n-ML: Mitigating Adversarial Examples via Ensembles of Topologically Manipulated Classifiers

Mahmood Sharif[†]
NortonLifeLock Research Group
mahmood_sharif@symantec.com

Lujo Bauer
Carnegie Mellon University
lbauer@cmu.com

Michael K. Reiter
University of North Carolina at Chapel Hill
reiter@cs.unc.edu

arXiv:1912.09059v1 [cs.CV] 19 Dec 2019

***Abstract*—This paper proposes a new defense called n-ML against adversarial examples, i.e., inputs crafted by perturbing benign inputs by small amounts to induce misclassifications by classifiers. Inspired by n-version programming, n-ML trains an ensemble of n classifiers, and inputs are classified by a vote of the classifiers in the ensemble. Unlike prior such approaches, however, the classifiers in the ensemble are trained specifically to classify adversarial examples differently, rendering it very difficult for an adversarial example to obtain enough votes to be misclassified. We show that n-ML roughly retains the benign classification accuracies of state-of-the-art models on the MNIST, CIFAR10, and GTSRB datasets, while simultaneously defending against adversarial examples with better resilience than the best defenses known to date and, in most cases, with lower classification-time overhead.**

I. Introduction

Adversarial examples—minimally and adversarially perturbed variants of benign samples—have emerged as a challenge for machine-learning (ML) algorithms and systems. Numerous attacks that produce inputs that evade correct classification by ML algorithms, and particularly deep neural networks (DNNs), at inference time have been proposed (e.g., [7], [20], [74]). The attacks vary in the perturbation types that they allow and the application domains, but they most commonly focus on adversarial perturbations with bounded L_p-norms to evade ML algorithms for image classification (e.g., [7], [11], [20], [57], [74]). Some attacks minimally change physical artifacts to mislead recognition DNNs (e.g., adding patterns on street signs to mislead street-sign recognition) [18], [69]. Yet others imperceptibly perturb audio signals to evade speech-recognition DNNs [58], [63].

In response, researchers have proposed methods to mitigate the risks of adversarial examples. For example, one method, called adversarial training, augments the training data with correctly labeled adversarial examples (e.g., [28], [36], [47], [67]). The resulting models are often more robust in the face of attacks than models trained via standard methods. However, while defenses are constantly improving, they are still far from perfect. Relative to standard models, defenses often reduce the accuracy on benign samples. For example, methods to detect the presence of attacks sometimes erroneously detect benign inputs as adversarial (e.g., [44], [45]). Moreover, defenses often fail to mitigate a large fraction of adversarial examples that are produced by strong attacks (e.g., [4]).

Inspired by n-version programming, this paper proposes a new defense, termed n-ML, that improves upon the state of the art in its ability to detect adversarial inputs and correctly classify benign ones. Similarly to other ensemble classifiers [46], [73], [83], an n-ML ensemble outputs the majority vote if more than a threshold number of DNNs agree; otherwise the input is deemed adversarial. The key innovation in this work is a novel method, *topological manipulation*, to train DNNs to achieve high accuracy on benign samples while simultaneously classifying adversarial examples *according to specifications that are drawn at random before training*. Because every DNN in an ensemble is trained to classify adversarial examples differently than the other DNNs, n-ML is able to detect adversarial examples because they cause disagreement between the DNNs' votes.

We evaluate n-ML using three datasets (MNIST [37], CIFAR10 [35], and GTSRB [72]) and against L_∞ (mainly) and L_2 attacks in black-, grey-, and white-box settings. Our findings indicate that n-ML can effectively mitigate adversarial examples while achieving high benign accuracy. For example, for CIFAR10 in the black-box setting, n-ML can achieve 94.50% benign accuracy (vs. 95.38% for the best standard DNN) while preventing all adversarial examples with L_∞-norm perturbation magnitudes of $\frac{8}{255}$ created by the best known attack algorithms [47]. In comparison, the state-of-the-art defense achieves 87.24% benign accuracy while being evaded by 14.02% of the adversarial examples. n-ML is also faster than most defenses that we compare against. Specifically, even the slowest variant of n-ML is $\times$45.72 to $\times$199.46 faster at making inferences than other defenses for detecting the presence of attacks.

Our contributions can be summarized as follows:[1]

- We propose *topology manipulation*, a novel method to train DNNs to classify adversarial examples according to specifications that are selected at training time, while also achieving high benign accuracy.
- Using topologically manipulated DNNs, we construct (n-ML) ensembles to defend against adversarial examples.
- Our experiments using two perturbation types and three datasets in black-, grey-, and white-box settings show that n-ML is an effective and efficient defense. n-ML roughly retains the benign accuracies of state-of-the-art DNNs,

[†]Work done as a Ph.D. student at Carnegie Mellon University.

[1]We will release our implementation of n-ML upon publication.

while providing more resilience to attacks than the best defenses known to date, and making inferences faster than most.

We next present the background and related work (Sec. II). Then, we present the technical details behind n-ML and topology manipulation (Sec. III). Thereafter, we describe the experiments that we conducted and their results (Sec. IV). We close the paper with a discussion (Sec. V) and a conclusion (Sec. VI).

II. Background and Related Work

This section summarizes related work and provides the necessary background on evasion attacks on ML algorithms and defenses against them.

A. Evading ML Algorithms

Inputs that are minimally perturbed to fool ML algorithms at inference time—termed adversarial examples—have emerged as a challenge to ML. Attacks to produce adversarial examples typically start from benign inputs and find perturbations with small L_p-norms (p typically $\in \{0, 2, \infty\}$) that lead to misclassification when added to the benign inputs (e.g., [6], [7], [11], [20], [51], [57], [74], [80]). Keeping the perturbations' L_p-norms small helps ensure the attacks' imperceptibility to humans, albeit imperfectly [64], [68]. The process of finding adversarial perturbations is usually formalized as an optimization problem. For example, Carlini and Wagner [11] proposed the following formulation to find adversarial perturbations that target class c_t:

$$\arg\min_{r} \; \mathit{Loss}_{cw}(x + r, c_t) + \kappa \cdot ||r||_2$$

where x is a benign input, r is the perturbation, and κ helps tune the L_2-norm of the perturbation. Loss_{cw} is roughly defined as:

$$\mathit{Loss}_{cw}^{targ}(x + r, c_t) = \max_{c \neq c_t}\{\mathbb{L}_c(x + r)\} - \mathbb{L}_{c_t}(x + r)$$

where $\mathbb{L}_c$ is the output for class c at the logits of the DNN—the output of the one-before-last layer. Minimizing Loss_{cw} leads $x + r$ to be (mis)classified as c_t. As this formulation targets a specific class c_t, the resulting attack is commonly referred to as a *targeted attack*. In the case of evasion where the aim is to produce an adversarial example that is misclassified as any class but the true class (commonly referred to as *untargeted attack*), Loss_{cw} is defined as:

$$\mathit{Loss}_{cw}^{untarg}(x + r, c_x) = \mathbb{L}_{c_x}(x + r) - \max_{c \neq c_x}\{\mathbb{L}_c(x + r)\}$$

where c_x is the true class of x. We use Loss_{cw} as a loss function to fool DNNs.

The Projected Gradient Descent (PGD) attack is considered as the strongest first-order attack (i.e., an attack that uses gradient decent to find adversarial examples) [47]. Starting from any random point close to a benign input, PGD consistently finds perturbations with constrained L_∞- or L_2-norms that achieve roughly the same loss value. Given a benign input x, a loss function, say Loss_{cw}, a target class c_t, a step size α, and an upper bound on the norm ϵ, PGD iteratively updates the adversarial example until a maximum number of iterations is reached such that:

$$x_{i+1} = \mathit{Project}_{x,\epsilon}(x_i + \alpha \mathit{sign}(\nabla_x \mathit{Loss}_{cw}(x_i, c_t)))$$

where $\mathit{Project}_{x,\epsilon}(\cdot)$ projects vectors on the ϵ-ball around x (e.g., by clipping for L_∞-norm), and $\nabla_x \mathit{Loss}_{cw}(\cdot)$ denotes the gradient of the loss function. PGD starts from a random point x_0 in the ϵ-ball around x, and stops after a fixed number of iterations (20–100 iterations are typical [47], [67]). In this work, we rely on PGD to produce adversarial examples.

Early on, researchers noticed that adversarial examples computed against one ML model are likely to be misclassified by other models performing the same task [20], [55], [56]. This phenomenon—transferability—serves as the basis of non-interactive attacks in black-box settings, where attackers do not have access to the attacked models and may not be able to query them. The accepted explanation for the transferability phenomenon, at least between DNNs performing the same task, is that the gradients (which are used to compute adversarial examples) of each DNN are good approximators of those of other DNNs [16], [26]. Several techniques can be used to enhance the transferability of adversarial examples [17], [42], [76], [82]. The techniques include computing adversarial perturbations against ensembles of DNNs [42], [76] and misleading the DNNs after transforming the input (e.g., by translation or resizing) [17], [82] as a means to avoid computing perturbations that generalize beyond a single DNN. We leverage these techniques to create strong attacks against which we evaluate our proposed defense.

B. Defending ML Algorithms

Researchers proposed a variety of defenses to mitigate ML algorithms' vulnerability to adversarial examples. Roughly speaking, defenses can be categorized as: adversarial training, certified defenses, attack detection, or input transformation. Similarly to our proposed defense, some defenses leverage randomness or model ensembles. In what follows, we provide an overview of the different categories.

Adversarial Training Augmenting the training data with correctly labeled adversarial examples that are generated throughout the training process increases models' robustness to attacks [20], [27], [28], [36], [47], [67], [74]. The resulting training process is commonly referred to as *adversarial training*. In particular, adversarial training with PGD (Adv_{PGD}) [47], [67] is one of the most effective defenses to date—we compare our defense against it.

Certified Defenses Some defenses attempt to certify the robustness of trained ML models (i.e., provide provable bounds on models' errors for different perturbation magnitudes). Certain certified defenses estimate how DNNs transform L_p balls around benign examples via convex shapes, and attempt to force classification boundaries to not cross the shapes (e.g., [34], [50], [59]). These defenses are less effective than adversarial training with PGD [61]. Other defenses estimate the output of the so-called *smoothed classifier* by

classifying many variants of the input after adding noise at the input or intermediate layers [14], [38], [41], [60]. The resulting smoothed classifier, in turn, is proven to be robust against perturbations of different L_2-norms. Unfortunately, such defenses do not provide guarantees against perturbations of different types (e.g., ones with bounded L_∞-norms), and perform less well against them in practice [14], [38].

Attack Detection Similarly to our defense, there were past proposals for defenses to detect the presence of attacks [2], [19], [21], [25], [43], [44], [48], [49], [52], [78]. While adaptive attacks have been shown to circumvent some of these defenses [4], [10], [43], detectors often significantly increase the magnitude of the perturbations necessary to evade DNNs and detectors combined [10], [44]. In this work, we compare our proposed defense with detection methods based on Local Intrinsic Dimensionality (LID) [45] and Network Invariant Checking (NIC) [44], which are currently the leading methods for detecting adversarial examples.

The LID detector uses a logistic regression classifier to tell benign and adversarial inputs apart. The input to the classifier is a vector of LID statistics that are estimated for every intermediate representation computed by the DNN. This approach is effective because the LID statistics of adversarial examples are presumably distributed differently than those of benign inputs [45].

The NIC detector expects certain invariants in DNNs to hold for benign inputs. For example, it expects the provenance of activations for benign inputs to follow a certain distribution. To model these invariants, a linear logistic regression classifier performing the same task as the original DNN is trained using the representation of every intermediate layer. Then, for every pair of neighboring layers, a one-class Support Vector Machine (*oc-SVM*) is trained to model the distribution of the output of the layers' classifiers on benign inputs. Namely, every *oc-SVM* receives concatenated vectors of probability estimates and emits a score indicative of how similar the vectors are to the benign distribution. The scores of all the *oc-SVM*s are eventually combined to derive an estimate for whether the input is adversarial. In this manner, if the output of two neighboring classifiers on an image, say that of a bird, is $(bat, bird)$, the input is likely to be benign (as the two classes are similar and likely have been observed for benign inputs during training). However, if the output is $(car, bird)$, then it is likely that the input is adversarial.

Input Transformation Certain defenses suggest to transform inputs (e.g., via quantization) to sanitize adversarial perturbations before classification [22], [40], [48], [62], [71], [81], [84]. The transformations often aim to hinder the process of computing gradients for the purpose of attacks. In practice, however, it has been shown that attackers can adapt to circumvent such defenses [3], [4], [24].

Randomness Defenses often leverage randomness to mitigate adversarial examples. As previously mentioned, some defenses inject noise at inference time at the input or intermediate layers [14], [25], [38], [41], [60]. Differently, Wang et al. train a hierarchy of layers, each containing multiple paths, and randomly switch between the chosen paths at inference time [79]. Other defenses randomly drop out neurons, shuffle them, or change their weights, also while making inferences [19], [78]. Differently from all these, our proposed defense uses randomness at *training time* to control how a set of DNNs classify adversarial examples at inference time and uses the DNNs strategically in an ensemble to deter adversarial examples.

Ensembles Similarly to our defense, several prior defenses suggested using ensembles to defend against adversarial examples. Abbasi and Gagné proposed to measure the disagreement between DNNs and specialized classifiers (i.e., ones that classify one class versus all others) to detect adversarial examples [2]. An adaptive attack to evade the specialized classifiers and the DNNs simultaneously can circumvent this defense [24]. Vijaykeerthy et al. train DNNs sequentially to be robust against an increasing set of attacks [77]. However, they only use the final model at inference time, while we use an ensemble containing several models for inference. A meta defense by Sengupta et al. strategically selects a model from a pool of candidates at inference time to increase benign accuracy while deterring attacks [65]. This defense is effective against black-box attacks only. Other ensemble defenses propose novel training or inference mechanisms, but do not achieve competitive performance [29], [53], [73].

Recent papers [46], [83], [87] proposed defenses that, similarly to ours, are motivated by n-version programming [5], [12], [15]. In a nutshell, n-version programming aims to provide resilience to bugs and attacks by running n ($\geq$2) variants of independently developed, or diversified, programs. These programs are expected to behave identically for normal (benign) inputs, and differently for unexpected inputs that trigger bugs or exploit vulnerabilities. When one or more programs behaves differently than the others, an unexpected (potentially malicious) input is detected. In the context of ML, defenses that are inspired by n-version programming use ensembles of models that are developed by independent parties [87], different inference algorithms or DNN architectures [46], [83], or models that are trained using different training sets [83]. In all cases, the models are trained via standard training techniques. Consequently, the defenses are often vulnerable to attacks (both in black-box and more challenging settings) as adversarial examples transfer with high likelihood regardless of the inference algorithm or the training data [20], [55], [56]. Moreover, prior work is limited to specific applications (e.g., speech recognition [87]). In contrast, we train the models comprising the ensembles via a novel training technique (see Sec. III), and our work is conceptually applicable to any domain.

III. Technical Approach

Here we detail our methodology. We begin by presenting the threat model. Subsequently, we present a novel technique, termed *topology manipulation*, which serves as a cornerstone for training DNNs that are used as part of the n-ML defense.

Last, we describe how to construct an n-ML defense via an ensemble of topologically manipulated DNNs.

A. Threat Model

Our proposed defense aims to mitigate attacks in *black-*, *grey-*, and *white-box* settings. In the black-box setting, the attacker has no access to the classifier's parameters and is unaware of the existence of the defense.[2] To evade classification, the attacker attempts a non-interactive attack by transferring adversarial examples from standard surrogate models. Similarly, in the grey-box setting, the attacker cannot access the classifier's parameters. However, the attacker is aware of the use of the defense and attempts to transfer adversarial examples that are produced against a surrogate defended model. In the white-box setting, the attacker has complete access to the classifier's and defense's parameters. Consequently, the attacker can adapt gradient-based white-box attacks (e.g., PGD) to evade classification. We do not consider interactive attacks that query models in a black-box setting (e.g., [8], [9], [13], [26]). These attacks are generally weaker than the white-box attacks that we do consider.

As is typical in the area (e.g., [34], [47], [61]), we focus on defending against adversarial perturbations with bounded L_p-norms. In particular, we mainly consider defending against perturbations with bounded L_∞-norms. Additionally, we demonstrate defenses against perturbations with bounded L_2-norms as a proof of concept. Conceptually, there is no reason why n-ML should not generalize to defend against other types of attacks.

B. Topologically Manipulating DNNs

The main building block of the n-ML defense is a *topologically manipulated DNN*—a DNN that is manipulated at training time to achieve certain topological properties with respect to adversarial examples. Specifically, a topologically manipulated DNN is trained to satisfy two objectives: *1)* obtaining high classification accuracy on benign inputs; and *2)* misclassifying adversarial inputs following a certain specification. The first objective is important for constructing a well-performing DNN to solve the classification task at hand. The second objective aims to change the adversarial directions of the DNN such that an adversarial perturbation that would normally lead a benign input to be misclassified as class c_t by a regularly trained DNN would likely lead to misclassification as class $\hat{c}_t$ ($\neq c_t$) by the topologically manipulated DNN. Fig. 1 illustrates the idea of manipulating the topology of a DNN via an abstract example, while Fig. 2 gives a concrete example.

To train a topologically manipulated DNN, two datasets are used. The first dataset, D, is a standard dataset. It contains pairs (x, c_x) of benign samples, x, and their true classes, c_x. The second dataset, $\tilde{D}$, contains adversarial examples. Specifically, it consists of pairs $(\tilde{x}, c_t)$ of targeted adversarial examples, $\tilde{x}$, and the target classes, c_t. These adversarial examples are produced against reference DNNs that are trained in a standard manner (e.g., to decrease cross-entropy loss). Samples in D are used to train the DNNs to satisfy the first objective (i.e., achieving high benign accuracy). Samples in $\tilde{D}$, on the other hand, are used to topologically manipulate the adversarial directions of DNNs.

More specifically, to specify the topology of the trained DNN, we use a derangement (i.e., a permutation with no fixed points), d, that is drawn at random over the number of classes, m. This derangement specifies that an adversarial example $\tilde{x}$ in $\tilde{D}$ that targets class c_t should be misclassified as $d[c_t]$ ($\neq c_t$) by the topologically manipulated DNN. For example, for ten classes (i.e., $m = 10$), the derangement may look like $d = [1, 6, 7, 0, 2, 8, 9, 3, 5, 4]$. This derangement specifies that adversarial examples targeting class 0 should be misclassified as class $d[0] = 1$, ones targeting class 1 should be misclassified as class $d[1] = 6$, and so on. For m classes, the number of derangements the we can draw from is known as the subfactorial (denoted by $!m$), and is defined recursively as $!m = (m-1)(!(m-1)+!(m-2))$, where $!2 = 1$ and $!1 = 0$.

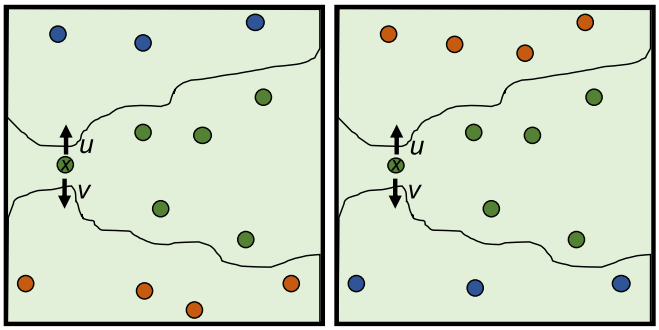
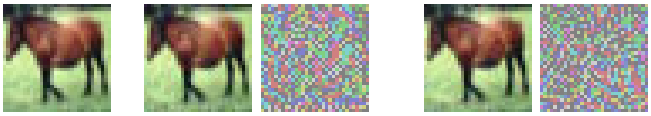


Fig. 1: An illustration of topology manipulation. Left: In a standard DNN, perturbing the benign sample x in the direction of u leads to misclassification as blue (zigzag pattern), while perturbing it in the direction of v leads to misclassification as red (diagonal stripes). Right: In the topologically manipulated DNN, direction u leads to misclassification as red, while v leads to misclassification as blue. The benign samples (including x) are correctly classified in both cases (i.e., high benign accuracy).



(a) (b) (c)

Fig. 2: A concrete example of topology manipulation. The original image of a horse (a) is adversarially perturbed to be misclassified as a bird (b) and as a ship (c) by standard DNNs. The perturbations, which are limited to L_∞-norm of $\frac{8}{255}$, are shown after multiplying $\times 10$. We train a topologically manipulated DNN to misclassify (b) as a ship and (c) as a bird, while classifying the original image correctly.

[2]For clarity, we distinguish the classifier from the defense. However, in certain cases (e.g., for n-ML or adversarial training) they are inherently inseparable.

The subfactorial grows almost as quickly as the factorial $m!$ (i.e., the number of permutations over a group of size m).

We specify the topology using derangements rather than permutations that may have fixed points because if d contained fixed points, there would exist a class c_t such that $d[c_t] = c_t$. In such case, the DNN would be trained to misclassify adversarial examples that target c_t into c_t, which would not inhibit an adversary targeting c_t. Such behavior is undesirable.

We use the standard cross-entropy loss, $Loss_{ce}$,[3] to train topologically manipulated DNNs. Formally, the training process minimizes:

$$\frac{1}{|D|}\sum_{(x,c_x)\in D} Loss_{ce}(x,c_x) + \frac{\lambda}{|\tilde{D}|}\sum_{(\tilde{x},c_t)\in \tilde{D}} Loss_{ce}(\tilde{x}, d[c_t]) \quad (1)$$

While minimizing the leftmost term increases the benign accuracy (as is usual in standard training processes), minimizing the rightmost term manipulates the topology of the DNN (i.e., forces the DNN to misclassify $\tilde{x}$ as $d[c_t]$ instead of as c_t). The parameter λ is a positive real number that balances the two objectives. We tune it via a hyperparameter search.

Although topologically manipulated DNNs aim to satisfy multiple objectives, it is important to point out that training them does not require significantly more time than for standard DNNs. For training, one first needs to create the dataset $\tilde{D}$ that contains the adversarial examples. This needs to be done only once, as a preprocessing phase. Once $\tilde{D}$ is created, training a topologically manipulated DNN takes the same amount of time as training a standard DNN.

C. n-ML: An Ensemble-Based Defense

As previously mentioned, n-ML is inspired by n-version programming. While n independent, or diversified, programs are used in an n-version programming defense, an n-ML defense contains an ensemble of n ($\geq$2) topologically manipulated DNNs. As explained above, all the DNNs in the n-ML ensemble are trained to behave identically for benign inputs (i.e., to classify them correctly), while each DNN is trained to follow a different specification for adversarial examples. This opens an opportunity to *1)* classify benign inputs accurately; and *2)* detect adversarial examples.

In particular, to classify an input x using an n-ML ensemble, we compute the output of all the DNNs in the ensemble on x. Then, if the number of DNNs that agree on a class is above or equal to a threshold τ, the input is classified to the majority class. Otherwise, the n-ML ensemble would abstain from classification and the input would be marked as adversarial. Formally, denoting the individual DNNs' classification results by the multiset $C = \{\mathbb{F}_i(x) | 1 \leq i \leq n\}$, the n-ML classification function, $\mathbb{F}$, is defined as:

$$\mathbb{F}(x) = \begin{cases} majority(C), & \text{if } |\{c \in C | c = majority(C)\}| \geq \tau \\ abstain, & \text{otherwise} \end{cases}$$

[3]In the case of one-hot encoding for the true class c_x and a probability estimate $\hat{p}_{c_x}$ emitted by the DNN, the cross-entropy loss is defined as $-\log(\hat{p}_{c_x})$.

Of course, increasing the threshold increases the likelihood of detecting adversarial examples (e.g., an adversarial example is less likely to be misclassified as the same target class c_t by all the n DNNs than by $n-1$ DNNs). In other words, increasing τ decreases attacks' success rates. At the same time, increasing the threshold harms the benign accuracy (e.g., the likelihood of n DNNs to emit c_x is lower than the likelihood of $n-1$ DNNs to do so). In practice, we set τ to a value $\geq \lceil \frac{n+1}{2} \rceil$, to avoid ambiguity when computing the majority vote, and $\leq n$, as the benign accuracy is 0 for $\tau > n$.

Similarly to n-version programming, where the defense becomes more effective when the constituent programs are more independent and diverse [5], [12], [15], an n-ML defense is more effective at detecting adversarial examples when the DNNs are more independent. Specifically, if two DNNs i and j ($i \neq j$) are trained with derangements d_i and d_j, respectively, and we are not careful enough, there might exist a class c_t such that $d_i[c_t] = d_j[c_t]$. If so, the two DNNs are likely to classify adversarial examples targeting c_t in the same manner, thus reducing the defense's likelihood to detect attacks. To avoid such undesirable cases, we train the n-ML DNNs (simultaneously or sequentially) while attempting to avoid pairs of derangements that map classes in the same manner to the greatest extent possible. More concretely, if n is lower than the number of classes m, then we draw n derangements that disagree on all indices (i.e., $\forall i \neq j$, $\forall c_t$, $d_i[c_t] \neq d_j[c_t]$). Otherwise, we split the DNNs to groups of $m-1$ (or smaller) DNNs, and for each group we draw derangements that disagree on all indices. For a group of n DNNs, $n < m$, we can draw $\prod_{i=1}^{n}!(m-i+1)$ derangements such that every pair of derangements disagrees on all indices.

IV. Results

In this section we describe the experiments that we conducted and their results. We initially present the datasets and the standard DNN architectures that we used. Then we describe how we trained individual topologically manipulated DNNs to construct n-ML ensembles, and evaluate the extent to which they met the training objectives. We close the section with experiments to evaluate the n-ML defense in various settings. We ran our experiments with Keras [30] and TensorFlow [1].

A. Datasets

We used three popular datasets to evaluate n-ML and other defenses: MNIST [37], CIFAR10 [35], and GTSRB [72]. MNIST is a dataset of 28×28 pixel images of digits (i.e., ten classes). It contains 70,000 images in total, with 60,000 images intended for training and 10,000 intended for testing. We set aside 5,000 images from the training set for validation. CIFAR10 is a dataset of 32×32-pixel images of ten classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset contains 50,000 images for training and 10,000 for testing. We set aside 5,000 images from the training set for validation. Last, GTSRB is a dataset containing traffic signs of 43 classes. The dataset contains

39,209 training images and 12,630 test images. We used 1,960 images that we set aside from the training set for validation. Images in GTSRB vary in size between 15×15 and 250×250. Following prior work [39], we resized the images to 48×48.

The three datasets have different properties that made them especially suitable for evaluating n-ML. MNIST has relatively few classes, thus limiting the set of derangements that we could use for topology manipulation for large values of n. At the same time, standard DNNs achieve high classification accuracy on MNIST ($\sim$99% accuracies are common), hence increasing the likelihood that DNNs in the ensemble would agree on the correct class for benign inputs. CIFAR10 also relatively has a few classes. However, differently from MNIST, even the best performing DNNs do not surpass $\sim$95% classification accuracy on CIFAR10. Consequently, the likelihood that a large number n of DNNs in an ensemble would achieve consensus may be low (e.g., an ensemble consisting of $n = 5$ DNNs with 95% accuracy each that incur independent errors could have benign accuracy as low as 77% if we require all DNNs to agree). In contrast, GTSRB contains a relatively high number of classes, and standard DNNs often achieve high classification accuracies on this dataset (98%–99% accuracies are common). As a result, there is a large space from which we could draw derangements for topology manipulation, and we had expected high benign accuracies from n-ML ensembles.

B. Training Standard DNNs

Dataset	#	*Architecture*	*Acc.*
MNIST	1	Convolutional DNN [11]	99.42%
	2	Convolutional DNN [47]	99.28%
	3	Convolutional DNN [11] w/o pooling [70]	99.20%
	4	Convolutional DNN [31]	99.10%
	5	Convolutional DNN [47] w/o pooling [70]	99.10%
	6	Multi-layer perceptron [32]	98.56%
CIFAR10	1	Wide-ResNet-22-8 [85]	95.38%
	2	Wide-ResNet-28-10 [85]	95.18%
	3	Wide-ResNet-16-10 [85]	95.06%
	4	Wide-ResNet-28-8 [85]	94.88%
	5	Wide-ResNet-22-10 [85]	94.78%
	6	Wide-ResNet-16-8 [85]	94.78%
GTSRB	1	Convolutional DNN [39]	99.46%
	2	Same as 1, but w/o first branch [39]	99.56%
	3	Same as 1, but w/o pooling [70]	99.11%
	4	Same as 1, but w/o second branch [39]	99.08%
	5	Convolutional DNN [75]	99.00%
	6	Convolutional DNN [66]	98.07%

TABLE I: The DNN architectures that we used for the different datasets. The DNNs' accuracies on the test sets of the corresponding datasets (after standard training) are reported to the right.

The DNNs that we used were based on standard architectures. We constructed the DNNs either exactly the same way as prior work or reputable public projects (e.g., by the Keras team [31]) or by modifying prior DNNs via a standard technique. In particular, we modified certain DNNs following the work of Springenberg et al. [70], who found that it is possible to construct simple, yet highly performing, DNNs by removing pooling layers (e.g., max- and average-pooling) and increasing the strides of convolutional operations.

For each dataset, we trained six DNNs of different architectures—a sufficient number of DNNs to allow us to evaluate n-ML and perform attacks via transferability from surrogate ensembles [42] while factoring out the effect of architectures (see Sec. IV-C and Sec. IV-D). The MNIST DNNs were trained for 20 epochs using the Adadelta optimizer with standard parameters and a batch size of 128 [31], [86]. The CIFAR10 DNNs were trained for 200 epochs with data augmentation (e.g., image rotation and flipping) and training hyperparameters set identically to prior work [85]. The GTSRB DNNs were trained with the Adam optimizer [33], with training hyperparameters and augmentation following prior work [39], [66], [75]. Table I reports the architectures and performance of the DNNs. In all cases, the DNNs achieved comparable performance to prior work.

C. Training Individual Topologically Manipulated DNNs

Now we describe how we trained individual topologically manipulated DNNs and report on their performance. Later, in Sec. IV-D, we report on the performance on the n-ML ensembles.

Training When training topologically manipulated DNNs, we aimed to minimize the loss described in Eqn. 1. To this end, we modified the training procedures that we use to train the standard DNNs in three ways:

1) We extended each batch of benign inputs with the same number of adversarial samples $(\tilde{x}, c_t) \in \tilde{D}$ and specified that $\tilde{x}$ should be classified as $d[c_t]$.
2) In certain cases, we slightly increased the number of training epochs to improve the performance of the DNNs.
3) We avoided data augmentation for GTSRB, as we found that it harmed the accuracy of (topologically manipulated) DNNs on benign inputs.

To set λ (the parameter that balances the DNN's benign accuracy and the success of topology manipulation, see Eqn. 1), we performed a hyperparameter search. We experimented with values in $\{0.1, 0.5, 1, 2, 10\}$ to find the best trade-off between the n-ML ensembles' benign accuracy and their ability to mitigate attacks. We found that $\lambda = 2$ achieved the highest accuracies at low attack success rates.

To train the best-performing n-ML ensemble, one should select the best performing DNN architectures to train topologically manipulated DNNs. However, since the goal of this paper is to evaluate a defense, we aimed to give the attacker the advantage to assess the resilience of the defense in a worst-case scenario (e.g., so that the attacker could use the better held-out DNNs as surrogates in transferability-based attacks). Therefore, we selected the DNN architectures with the lower benign accuracy to train topologically manipulated DNNs. More specifically, for each dataset, we trained n-ML ensembles by selecting round robin from the architectures shown in rows 4–6 from Table I.

Constructing a dataset of adversarial examples, $\tilde{D}$, is a key part of training topologically manipulated DNNs. As we

aimed to defend against attacks with bounded L_∞-norms, we used the corresponding PGD attack to produce adversarial examples: For each training sample x of class c_x, we produced $m-1$ adversarial examples, one targeting every class $c_t \neq c_x$. Moreover, we produced adversarial examples with perturbations of different magnitudes to construct n-ML ensembles that can resist attacks with varied strengths. For MNIST, where attacks of magnitudes $\epsilon \leq 0.3$ are typically considered [47] we used $\epsilon \in \{0.1, 0.2, 0.3, 0.4\}$. For CIFAR10 and GTSRB, where attacks of magnitudes $\epsilon \leq \frac{8}{255}$ are typically considered [47], [54], we used $\epsilon \in \{\frac{2}{255}, \frac{4}{255}, \frac{6}{255}, \frac{8}{255}\}$. (Thus, in total, $|\tilde{D}| = 4 \times (m-1) \times |D|$.) We ran PGD for 40 iterations, since prior work found that his leads to successful attacks [47], [67]. Additionally, to avoid overfitting to the standard DNNs that were used for training, we used state-of-the-art techniques to enhance the transferability of the adversarial examples, both by making the examples invariant to spatial transformations [17], [82] and by producing them against an ensemble of models [42], [76]—three standard DNNs of architectures 4–6.

For each dataset, we trained a total of 18 topologically manipulated DNNs. Depending on the setting, we used a different subset of the DNNs to construct n-ML ensembles (see Sec. IV-D). The DNNs were split into two sets of nine DNNs each ($< m$ for MNIST and CIFAR10), such that the derangements of every pair of DNNs in the same set disagreed on all indices.

Evaluation Each topologically manipulated DNN was trained with two objectives in mind: classifying benign inputs correctly (i.e., high benign accuracy) and classifying adversarial examples as specified by the derangement drawn at training time. Here we evaluate the extent to which the DNNs we trained met these objectives. Note that these DNNs were not meant to be used individually, but instead in the ensembles evaluated in Sec. IV-D.

To measure the benign accuracy, we classified the original (benign) samples from the test sets of datasets using the 18 topologically manipulated DNNs as well as the (better-performing) standard DNNs that we held out from training topologically manipulated DNNs (i.e., architectures 1–3). Table II reports the average and standard deviation of the benign accuracy. Compared to the standard DNNs, the topologically manipulated ones had only slightly lower accuracy (0.64%–2.39% average decrease in accuracy). Hence, we can conclude that topologically manipulated DNNs were accurate.

Next, we measured the extent to which topology manipulation was successful. To this end, we computed adversarial examples for the DNNs used to train topologically manipulated DNNs (i.e., architectures 4–6) or DNNs held out from training (i.e., architectures 1–3). Again, we used PGD and techniques to enhance transferability to compute the adversarial examples. As in prior work, we set $\epsilon = 0.3$ for MNIST and $\epsilon = \frac{8}{255}$ for CIFAR10 and GTSRB, and we ran PGD for 40 iterations. For each benign sample, x, we created $m-1$ corresponding adversarial examples, one targeting every class $c_t \neq c_x$. To reduce the computational load, we used a random subset of benign samples from the test sets: 1024 samples for MNIST and 512 samples for the other datasets.

For constructing robust n-ML ensembles, the topologically manipulated DNNs should classify adversarial examples as specified during training, or, at least, differently than the adversary anticipates. We estimated the former via the *manipulation success rate* (*MSR*)—the rate at which adversarial examples were classified as specified by the derangements drawn at training time—while we estimated the latter via the *targeting success rate* (*TSR*)—the rate at which adversarial examples succeeded at being misclassified as the target class. A successfully trained topologically manipulated DNN should obtain a high *MSR* and a low *TSR*.

Table II presents the average and standard deviation of *TSR*s and *MSR*s for topologically manipulated DNNs, as well as the *TSR*s for standard DNNs. One can immediately see that targeting was much less likely to succeed for a topologically manipulated DNN (average *TSR*$\leq$6.82%) than for a standard DNN (average *TSR*$\geq$20.17%, and as high as 98.57%). In fact, across all datasets and regardless of whether the adversarial examples were computed against held-out DNNs, the likelihood of targeting to succeed for standard DNNs was $\times$6.31 or higher than for topologically manipulated DNNs. This confirms that the adversarial directions of topologically manipulated DNNs were vastly different than those of standard DNNs. Furthermore, as reflected in the *MSR* results, we found that topologically manipulated DNNs were likely to classify adversarial examples as specified by their corresponding derangements. Across all datasets, and regardless of whether the adversarial examples were computed against held-out DNNs or not, the average likelihood of topologically manipulated DNNs to classify adversarial examples according to specification was $\geq$48.26%. For example, an average of 99.99% of the adversarial examples produced against the held-out DNNs of CIFAR10 were classified according to specification by the topologically manipulated DNNs.

In summary, the results indicate that the topologically manipulated DNNs satisfied their objectives to a large extent: they accurately classified benign inputs, and their topology with respect to adversarial directions was different than that of standard DNNs, as they often classified adversarial examples according to the specification that was selected at training time.

D. Evaluating n-ML Ensembles

Now we describe our evaluation of n-ML ensembles. We begin by describing the experiment setup and reporting the benign accuracy of ensembles of various sizes and thresholds. We then present experiments to evaluate n-ML ensembles in various settings and compare n-ML against other defenses. We finish with an evaluation of the time overhead incurred when deploying n-ML for inference.

1) Setup: The n-ML ensembles that we constructed were composed of the topologically manipulated DNNs described in the previous section. Particularly, we constructed ensembles containing five (5-ML), nine (9-ML), or 18 (18-ML) DNNs, as

	Standard		Topologically manipulated				
Dataset	*Acc.*	*TSR*	*Acc.*	*TSR*	*TSR h/o*	*MSR*	*MSR h/o*
MNIST	99.30%±0.09%	43.05%±7.97%	98.66%±0.42%	<0.01%	6.82%±2.45%	99.98%±0.02%	53.23%±14.94%
CIFAR10	95.21%±0.13%	98.57%±0.66%	92.93%±0.39%	<0.01%	<0.01%	99.98%±0.02%	99.99%±0.01%
GTSRB	99.38%±0.19%	20.17%±1.48%	96.99%±1.45%	1.20%±0.27%	1.35%±0.27%	52.86%±9.05%	48.26%±4.68%

TABLE II: The performance of topologically manipulated DNNs compared to standard DNNs. For standard DNNs, we report the average and standard deviation of the (benign) accuracy and the targeting success rate (*TSR*). *TSR* is defined as the rate at which the DNN emitted the target class on a transferred adversarial example. For topologically manipulated DNNs, we report the average and standard deviation of the accuracy, the *TSR*, and the manipulation success rate (*MSR*). *MSR* is the rate at which adversarial examples were classified as specified by the derangements drawn at training time. *TSR* and *MSR* are reported for adversarial examples produced against the same DNNs used during training or ones produced against held-out (*h/o*) DNNs.

we found ensembles of certain sizes to be more suitable than others at balancing benign accuracy, security, and inference time in different settings. For number of variants $n \leq 9$, we selected DNNs whose derangements disagreed in all indices for the ensembles. For $n = 18$, we selected all the DNNs. Note that since GTSRB contains a large number of classes ($m = 43$), we could train 18 DNNs with derangements that disagreed on all indices. However, we avoided doing so to save compute cycles, as the DNNs that we trained performed well despite having derangements that agreed on certain indices.

Other Defenses We compared n-ML with three state-of-the-art defenses: Adv_{PGD} [47], [67], *LID* [45], and *NIC* [44].

Adv_{PGD} and *LID* use adversarial examples at training time; we set the magnitude of the L_∞-norm perturbations to $\epsilon = 0.3$ to produce adversarial examples for MNIST, and to $\epsilon = \frac{8}{255}$ for CIFAR10 and GTSRB, as these are typical attack magnitudes that defenses attempt to prevent (e.g., [47], [54]).

For Adv_{PGD}, we implemented and used the *free adversarial training* method of Shafahi et al. [67], which adversarially trains DNNs in the same amount of time as standard training. We used Adv_{PGD} to train four defended DNNs for each dataset—one to be used by the defender, and three to be used for transferring attacks in the grey-box setting (see below). To give the defense an advantage, we used the best performing architecture for the defender's DNN (architecture 1 from Table I), and the least performing architectures for the attacker's DNNs (architectures 4–6). For CIFAR10, as the DNN that we obtained after training did not perform as well as prior works', we used the adversarially trained DNN released by Madry et al. [47] as the defender's DNN.

For training *LID* detectors, we used the implementation that was published by the authors [45]. As described in Sec. II, *LID* detectors compute *LID* statistics for intermediate representations of inputs and feed the statistics to a logistic regression classifier to detect adversarial examples. The logistic regression classifier is trained using *LID* statistics of benign samples, adversarial examples, and noisy variants of benign samples (created by adding non-adversarial Gaussian noise). We tuned the amount of noise for best performance (i.e., highest benign accuracy and detection rate of adversarial examples). For CIFAR10 and GTSRB, we trained *LID* detectors for DNNs of architecture 1. For MNIST, we trained a *LID* detector for the DNN architecture that was used in the original work—architecture 4.

Using code that we obtained directly from the authors, we trained two *NIC* detectors per dataset—one to be used by the defender (in all settings), and one by the attacker in the grey-box setting. The defender's detectors were trained for DNNs of the same architectures as for *LID*. For the attacker, we trained detectors for DNNs of architectures 1, 4, and 2, for MNIST, CIFAR10, and GTSRB, respectively. We selected the attackers' DNN architectures arbitrarily, and expect that other architectures would perform roughly the same. The *oc-SVM*s that we trained for *NIC* have Radial Basis Functions (RBF) as kernels, since these were found to perform best for detection [44].

Attack Methodology We evaluated n-ML and other defenses against untargeted attacks, as they are easier to attain from the point of view of attackers, and more challenging to defend against from the point of view of the defender. For Adv_{PGD}, *LID*, and *NIC*, we used typical *PGD* untargeted attacks with various adaptations depending on the setting. Typical untargeted attacks, however, are unlikely to evade n-ML ensembles, as if the target is not specified by the attack then each DNN in the ensemble may classify the resulting adversarial example differently, thus detecting the presence of an attack. To address this, we used a more powerful attack, similarly to Carlini and Wagner [11]. The attack builds on targeted *PGD* to generate adversarial examples targeting all possible incorrect classes (i.e., $m - 1$ in total) and checks if one of these adversarial examples is misclassified by a large number of DNNs in the ensemble ($\geq \tau$), and so is not detected as adversarial by the n-ML ensemble. Because targeting every possible class increases the computational load of attacks, we used random subsets of test sets to produce adversarial examples against n-ML. In particular, we used 1,000 samples for MNIST and 512 samples for CIFAR10 and GTSRB. The magnitude of L_∞-norm of perturbations that we considered were $\epsilon \leq 0.3$ for MNIST and $\epsilon \leq \frac{8}{255}$ for CIFAR10 and GTSRB, and we ran attacks for 40 iterations. Again, we used techniques to attack ensembles and enhance the transferability of attacks [17], [42], [76], [82].

When directly attacking a DNN defended by *LID*, we simply produced adversarial examples that were misclassified with high confidence against the DNN while ignoring the

defense. This approach is motivated by prior work, which found that high confidence adversarial examples mislead *LID* with high likelihood [4]. When attacking a DNN defended by *NIC*, we created a new DNN by combining the logits of the original DNN and those of the classifiers built on top of every intermediate layer. We found that forcing the original DNN and the intermediate classifiers to (mis)classify adversarial examples in the same manner often led the *oc-SVM*s to misclassify adversarial examples as benign.

Measures In the context of adversarial examples, a defense's success is measured in its ability to prevent adversarial examples, while maintaining high benign accuracy (e.g., close to that of a standard classifier). The benign accuracy is the rate at which benign inputs are classified correctly and not detected as adversarial. In contrast, the success rate of attacks is indicative of the defense's ability to prevent adversarial examples (high success rate indicates a weak defense, and vice versa). For untargeted attacks, the success rate can be measured by the rate at which adversarial examples are not detected and are classified to a class other than the true class. Note that Adv_{PGD} is a method for robust classification, as opposed to detection, and so adversarially trained DNNs always output an estimation of the most likely class (i.e., abstaining from classifying an input that is suspected to be adversarial is not an option).

We tuned the defenses at inference time to compute different tradeoffs on the above metrics. In the case of n-ML, we computed the benign accuracy and attacks' success rates for threshold values $\lceil \frac{n+1}{2} \rceil \leq \tau \leq n$. For *LID*, we evaluated the metrics for different thresholds $\in [0, 1]$ on the logistic regression's probability estimates that inputs are adversarial. *NIC* emits scores in arbitrary ranges—the higher the score the more likely the input to be adversarial. We computed accuracy and success rate tradeoffs of *NIC* for thresholds between the minimum and maximum values emitted for benign samples and adversarial examples combined. In all cases, both the benign accuracy and attacks' success rates decreased as we increased the thresholds. Adv_{PGD} results in a single model that cannot be tuned at inference time. We report the single operating point that it achieves.

2) Results: We now present the results of our evaluations, in terms of benign accuracy; resistance to adversarial examples in the black-, grey-, and white-box settings; and overhead to classification performance.

Benign Accuracy We first report on the benign accuracy of n-ML ensembles. In particular we were interested in finding how different was the accuracy of ensembles from single standard DNNs. Ideally, it is desirable to maintain accuracy that is as close to that of standard training as possible.

Fig. 3 compares n-ML ensembles' accuracy with standard DNNs, as well as with hypothetical ensembles whose DNN members have the average accuracy of topologically manipulated DNNs and independent errors. For low thresholds, it can be seen that the accuracy of n-ML was close to the average accuracy of standard benign DNNs. As we increased the thresholds, the accuracy decreased. Nonetheless, it did not decrease as dramatically as for ensembles composed from DNNs with independent error. For example, the accuracy of an ensemble containing five independent DNNs each with an accuracy of 92.93% (the average accuracy of topologically manipulated DNNs on CIFAR10) is 69.31% when $\tau = 5$ (i.e., we require all DNNs to agree). In comparison, 5-ML achieved 84.82% benign accuracy for the same threshold on CIFAR10.

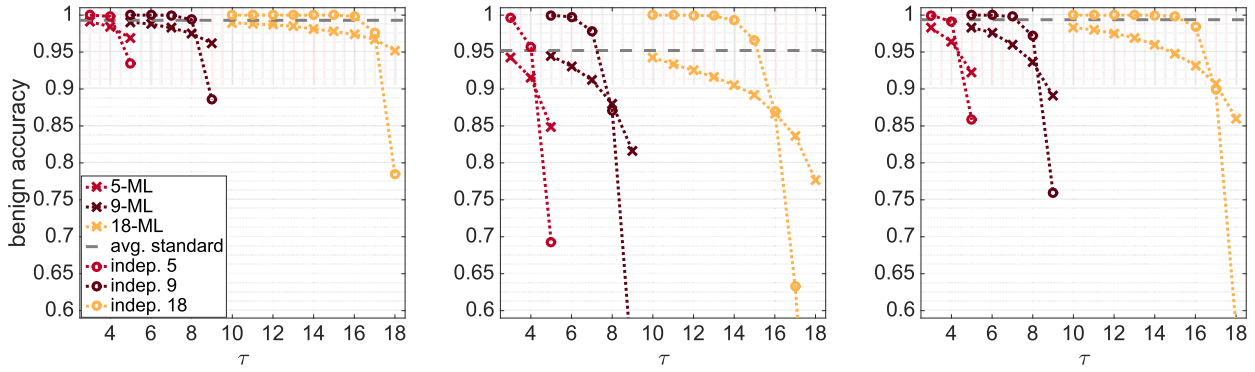
We can conclude that n-ML ensembles were almost as accurate as standard models for low thresholds, and that they did not suffer from dramatic accuracy loss as thresholds were increased.

Black-box Attacks In the black-box setting, as the attacker is unaware of the use of defenses and has no access to the classifiers, we used non-interactive transferability-based attacks to transfer adversarial examples produced against standard surrogate models. For n-ML and Adv_{PGD}, we used a strong attack by transferring adversarial examples produced against the standard DNNs held-out from training the defenses. For *LID* and *NIC* we found that transferring adversarial examples produced against the least accurate standard DNNs (architecture 6) was sufficient to evade classification with high success rates.

Fig. 4 summarizes the results for the attacks with the highest magnitudes. For n-ML, we report the performance of 5-ML and 9-ML, which we found to perform well in the black-box setting (i.e., they achieved high accuracy while mitigating attacks). It can be seen that n-ML outperformed other defenses across all the datasets. For example, for CIFAR10, 9-ML was able to achieve 94.50% benign accuracy at 0.00% attack success rate. In contrast, the second best defense, Adv_{PGD}, achieved 87.24% accuracy at 14.02% attack success rate.

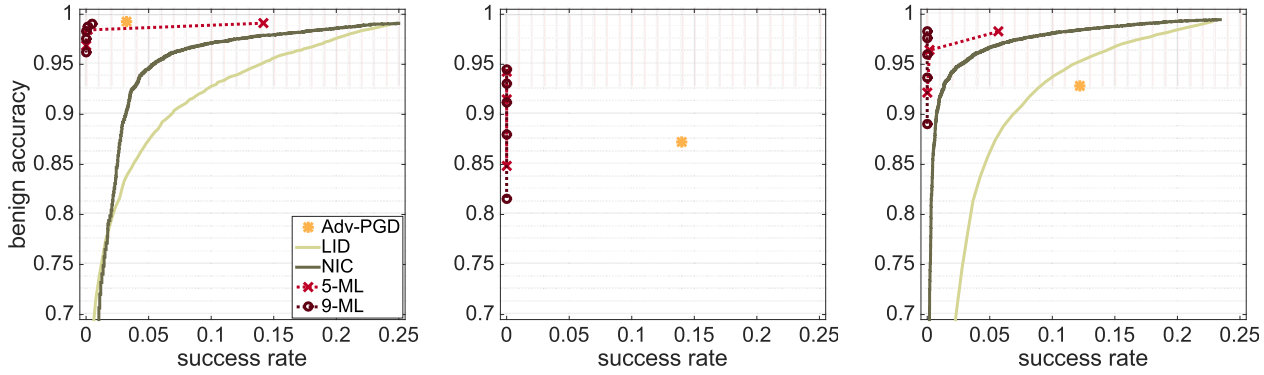
While not shown for the sake of brevity, additional experiments that we performed demonstrated that n-ML in the black-box setting performed roughly the same as in Fig. 4 when *1)* different perturbation magnitudes were used ($\epsilon \in \{0.05, 0.1, \ldots, 0.4\}$ for MNIST and $\epsilon \in \{\frac{1}{255}, \frac{2}{255}, \ldots, \frac{8}{255}\}$ for CIFAR10 and GTSRB); *2)* individual standard DNNs were used as surrogates to produce adversarial examples; and *3)* the same DNNs used to train topologically manipulated DNNs were used as surrogates.

Grey-box Attacks In the grey-box setting (where attackers are assumed to be aware of the deployment of defenses, but have no visibility to the parameters of the classifier and the defense), we attempted to transfer adversarial examples produced against surrogate defended classifiers. For n-ML, we evaluated 5-ML and 9-ML in the grey-box setting. As surrogates, we used n-ML ensembles of the same sizes and architectures to produce adversarial examples. The derangements used for training the DNNs of the surrogate ensembles were picked independently from the defender's ensembles (i.e., the derangements could agree with the defender's derangements on certain indices). For Adv_{PGD}, we used three adversarially trained DNNs different than the defender's DNNs as surrogates. For *NIC* we used standard DNNs and corresponding detectors (different than the defender's, see above for training details) as surrogates. For *LID*, we simply produced adver-



(a) MNIST (b) CIFAR10 (c) GTSRB

Fig. 3: The benign accuracy of n-ML ensembles of different sizes as we varied the thresholds. For reference, we show the average accuracy of a single standard DNN (avg. standard), as well as the accuracy of hypothetical ensembles whose constituent DNNs are assumed to have independent errors and the average accuracy of topologically manipulated DNNs (indep. n). The dotted lines connecting the markers were added to help visualize trends, but do not correspond to actual operating points.



(a) MNIST (b) CIFAR10 (c) GTSRB

Fig. 4: Comparison of defenses' performance in the *black-box setting*. The L_∞-norm of perturbations was set to $\epsilon = 0.3$ for MNIST and $\epsilon = \frac{8}{255}$ for CIFAR10 and GTSRB. Due to poor performance, the *LID* and *NIC* curves were left out from the CIFAR10 plot after zooming in. The dotted lines connecting the n-ML markers were added to help visualize trends, but do not correspond to actual operating points.

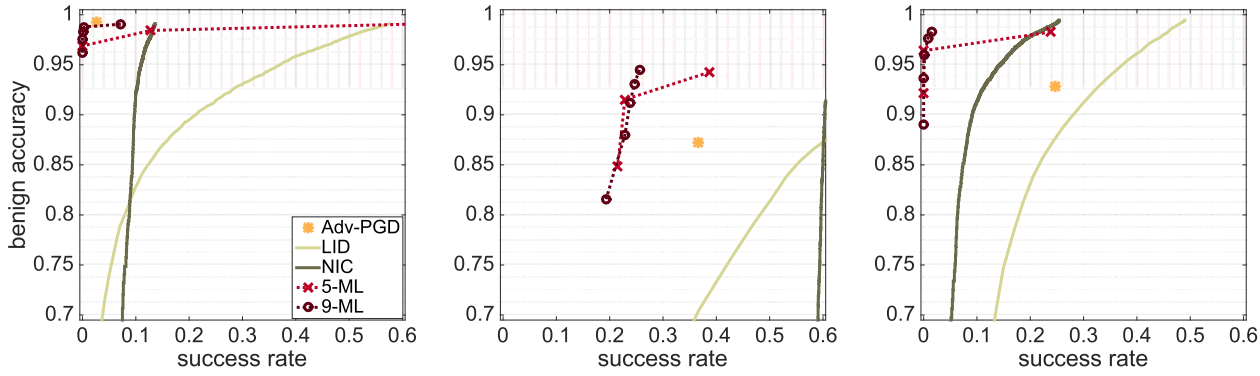
sarial examples that were misclassified with high confidence against undefended standard DNNs of architecture 2 (these were more architecturally similar to the defenders' DNNs than the surrogates used in the black-box setting).

Fig. 5 presents the performance of the defenses against the attacks with the highest magnitudes. Again, we found n-ML to achieve favorable performance over other defenses. In the case of GTSRB, for instance, 9-ML could achieve 98.30% benign accuracy at 1.56% attack success rate. None of the other defenses was able to achieve a similar accuracy while preventing $\geq$98.44% of the attacks for GTSRB.

Additional experiments that we performed showed that n-ML maintained roughly the same performance as we varied the number of DNNs in the attacker's surrogate ensembles ($n \in \{1, 5, 9\}$) and the attacks' magnitudes ($\epsilon \in \{0.05, 0.1, \ldots, 0.4\}$ for MNIST and $\epsilon \in \{\frac{1}{255}, \frac{3}{255}, \frac{5}{255}, \frac{7}{255}\}$ for CIFAR10 and GTSRB).

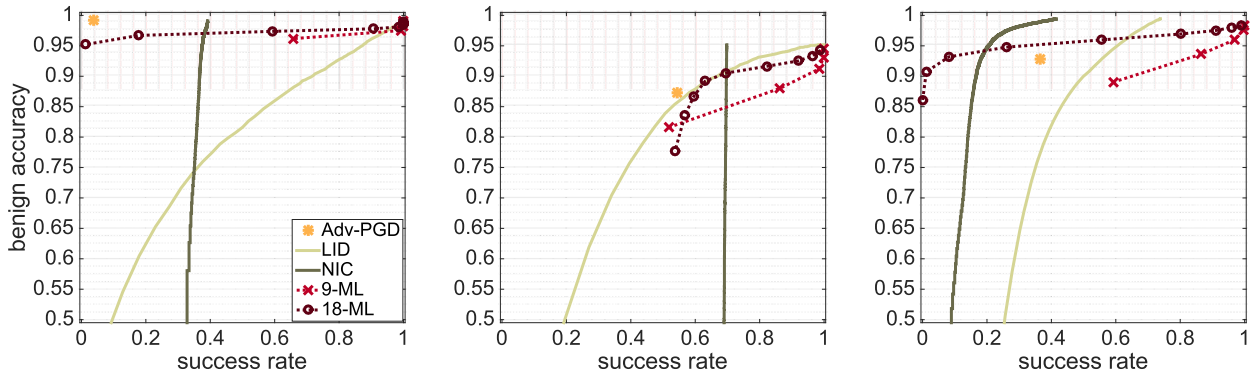
White-Box Attacks Now we turn our attention to the white-box setting, where attackers are assumed to have complete access to classifiers' and defenses' parameters. In this setting, we leveraged the attacker's knowledge of the classifiers' and defenses' parameters to directly optimize the adversarial examples against them.

Fig. 6 shows the results. One can see that, depending on the dataset, n-ML outperformed other defenses, or achieved comparable performance to the leading defenses. For GTSRB, n-ML significantly outperformed other defenses: 18-ML achieved a benign accuracy of 86.01%–93.19% at attack success-rates $\leq$8.20%. No other defense achieved comparable benign accuracy for such low attack success-rates. We hypothesize that n-ML was particularly successful for GTSRB, since the dataset contains a relatively large number of classes, and so there was a large space from which derangements for topology manipulation could be drawn. The choice of the leading



(a) MNIST (b) CIFAR10 (c) GTSRB

Fig. 5: Comparison of defenses' performance in the *grey-box setting*. The L_∞-norm of perturbations was set to $\epsilon = 0.3$ for MNIST and $\epsilon = \frac{8}{255}$ for CIFAR10 and GTSRB. The dotted lines connecting the n-ML markers were added to help visualize trends, but do not correspond to actual operating points.



(a) MNIST (b) CIFAR10 (c) GTSRB

Fig. 6: Comparison of defenses' performance in the *white-box setting*. The L_∞-norm of perturbations was set to $\epsilon = 0.3$ for MNIST and $\epsilon = \frac{8}{255}$ for CIFAR10 and GTSRB. The dotted lines connecting the n-ML markers were added to help visualize trends, but do not correspond to actual operating points.

defense for MNIST and CIFAR10 is less clear (some defenses achieved slightly higher benign accuracy, while others were slightly better at preventing attacks), and depends on the need to balance benign accuracy and resilience to attacks at deployment time. For example, 18-ML was slightly better at preventing attacks against MNIST than Adv_{PGD} (1.30% vs. 3.70% attack success-rate), but Adv_{PGD} achieved slightly higher accuracy for the same 18-ML operating point (99.25% vs. 95.22%).

L_2-norm Attacks The previous experiments showed that n-ML ensembles could resist L_∞-based attacks in various settings. We performed a preliminary exploration using MNIST to assess whether n-ML could also prevent L_2-based attacks. Specifically, we trained 18 topologically manipulated DNNs to construct n-ML ensembles. The training process was the same as before, except that we projected adversarial perturbations to the L_2 balls around benign samples when performing PGD to produce adversarial examples for $\tilde{D}$. We created adversarial examples with $\epsilon \in \{0.5, 1, 2, 3\}$, as we aimed to prevent adversarial examples with $\epsilon \leq 3$, following prior work [47]. The resulting topologically manipulated DNNs were accurate (an average accuracy of 98.39%±0.55%).

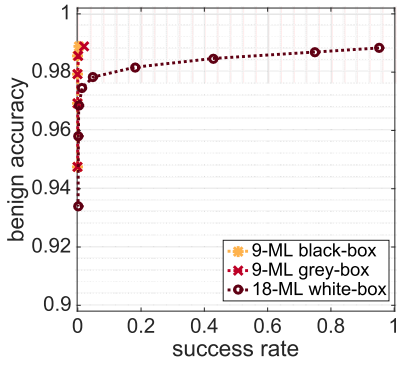


Fig. 7: Performance of MNIST n-ML ensembles against L_2-norm PGD attacks with $\epsilon = 3$.

Dataset	*Standard / Adv_{PGD}*	*5-ML*	*9-ML*	*18-ML*	*LID*	*NIC*
MNIST	0.15ms	×1.93	× 2.80	× 4.73	×943.47	× 601.07
CIFAR10	3.72ms	×3.57	× 6.07	× 12.07	×551.88	× 581.51
GTSRB	0.68ms	×5.35	× 8.26	× 16.41	×852.04	×1457.26

TABLE III: Defenses' overhead at inference time. The second column reports the average inference time in milliseconds for batches containing 32 samples for a single (standard or adversarially trained) DNN. The columns to its right report the overhead of defenses with respect to using a single DNN for inference.

sing the models that we trained, we constructed n-ML ensembles of different sizes and evaluated attacks in black-, grey-, and white-box settings. The evaluation was exactly the same as before, except that we use L_2-based PGD with $\epsilon = 3$ to produce adversarial examples. Fig. **??** summarizes the results for 9-ML in black- and grey-box settings, and for 18-ML in a white-box setting. It can be seen that n-ML was effective at preventing L_2-norm attacks while maintaining high benign accuracy. For example, 9-ML was able to achieve 98.56% accuracy at ∼0% success rates for black- and grey-box attacks, and 18-ML was able to achieve 97.46% accuracy at a success rate of 1.40% for white-box attacks.

Overhead Of course, as n-ML requires us to run several DNNs for inference instead of only one, using n-ML comes at the cost of increased inference time at deployment. Now we show that the overhead is relatively small, especially compared to LID and NIC.

To measure the inference time, we sampled 1024 test samples from every dataset and classified them in batches of 32 using the defended classifiers. We used 32 because it is common to use for inspecting the behavior of DNNs [23], but the trends in the time estimates remained the same with other batch sizes. We ran the measurements on a machine equipped with 64GB of memory and 2.50GHz Intel Xeon CPU using a single NVIDIA Tesla P100 GPU.

The results are shown in Table III. Adv_{PGD} did not incur time overhead at inference due to producing a single DNN to be used for inference. Compared to using a single DNN, inference time with n-ML increased with n. At the same time the increase was often sublinear in n. For example, for 18-ML, the inference time increased ×4.73 for MNIST, ×12.07 for CIFAR10, and ×16.41 for GTSRB. Moreover, the increase was significantly less dramatic than for LID (×551.88–943.47 increase) and NIC (×581.51–1,457.26 increase). There, we observed that the main bottlenecks were computing the LID statistics for LID, and classification with *oc-SVM*s for NIC.

V. Discussion

Our experiments demonstrated the effectiveness and efficiency of n-ML against L_∞ and L_2 attacks in various settings for three datasets. Still, there are some limitations and practical considerations to take into account when deploying n-ML. We discuss these below.

Limitations Our experiments evaluated n-ML against L_2 and L_∞ attacks, as is typical in the literature (e.g., [47], [67]). However, in reality, attackers can use other perturbation types to evade n-ML (e.g., adding patterns to street signs to evade recognition [18]). Conceptually, it should be possible to train n-ML to defend against such perturbation types. We defer the evaluation to future work.

As opposed to using a single ML algorithm for inference (e.g., one standard DNN), n-ML requires using n DNNs. As a result, more compute resources and more time are needed to make inferences with n-ML. This may make it challenging to deploy n-ML in settings where compute resources are scarce and close to real-time feedback is expected (e.g., face-recognition on mobile phones). Nonetheless, it is important to highlight that n-ML is remarkably faster at making inferences than state-of-the-art methods for detecting adversarial examples [44], [45], as our experiments showed.

Currently, perhaps the most notable weakness of n-ML is that it is limited to scenarios where the number of classes, m, is large. In cases where m is small, one cannot draw many distinct derangements to train DNNs with different topologies with which to construct n-ML ensembles. For example, when there are two classes ($m = 2$), there is only one derangement that one can use to train a topologically manipulated DNN (remember that $!2 = 1$), and so it is not possible to construct an ensemble containing $n \geq 2$ DNNs with distinct topologies. A possible solution is to find a new method that does not require derangements to topologically manipulate DNNs. We plan to pursue this direction in future work.

Practical considerations ML systems often take actions based on inferences made by ML algorithms. For example, a biometric system may give or deny access to users based on the output of a face-recognition DNN; an autonomous vehicle may change the driving direction, accelerate, stop, or slow down based on the output of a DNN for pedestrian detection; and an anti-virus program may delete or quarantine a file based on the output of an ML algorithm for malware detection. This raises the question of how should a system that uses n-ML for inference react when n-ML flags an input as adversarial.

We have a couple of suggestions for courses of action that are applicable to different settings. One possibility is to fall back to a more expensive, but less error prone, classification mechanism. For example, if an n-ML ensemble is used for face recognition and it flags an input as adversarial, a security guard may be called to identify the person, and possibly override the output of the ensemble. This solution is viable when the time and resources to use an expensive classification mechanism are available. Another possibility is to resample the input, or classify a transformed variant of the input, to increase the confidence in the detection or to correct the classification result. For example, if an n-ML ensemble that is used for face recognition on a mobile phone detects an input as adversarial, the user may be asked to reattempt identifying herself using a new image. In this case, because the benign accuracy of n-ML is high and the attack success rate is low, a benign user

is likely to succeed at identifying, while an attacker is likely to be detected.

VI. Conclusion

This paper presents n-ML, a defense that is inspired by n-version programming, to defend against adversarial examples. n-ML uses ensembles of DNNs to classify inputs by a majority vote (when a large number of DNNs agree) and to detect adversarial examples (when the DNNs disagree). To ensure that the ensembles have high accuracy on benign samples while also defending against adversarial examples, we train the DNNs using a novel technique (*topology manipulation*) which allows one to specify how adversarial examples should be classified by the DNN at inference time. Our experiments using two perturbation types (ones with bounded L_2- and L_∞-norms) and three datasets (MNIST, CIFAR10, and GTSRB) in black-, grey-, and white-box settings showed that n-ML is an effective and efficient defense. In particular, n-ML roughly retains the benign accuracies of state-of-the-art DNNs, while providing more resilience to attacks than the best defenses known to date and making inferences faster than most.

Acknowledgments

This work was supported in part by the Multidisciplinary University Research Initiative (MURI) Cyber Deception grant; by NSF grants 1801391 and 1801494; by the National Security Agency under Award No. H9823018D0008; by gifts from Google and Nvidia, and from Lockheed Martin and NATO through Carnegie Mellon CyLab; and by a CyLab Presidential Fellowship and a NortonLifeLock Research Group Fellowship.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Proc. OSDI*, 2016.

[2] M. Abbasi and C. Gagné. Robustness to adversarial examples through an ensemble of specialists. In *Proc. ICLRW*, 2017.

[3] A. Athalye and N. Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

[4] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. ICML*, 2018.

[5] A. Avizienis. The N-version approach to fault-tolerant software. *IEEE Transactions on software engineering*, (12):1491–1501, 1985.

[6] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. In *Proc. AAAI*, 2018.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Proc. ECML PKDD*, 2013.

[8] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proc. ICLR*, 2018.

[9] T. Brunner, F. Diehl, M. T. Le, and A. Knoll. Guessing smart: Biased sampling for efficient black-box adversarial attacks. In *Proc. ICCV*, 2019.

[10] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. AISec*, 2017.

[11] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Proc. IEEE S&P*, 2017.

[12] L. Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *Proc. ISFTC*, 1995.

[13] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proc. AISec*, 2017.

[14] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. Certified adversarial robustness via randomized smoothing. In *Proc. ICML*, 2019.

[15] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser. N-variant systems: A secretless framework for security through diversity. In *Proc. USENIX Security*, 2006.

[16] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli. Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In *Proc. USENIX Security*, 2019.

[17] Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proc. CVPR*, 2019.

[18] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on machine learning models. In *Proc. CVPR*, 2018.

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

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

[21] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[22] C. Guo, M. Rana, M. Cisse, and L. van der Maaten. Countering adversarial images using input transformations. In *Proc. ICLR*, 2018.

[23] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, and P. Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training. In *Proc. SOSP*, 2019. To appear.

[24] W. He, J. Wei, X. Chen, N. Carlini, and D. Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *Proc. WOOT*, 2017.

[25] B. Huang, Y. Wang, and W. Wang. Model-agnostic adversarial detection by random perturbations. In *Proc. IJCAI*, 2019.

[26] A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *Proc. ICLR*, 2019.

[27] H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[28] A. Kantchelian, J. Tygar, and A. D. Joseph. Evasion and hardening of tree ensemble classifiers. In *Proc. ICML*, 2016.

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

[30] Keras team. Keras: The Python deep learning library. https://keras.io/, 2015. Accessed on 09-30-2019.

[31] Keras team. MNIST CNN. https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py, 2018. Accessed on 09-28-2019.

[32] Keras team. MNIST MLP. https://keras.io/examples/mnist_mlp/, 2018. Accessed on 09-28-2019.

[33] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[34] J. Z. Kolter and E. Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proc. ICML*, 2018.

[35] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[36] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *Proc. ICLR*, 2017.

[37] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. Accessed on 10-01-2019.

[38] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *Proc. IEEE S&P*, 2019.

[39] J. Li and Z. Wang. Real-time traffic sign recognition based on efficient CNNs in the wild. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):975–984, 2018.

[40] F. Liao, M. Liang, Y. Dong, T. Pang, J. Zhu, and X. Hu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proc. CVPR*, 2018.

[41] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh. Towards robust neural networks via random self-ensemble. In *Proc. ECCV*, 2018.

[42] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. ICLR*, 2017.

[43] P.-H. Lu, P.-Y. Chen, and C.-M. Yu. On the limitation of local intrinsic dimensionality for characterizing the subspaces of adversarial examples. *arXiv preprint arXiv:1803.09638*, 2018.

[44] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang. NIC: Detecting adversarial samples with neural network invariant checking. In *Proc. NDSS*, 2019.

[45] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *Proc. ICLR*, 2018.

[46] F. Machida. N-version machine learning models for safety critical systems. In *Proc. DSN DSMLW*, 2019.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. ICLR*, 2018.

[48] D. Meng and H. Chen. Magnet: A two-pronged defense against adversarial examples. In *Proc. CCS*, 2017.

[49] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. In *Proc. ICLR*, 2017.

[50] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proc. ICML*, 2018.

[51] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *Proc. CVPR*, 2016.

[52] T. Pang, C. Du, Y. Dong, and J. Zhu. Towards robust detection of adversarial examples. In *Proc. NeurIPS*, 2018.

[53] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu. Improving adversarial robustness via promoting ensemble diversity. In *Proc. ICML*, 2019.

[54] N. Papernot and P. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[55] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[56] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proc. AsiaCCS*, 2017.

[57] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. IEEE Euro S&P*, 2016.

[58] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *Proc. ICML*, 2019.

[59] A. Raghunathan, J. Steinhardt, and P. S. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Proc. NeurIPS*, 2018.

[60] H. Salman, G. Yang, J. Li, P. Zhang, H. Zhang, I. Razenshteyn, and S. Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proc. NeurIPS*, 2019. To appear.

[61] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Proc. NeurIPS*, 2019. To appear.

[62] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *Proc. ICLR*, 2018.

[63] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Proc. NDSS*, 2019.

[64] A. Sen, X. Zhu, L. Marshall, and R. Nowak. Should adversarial attacks use pixel p-norm? *arXiv preprint arXiv:1906.02439*, 2019.

[65] S. Sengupta, T. Chakraborti, and S. Kambhampati. MTDeep: Moving target defense to boost the security of deep neural nets against adversarial attacks. In *Proc. GameSec*, 2019.

[66] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Proc. IJCNN*, 2011.

[67] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In *Proc. NeurIPS*, 2019. To appear.

[68] M. Sharif, L. Bauer, and M. K. Reiter. On the suitability of lp-norms for creating and preventing adversarial examples. In *Proc. CVPRW*, 2018.

[69] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. CCS*, 2016.

[70] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proc. ICLR*, 2015.

[71] V. Srinivasan, A. Marban, K.-R. Müller, W. Samek, and S. Nakajima. Counterstrike: Defending deep learning architectures against adversarial samples by Langevin dynamics with supervised denoising autoencoder. *arXiv preprint arXiv:1805.12017*, 2018.

[72] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[73] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423*, 2017.

[74] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[75] L. Tian. Traffic sign recognition using CNN with learned color and spatial transformation. https://github.com/hello2all/GTSRB_Keras_STN/blob/master/conv_model.py, 2017. Accessed on 09-28-2019.

[76] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *Proc. ICLR*, 2018.

[77] D. Vijaykeerthy, A. Suri, S. Mehta, and P. Kumaraguru. Hardening deep neural networks via adversarial model cascades. 2019.

[78] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang. Adversarial sample detection for deep neural network through model mutation testing. In *Proc. ICSE*, 2019.

[79] X. Wang, S. Wang, P.-Y. Chen, Y. Wang, B. Kulis, X. Lin, and P. Chin. Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses. *arXiv preprint arXiv:1908.07116*, 2019.

[80] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. In *Proc. IJCAI*, 2018.

[81] C. Xie, Y. Wu, L. van der Maaten, A. Yuille, and K. He. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411*, 2018.

[82] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proc. CVPR*, 2019.

[83] H. Xu, Z. Chen, W. Wu, Z. Jin, S.-y. Kuo, and M. Lyu. NV-DNN: Towards fault-tolerant DNN systems with N-version programming. In *Proc. DSN DSMLW*, 2019.

[84] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proc. NDSS*, 2018.

[85] S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proc. BMVC*, 2016.

[86] M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[87] Q. Zeng, J. Su, C. Fu, G. Kayas, L. Luo, X. Du, C. C. Tan, and J. Wu. A multiversion programming inspired approach to detecting audio adversarial examples. In *Proc. DSN*, 2019.