each of the models in the ensemble is targeted. We then choose two adversarial examples with the two smallest perturbations and simultaneously impose both perturbations to create adversarial examples.

In these experiments, for consistent and fair comparison with the SOTA method, we use the l_2 version of the C&W algorithms. To generate each adversarial example, we select a model from an ensemble (SAE and DAE), an image from our test datasets, and a target label that is different from the ground-truth label of the selected image. Formally, for a sample input x and a target label t, for each F^k we generate an adversarial example $A(F^k, x, t)$ using the C&W attack[2]. The confidence, max iteration, and initial constant parameters of the attacks are 0, 500 and 0.01, respectively for MNIST data set and 0, 50, 0.01 for the CIFAR10 dataset.

To craft the adversarial examples, we generate a set of images from the test set that were correctly classified by the ensemble before the attack. For example, we use 128 images from

2. The source code is available at: https://github.com/carlini/

Table 1: The comparison of accuracy results for similar-architecture and diverse-architecture (including the proposed DENL method) ensembles using single and superimposition attacks on the CIFAR10 dataset over 5 folds in Phase 1 training

	Ensemble Clean Acc. (%)	**Avg. of Individuals Accuracy (%)**	**Adversarial Acc. Single Attack (%)**	**Adversarial Acc. Superimposition Attack (%)**
Individual Model	-	77.61 ± 0.09	0.00 ± 0.00	-
SAE	80.75 ± 0.20	75.46 ± 0.05	94.48 ± 0.05	91.81 ± 0.17
SAE + Noisy Logits	79.49 ± 0.11	73.86 ± 0.14	96.06 ± 0.24	94.65 ± 0.21
DAE	82.98 ± 0.13	76.15 ± 0.18	95.11 ± 0.23	95.34 ± 0.19
DAE + Noisy Logits	81.31 ± 0.17	74.08 ± 0.09	96.32 ± 0.26	96.47 ± 0.34

Table 2: The comparison of accuracy results for similar-architecture and diverse-architecture (including the proposed DENL method) ensembles using single and superimposition attacks on the MNIST dataset over 5 folds in Phase 1 training

	Ensemble Clean Acc. (%)	**Avg. of Individuals Accuracy (%)**	**Adversarial Acc. Single Attack (%)**	**Adversarial Acc. Superimposition Attack (%)**
Individual Model	-	99.45 ± 0.02	0.00 ± 0.00	-
SAE	99.27 ± 0.06	97.52 ± 0.04	89.40 ± 0.34	31.82 ± 0.14
SAE + Noisy Logits	98.31 ± 0.29	91.36 ± 0.04	80.59 ± 0.12	82.07 ± 0.02
DAE	98.76 ± 0.06	97.54 ± 0.05	90.98 ± 0.07	33.69 ± 1.18
DAE + Noisy Logits	97.89 ± 0.01	90.71 ± 0.10	81.69 ± 0.05	84.02 ± 0.96

the correctly classified test set of our data x_i such that $i = 1, 2, ..., 128$ for the single attack. From each clean image, we generate adversarial examples to attack each of the models in the ensemble and target any class t_j except the correct label t_c where $j \neq c$, i.e., 9 classes out of 10 possible classes to cause misclassification. Thus, for each adversarial example $A(F^k, x_i, t_j)$ on model $F^k, k \in \{1, ..., 10\}$ for target class, the total number of adversarial examples created for the single attack is $128 \times 10 \times 9 = 11520$. For superimposition attacks, we take 256 clean images from the correctly classified test data and attack all the models of the ensemble, but we select the top two smallest perturbations among them. We then simultaneously add those perturbations to the clean image to create the adversarial example. Thus, the total number of superimposition attacks is $256 \times 9 = 2304$.

As we described in Section 3.4, we add noise to the inputs to impair the C&W attack algorithm. In order to be able to evaluate the effect of adding noise to the inputs, we generate adversarial examples with and without noise. For generating random noise, we use a Gaussian noise function, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 0.5$ and 0.03 for MNIST and CIFAR10, respectively (these values provide sufficient noise while minimizing accuracy loss based on the SOTA method). Thus, we use the same values of σ parameters to generate our random noise on MNIST and CIFAR10. In the following, we evaluate SAE and DAE ensembles after each one of Phase 1 and Phase 2 trainings.

4.3. Phase 1 Results

The training results are presented in Tables 1 and 2 for CIFAR10 and MNIST, respectively. The adversarial examples classification and the perturbation details of the experiments are reported in Appendix B, and the required time to run the experiments is reported in Appendix C of the supplementary material.

Clean Accuracy Improvement. As shown in the first column of Table 1, for CIFAR10, the ensemble clean accuracy is around 80.75% for SAE, where the accuracy increases to 82.98% for DAE. When we add noise to the inputs, the ensemble clean accuracy shows a decrease in comparison when we have no noise on inputs.

As shown in Table 2, for MNIST, the ensemble clean accuracy in SAE and DAE are around 99.27% and 98.76%, respectively. In contrast to CIFAR10, the ensemble clean accuracy is slightly decreased after adding architectural diversity to the ensemble. Similar to CIFAR10, when we add noise to the inputs, the ensemble clean accuracy of SAE and DAE decreases.

As shown in the first and second columns of Tables 1 and 2, we can observe that in CIFAR10 and MNIST, the ensemble clean accuracy has increased in comparison with the average of the individuals accuracies for both datasets.

Transferability. As shown in the third column of Table 1, for CIFAR10 the adversarial accuracy (single attack) for an individual model is 0.00%. The adversarial accuracy increases to around 94.48% and 95.11% for SAE and DAE, respectively. When noisy logit is used, the same values for SAE and DAE increase to 96.06% and 96.32%, respectively.

As shown in the third column of Table 2, for MNIST, the adversarial accuracy are around 89.40% and 90.98% for SAE and DAE, respectively. In the MNIST dataset, noisy logits decrease the adversarial accuracy, which means noisy logit could not decrease the transferability of attacks among models trained on MNIST. This is consistent with the results reported in (Liang and Samavi, 2023). However, using DAE improved the resistance of the ensemble against transferability of the adversarial examples.

Superimposition Attack. As shown in the fourth column of Table 1, for CIFAR10 the adversarial accuracy (superimposition attack) is 91.81% and 95.34% for SAE and DAE, respectively. When we add noise, both SAE and DAE show an improvement in adversarial accuracy. This trend is similar in MNIST, as shown in the fourth column of Table 2.

4.4. Phase 2 Results

The results are summarized in Tables 3 and 4 for CIFAR10 and MNIST, respectively. In this phase, we observe the same trends as reported for Phase 1. For example, in the first column of Table 3 for CIFAR10, the accuracy of the ensemble for SAE and DAE are around 80.11% and 82.81%, respectively. Furthermore, if we compare the ensemble accuracy of SAE and DAE with the average of individual accuracy, we observe an increase in accuracy while an ensemble (majority vote) is used. The results of the adversarial examples classification and the perturbation details of the experiments are reported in Appendix B, and the required time to run the experiments is also reported in Appendix C of the supplementary material. We have also the same trend and similar observation for MNIST as shown in Table 4.

4.5. Discussion

By comparing the first and second columns of Tables 1 and 3 for CIFAR10, and Tables 2 and 4 for MNIST, we observe that the ensemble clean accuracy and the average of individual accuracy are slightly different between Phase 1 and Phase 2 training. For example, in CIFAR10, ensemble accuracy of SAE and DAE are 80.75% and 82.98% after Phase 1. The same values recorded after Phase 2, are 80.11% and 82.81%. Comparing the third columns

Table 3: The comparison of accuracy results for similar-architecture and diverse-architecture ensemble (including DENL) using single and superimposition attacks on adversarial CIFAR10 images over 5 folds in Phase 2 ensemble training

	Ensemble Clean Accuracy (%)	**Average of Individuals Accuracy (%)**	**Adversarial Accuracy Single Attack (%)**	**Adversarial Accuracy Superimposition Attack (%)**
SAE	80.11 ± 0.31	75.51 ± 0.24	96.01 ± 0.27	92.97 ± 0.21
SAE + Noisy Logits	79.46 ± 0.23	73.66 ± 0.41	97.21 ± 0.13	96.11 ± 0.05
DAE	82.81 ± 0.33	76.94 ± 0.30	96.82 ± 0.28	96.90 ± 0.74
DAE + Noisy Logits	81.01 ± 0.42	74.24 ± 0.37	98.29 ± 0.61	97.46 ± 0.41

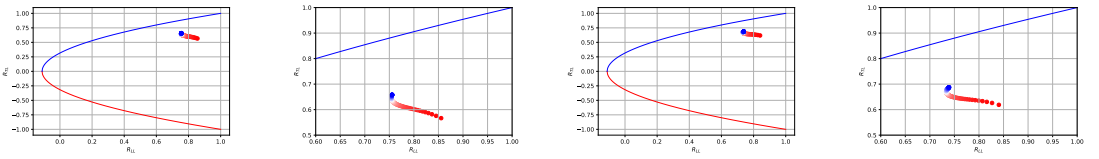
Table 4: The comparison of accuracy results for similar and diverse architecture ensembles (including DENL) using single and superimposition attacks on adversarial MNIST images over 5 folds in Phase 2 ensemble training

	Ensemble Clean Accuracy (%)	**Average of Individuals Accuracy (%)**	**Adversarial Accuracy Single Attack (%)**	**Adversarial Accuracy Superimposition Attack (%)**
SAE	98.90 ± 0.13	97.60 ± 0.35	90.39 ± 0.81	32.55 ± 0.45
SAE + Noisy Logits	97.85 ± 0.14	91.57 ± 0.23	82.61 ± 0.95	83.94 ± 0.17
DAE	98.58 ± 0.39	98.40 ± 0.41	91.50 ± 0.87	34.10 ± 0.91
DAE + Noisy Logits	97.35 ± 0.62	91.08 ± 0.54	83.02 ± 0.94	86.65 ± 1.61

of the same tables shows an increase of the adversarial accuracy in single attack for all cases after Phase 2 training. For example, in CIFAR10, the adversarial accuracy (single attack) of SAE and DAE are 94.48% and 95.11% after Phase 1 training, while the adversarial accuracy increases to 96.01% and 96.82% after Phase 2. In a similar trend, the adversarial accuracy in superimposition attack on CIFAR10, for SAE and DAE, increases to 92.97% and 96.90%, respectively. This trend is similar when we have noisy logits too. For example, after Phase 1, the ensemble accuracy of DAE + noisy logits is 96.32%, which increased to 98.29% for Phase 2 training of CIFAR10.

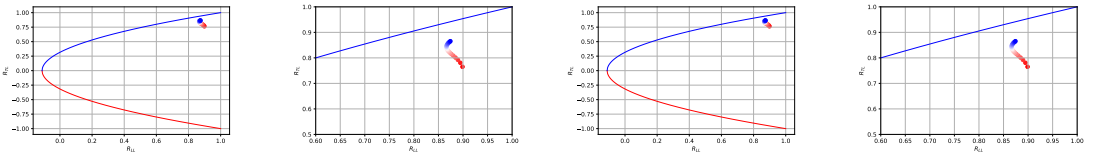
Figure 2 demonstrates how R_{TL} and R_{LL} of SAE (Figures 2(a) and 2(e)) and DAE (Figures 2(c) and 2(g)) ensembles are modified during Phase 2 training for both CIFAR10 and MNIST datasets. R_{TL} and R_{LL} are dependent to each other so that there are two borders to demonstrate the limits of R_{LL} and R_{TL} and how each metric changes with respect to each other. Any ensemble falls within the region bounded by blue (upper bound) and red (lower bound) Li et al. (2021). What is shown in these figures, is consistent with our interpretation of how the diversity-promoting loss function worked during Phase 2 training. The red points show the trend of R_{TL} and R_{LL} in each epoch from the starting point, and the blue point is the final point after the last training epoch. Thus, the ensemble moves in the direction of decreasing R_{LL} and increasing R_{TL}. This means both SAE and DAE increase the diversity in the ensemble and single accuracy of the models. For example in CIFAR10, comparing the robustness of the ensemble shown in Tables 1 and 3 shows that the robustness against adversarial examples generated by single and superimposition attack improved at the end of this trajectory.

By comparing the results of the tables in Section 4.3 and 4.4, we can infer that in CIFAR10, DAEs outperform SAEs in terms of adversarial accuracy against examples generated by single and superimposition attacks. When we add noise to the inputs, a similar trend is observed. Our results demonstrate that adding noise to the inputs in inference time and adding diversity through our two-phase training process, increased the robustness



(*a*) SAE (*b*) SAE (Zoomed) (*c*) DAE (*d*) DAE (Zoomed)

CIFAR10 dataset



(*e*) SAE (*f*) SAE (Zoomed) (*g*) DAE (*h*) DAE (Zoomed)

MNIST dataset

Figure 2: Changes in R_{TL} and R_{LL} during training epochs for both datasets. The ensemble starts from the red point and reaches to the blue point after Phase 2 training

of the ensemble against adversarial examples, while the accuracy is maintained. The best performance is observed when DAE ensemble and noisy logits are both used.

In MNIST, the adversarial accuracy against superimposition attacks shows a similar increasing trend. It means our compound method (DAE + noisy logits) improved the robustness of the ensemble against adversarial examples. In single attack scenario, there is an increase in the robustness when ensemble diversity is improved. However, in a single attack scenario, adding noise decreased the adversarial accuracy of the ensemble. Thus, in contrast to CIFAR10, noise addition causes more adversarial examples to successfully transfer among models of the ensemble. The diversity among members made the ensembles more robust against the transferability of the adversarial examples generated by single and superimposition attacks.

Limitations. Despite of the benefits that an ensemble offers compared to an individual network, the additional computational cost of training and inferring an ensemble is a limiting factor. Our experimental evaluation supports achieving a more robust DNN when the ensemble is diversified, however, a theoretical investigation of this problem will provide a deeper understanding of the effect.

5. Conclusion

In this paper, we proposed a method to improve the robustness of NNs. To defend against adversarial attacks, we trained the architecturally diversified models in two phases: individually using the cross-entropy loss function and the ensemble using a diversity-promoting loss function. In inference time, we added noise to the inputs to hamper attacks that generate adversarial examples. Our experimental evaluations showed that architectural ensemble diversity improves the resistance of the ensembles against adversarial examples. We

also demonstrated that increasing ensemble diversity through an ensemble training process improved the robustness of the ensembles.

As a research direction for future work, investigating the impact of other diversity factors, e.g., R_{TE} to represent the correlation between ground truth labels and the ensemble's output is important. Also, using the same Gaussian noise as a data augmentation method in the training time is an interesting future research to pursue.

Acknowledgments

We would like to express our sincere gratitude to the anonymous reviewers for their valuable comments, which greatly improved the quality of this manuscript. This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant RGPIN-2016-06062.

References

Mahdieh Abbasi and Christian Gagné. Robustness to Adversarial Examples Through an Ensemble of Specialists. *ICLR*, 2017.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *ICML*, pages 284–293. PMLR, 2018.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring Neural Net Robustness with Constraints. *NeurIPS*, 2016.

L. Breiman. Bagging Predictors. *Machine Learning*, 24, 1996. doi: 10.1007/BF00058655.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *ICML*, 2019.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE conf. on CVPR*, 2018.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv*, 2014.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6), 2012.

Bo Huang, Zhiwei Ke, Yi Wang, Wei Wang, Linlin Shen, and Feng Liu. Adversarial Defence by Diversified Simultaneous Training of Deep Ensembles. *AAAI Conference*, 2021.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. In *Artificial Intelligence Safety and Security*. Chapman and Hall/CRC, 2018.

Wenjing Li, Randy C. Paffenroth, and David Berthiaume. Neural Network Ensembles: Theory, Training, and the Importance of Explicit Diversity. *arXiv*, 2021.

Yuting Liang and Reza Samavi. Towards Robust Deep Learning with Ensemble Networks and Noisy Layers. *AAAI Workshop on RSEML, https://arxiv.org/abs/2007.01507*, 2021.

Yuting Liang and Reza Samavi. Advanced defensive distillation with ensemble voting and noisy logits. *Applied Intelligence*, 53(3):3069–3094, 2023.

Ling Liu, Wenqi Wei, Ka-Ho Chow, Margaret Loper, Emre Gursoy, Stacey Truex, and Yanzhao Wu. Deep Neural Network Ensembles against Deception: Ensemble Diversity, Accuracy and Robustness. *IEEE Int. Conference on MASS*, 2019.

Ying Meng, Jianhai Su, Jason O'Kane, and Pooyan Jamshidi. ATHENA: A Framework Based on Diverse Weak Defenses for Building Adversarial Defense. *arXiv*, 2020.

Nicolas Papernot and Patrick McDaniel. On the Effectiveness of Defensive Distillation. *arXiv*, 2016.

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*, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*, 2016b.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses Against Adversarial Examples. *arXiv*, 2018.

Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial Attacks and Defenses in Deep Learning. *Engineering*, 6(3), 2020.

Amanda J. C. Sharkey and Noel E. Sharkey. Combining Diverse Neural Nets. *The Knowledge Engineering Review*, 12(3), 1997.

Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble Methods as a Defense to Adversarial Perturbations Against DNNs. *arXiv*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. *ICLR*, 2014.

Wenqi Wei and Ling Liu. Robust Deep Learning Ensemble Against Deception. *IEEE Transactions on Dependable and Secure Computing*, 18(4), 2020.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *IEEE Conference on CVPR*, 2018.