**MULDEF: Multi-model-based Defense Against Adversarial Examples for Neural Networks**

Siwakorn Srisakaokul*, Zexuan Zhong*, Yuhao Zhang†, Wei Yang‡ and Tao Xie*

*University of Illinois at Urbana-Champaign
†Peking University
‡University of Texas at Dallas

\{srisaka2, zexuan2, weiyang3, taoxie\}@illinois.edu,\{zhang_yuhao\}@pku.edu.cn

**Abstract**—Despite being popularly used in many application domains such as image recognition and classification, neural network models have been found to be vulnerable to adversarial examples: given a model and an example correctly classified by the model, an adversarial example is a new example formed by applying small perturbation (imperceptible to human) on the given example so that the model misclassifies the new example. Adversarial examples can pose potential risks on safety or security in real-world applications. In recent years, given a vulnerable model, defense approaches, such as adversarial training and defensive distillation, improve the model to make it more robust against adversarial examples. However, based on the improved model, attackers can still generate adversarial examples to successfully attack the model. To address such limitation, we propose a new defense approach, named MULDEF, based on the design principle of diversity. Given a target model (as a seed model) and an attack approach to be defended against, MULDEF constructs additional models (from the seed model) together with the seed model to form a family of models, such that the models are complementary to each other to accomplish robustness diversity (i.e., one model’s adversarial examples typically do not become other models’ adversarial examples), while maintaining about the same accuracy for normal examples. At runtime, given an input example, MULDEF randomly selects a model from the family to be applied on the given example. The robustness diversity of the model family and the random selection of a model from the family together lower the success rate of attacks. Our evaluation results show that MULDEF substantially improves the target model’s accuracy on adversarial examples by 35–50% and 2–10% in the white-box and black-box attack scenarios, respectively, while still maintaining about the same accuracy as the target model on normal examples.

**Keywords**—neural networks; adversarial examples; defense approach;

I. INTRODUCTION

Neural networks recently have been used to solve many real-world tasks such as image recognition and classification and can reach very high performance on such tasks [1]. Although nowadays a lot of examples are available to be used to train a neural network model for a certain task, it is still very hard to make the model robust against given examples. Given a normal example that the model can correctly classify, previous research [2–5] proposed various attack approaches to perturb the example by applying only a slight modification (which is not noticeable to human eyes) on the example to fool the model, i.e., causing the model to misclassify the perturbed example. We name such perturbed example adversarial example and the model being attacked target model. These attack approaches can be used in two attack scenarios: (1) white-box attack scenarios where the attackers have complete knowledge about the target model (and also its defense approach), and (2) black-box attack scenarios where the attackers do not know anything about the target model (and its defense approach), but know the output of the model, given an arbitrary example.

While there are various effective attack approaches falling into the two attack scenarios, there are only a limited number of defense approaches such as adversarial training [2, 6] and defensive distillation [7]. Adversarial training is a simple defense approach, which improves model robustness by augmenting the training set with some adversarial examples. The key idea of adversarial training is to make the model more general and have some exposure to adversarial examples. Defensive distillation is another defense approach that hides the gradient of the target model by modifying the softmax function used in the last layer of the model to increase the magnitude of the inputs to the softmax layer, because some attack approaches [6, 8] rely on maximizing the loss function by computing the model’s gradient.

However, these defense approaches still have three main limitations. (1) Ineffectiveness to re-attack. the improved target model resulted from some defense approaches such as adversarial training is still vulnerable to new adversarial examples generated by white-box attack approaches based on the improved target model. These attack approaches rely on computing the model’s gradient. Even after the defense approach of adversarial training improves the model with additional adversarial examples in the training set, the attack approaches can still compute the gradient of the improved model and generate new adversarial examples. Our evaluation results (shown in Section VI) show that adversarial training alone is not effective in a white-box attack scenario, where attackers have complete knowledge about the model (i.e., weights, architecture). One variation of adversarial training that is more effective includes modification of the loss function used to optimize the model parameters [5], but that variation requires manual efforts of changing the target model’s implementation. (2) Ineffectiveness to transferable attack. Adversarial examples can have the transferability property: adversarial examples can be used to transfer attacks across models [9], so attackers can train a substitute model (which is easier to attack) for the target model and generate adversarial examples for the substitute model to indirectly attack the target model [2]. This transferability property can also be used to attack the target model in a black-box attack scenario, where attackers can train
a substitute model for the black-box target model to indirectly attack the target model \cite{10}. (3) Bypassable distillation. Even after defensive distillation, attackers can still compute the gradient of the actual output of the network (the gradient of the inputs to the pre-softmax layer) and reduce the magnitude of the inputs to the softmax layer \cite{11}.

To address these three main limitations, in this paper, we propose a new defense approach, named MULDEF, consisting of two components (the model generator and runtime model selector) designed based on the design principle of diversity. Such diversity helps hide implementation details from adversaries and introduces uncertainty in the target model \cite{12}. Given a target model (as a seed model) and an attack approach to be defended against, the model generator constructs additional models (from the seed model) together with the seed model to form a family of models. At runtime, given an input example, the runtime model selector randomly selects a model from the family to be applied on the given example.

In particular, to address the limitations of ineffectiveness to re-attack and ineffectiveness to transferable attack, the model generator constructs additional diverse models together with the target model to form a family. The construction of the model family is to assure that an adversarial example generated for one model in the family cannot fool at least another model (desirably as many other models as possible) in the family to break the transferability property, while maintaining about the same accuracy for normal examples across models in the family. Constructing these additional diverse models in the family can increase the cost for the attackers to generate adversarial examples that can fool many or most of the models in the family.

To address the limitation of bypassable distillation, the runtime model selector uses a random strategy (which has low runtime cost) to select a model in the family at runtime to be applied on a given example such that the attackers do not know beforehand which model to compute the gradient even when distillation can be bypassed. Note that generally (deterministic) runtime analysis (such as multi-model majority voting \cite{13} \cite{14}) can be conducted on the given example and all the models in the family in order to select a model that is likely to be robust to the example. However, such runtime analysis can be costly and not scalable. In addition, with a candidate adversarial example, the attackers can also simulate such runtime analysis ahead of time and know beforehand which model in the family to (re-)compute the gradient to improve the candidate adversarial example if needed.

To increase the chance that runtime model selection randomly selects a model robust to a given example, the model generator leverages the ideas from both adversarial training \cite{2} \cite{6} and defensive distillation \cite{7} to generate models in the family to satisfy the following two properties: (1) most models in the family must have high accuracy on normal examples and (2) given an adversarial example, the majority of the models in the family are robust to the example. For the first property, the model generator constructs each additional model by using the same architecture and parameter configuration as the target model and the majority of the training examples are from the original training set used to train the target model. For the second property, the model generator constructs and trains each model in the family with some additional adversarial examples for other earlier-constructed models (i.e., adversarial training).

We design our MULDEF approach based on our two insights of how attack approaches work. Our first insight is that many attack approaches rely on the target model to generate adversarial examples that work well specifically on the target model. Thus, our approach constructs other models to help defend adversarial examples for the target model. Although Papernot et al. \cite{10} find that adversarial examples largely transfer well across models trained with the same machine learning technique, and across models trained with different techniques, it is still not necessarily the case that these adversarial examples can transfer to other models trained with some other adversarial examples. Our evaluation results (Section \textbf{VI}) show that these adversarial examples can also attack other models trained with some other adversarial examples, but these adversarial examples are not as effective as they are for the target model itself, so adversarial training can still be used to construct other additional models. Note that because our approach leverages the idea of adversarial training to construct other models, it still needs to know what attack approach the attacker is using in order to construct the family of models.

Our second insight is that to the best of our knowledge, no attack approach is intentionally built to attack multiple models at the same time. Existing attacks typically rely on computing the target model’s gradient, which is based on its loss function. It is challenging to combine the gradients of multiple models, because each model has its own loss function. Our approach constructs a family of models, and then randomly selects one model in the family to be applied on a given adversarial example; doing so can lower the success rate of the attack because the runtime model selection may not select a model vulnerable to the given example. This randomization also forces the attackers to generate an adversarial example that works on multiple models in the family to achieve a high success rate of attack.

We investigate how the design principle of diversity can help defend against adversarial examples for neural networks and evaluate our defense approach against two attack approaches in white-box and black-box scenarios: (1) Fast Gradient Sign Method (FGSM) \cite{5} and (2) Carlini & Wagner attack (C&W) \cite{2}. FGSM is fast, simple, and can be powerful. C&W is a state-of-the-art attack approach to the best of our knowledge. We use the MNIST dataset \cite{15} and CIFAR-10 dataset \cite{16} to train our target model. The evaluation results show that our defense approach substantially improves the robustness of the target model in the case of a white-box attack scenario, and slightly improves the target model in the case of a black-box attack scenario.

In summary, this paper makes the following main contributions:

- A new defense approach for adversarial example attack, which is based on the design principle of diversity.
- An evaluation of our approach on two attack approaches. The evaluation results show that our approach can substantially improve the robustness of the target model in a white-box attack scenario and perform reasonably well in a black-box attack scenario,
while maintaining about the same accuracy (as the target model) on normal examples.

- Defending by having multiple models in our approach opens up more future research on how to create additional models to cooperate with the runtime model selection.

II. BACKGROUND

In this section, we illustrate the terminology and basic attack and defense approaches in previous work.

A. Normal vs. Adversarial Examples

In this paper, we focus on addressing adversarial example attack on neural network models for a classification task. A normal example \( x \) is an example that occurs naturally \([17]\) for the classification task. For example, if the classification task is about classifying digits, normal examples can be images of real digits without other figures. Usually, the training set consists of only normal examples. An adversarial example \([18]\) \( x' \) is an example similar to a normal example with imperceptible or unnoticeable changes (of the normal example) that can change the target model's prediction.

B. Attack Approaches

We aim to evaluate our defense approach against two state-of-the-art attack approaches: FGSM \([6]\) and an attack approach proposed by Carlini and Wagner \([2]\), whose details are discussed below:

Fast Gradient Sign Method (FGSM). FGSM \([6]\) generates adversarial example \( x' \) from normal example \( x \) by applying a perturbation on \( x \). Let \( l \) be the ground-truth label of \( x \). Let \( J(x, l) \) be the loss function of classifying \( x \) as label \( l \). For each pixel, FGSM updates the pixel by the sign of the gradient of the loss function at the pixel. Formally, FGSM iteratively modifies \( x \) as follows:

\[
x' = x + \epsilon \cdot \text{sign} (\nabla_x J(x, l))
\]

Note that the gradient can be computed using backpropagation on the neural network model. Thus, FGSM works very well on white-box attack when it has complete knowledge about the model.

Carlini & Wagner attack (C&W). C&W \([2]\) generates adversarial examples with small perturbation \( \delta \) through the following optimization

\[
\delta = \arg \min_{\delta} D(x, x + \delta') + c \cdot f(x + \delta')
\]

where \( D \) is a distance metric, which can be \( L_0, L_2, \) or \( L_{\infty} \). \( c \) is a constant to balance constraints, and objective function \( f(\cdot) \) is designed in such a way that \( f(x') \leq 0 \) if and only if the classifier misclassifies \( x' \), indicating that the attack succeeds. C&W uses a logits-based objective function instead of the cross-entropy loss used in other attacks such as FGSM. Using such logits-based objective function results in that C&W is able to generate adversarial examples robust against the defensive distillation. C&W performs gradient-descent-based algorithms to search for a solution to the optimization problem.

C. Defense Approaches

Next, we describe two main defense approaches introduced in previous work. We leverage these two approaches to apply to the design principle of diversity for MULDEF.

Adversarial training. Adversarial training defends against adversarial examples by training a better model being more robust on adversarial examples. The idea of adversarial training is to make the target model more general and have some exposure to adversarial examples. A straightforward way is to augment the training set by replacing some training samples with the corresponding adversarial examples generated by an attack approach \([19]\). Also, one can train the model using an adversarial objective function to improve the robustness and generalizability of a classifier \([20]\).

Defensive distillation. Some attack approaches rely on optimizing an objective function by computing in the gradient of the target model. Thus it would be useful for a defender to hide the gradient of the target model. Defensive distillation \([7]\) trains a classifier to cause a rapid reduction of its gradient over an input, resulting in that an attacker can hardly perform an attack requiring computing gradients. Defensive distillation hides the gradient between the pre-softmax layer and softmax outputs by using distillation training.

D. Datasets

We evaluate our approach on two public datasets as listed below.

MNIST is a dataset of handwritten digits, consisting of ten labels for the ten digits. We select 60,000 examples for the training set and 10,000 examples for the test set. Each image is a 28x28 black and white image.

CIFAR-10 is a widely used dataset consisting of ten labels. We select 50,000 examples for the training set and 10,000 examples for the test set. So there are 6,000 images per class. Each image is a 32x32 color image.

III. PROBLEM DEFINITION AND TERMINOLOGY

In this section, we first define some terminologies used to explain adversarial examples and how to evaluate attack and defense approaches.

A. Terminology

For a certain image classification task \( t \), we define the following terms:

Sample space \( \mathbb{S} \): the set of all images in the sample space.

Label space \( \mathbb{L} \): the set of all labels/classes in the classification task \( t \). For example, we have 10 labels, each of which represents each digit (0-9) for a digit classification task. Thus, \( \mathbb{L} = \{0, 1, 2, \ldots, 9\} \).

A classifier \( f \) for the classification task \( t \) can be defined as a function that takes as input an image in \( \mathbb{S} \) and outputs the class in \( \mathbb{C} \) for the image.

Normal examples \( \mathbb{N} \): the set of all normal image examples that occur naturally \([17]\) for the classification task \( t \). Thus, \( \mathbb{N} \subseteq \mathbb{S} \). During the training process of neural network models,
we assume that the training set and test set contain only normal examples.

The class for each example for the task \( t \) is determined by human judgment. Normally, the dataset being used to train a classifier for the task contains each example along with its class in \( \mathbb{L} \) (derived with human judgment).

An adversarial example \( x \) for the classification task \( t \) and the classifier \( f \) is the example that is not normal \((x \not\in \mathbb{M})\) and the classifier \( f \) mispredicts the class of \( x \).

An attack approach takes a normal example and then tries to generate an adversarial example that is not significantly different from the normal example. Thus in our evaluation, we control each attack approach’s configuration to generate only an adversarial example that is similar to the given normal example in order to avoid human effort to label the adversarial example by just using the label from the normal example. Moreover, if an adversarial example is significantly different from the given normal example, it can be hard even for human judgment to determine what the label should be.

\section{Attack Scenarios}

Attack approaches can be evaluated in two main scenarios:

\paragraph*{White-box attack.} The attacker has complete knowledge of the target model. Such knowledge includes the internal weights of a neural network and its network architecture. The attacker also knows the parameters of the defense approach and how the defense works.

\paragraph*{Black-box attack.} The attacker does not know anything about the target model and the defense approach. The target model and defense are just a black-box model where the attacker can only invoke the model with any arbitrary input and get back its output.

In our evaluation, we also consider these two attack scenarios to see how well \textsc{MulDef} performs.

\section{Evaluating Defense Approaches}

There are various defense approaches as discussed in Section 1. The defense goal is to make the target model more robust against its adversarial example attack, while still preserving the target model’s accuracy on normal examples. It is undesirable if a defense approach makes the model perform worse on normal examples. In contrast to the attacker, the defender is assumed to have complete knowledge about the target model under defense. The defender may or may not know about the attack approach that the attacker is using. Thus, some defense approaches can be independent of the process of generating adversarial examples (by an attack approach). In other words, these defense approaches such as \textsc{MagNet} do not require knowledge about the used attack approach and are designed to defend any kind of attack approaches. However, these defense approaches face difficulties in defending against white-box attacks. Our \textsc{MulDef} approach is attack-dependent, because it takes into account what attack approach it is defending against.

In general, defense approaches extend the target model in multiple major ways. Some approaches improve the target model against adversarial examples by modifying the weights and parameters of the target model. Some approaches do not modify the target model at all, but detect and modify adversarial examples before passing them to the target model. Our \textsc{MulDef} approach does not modify the target model nor adversarial examples, but treat the defense as constructing a new classifier \( D \) as a whole. We evaluate our \textsc{MulDef} approach on two metrics:

\paragraph*{Test accuracy:} the accuracy of a classifier on a normal example in the test set. The defense \( D \) makes a correct decision on a normal example if and only if \( D \) outputs the same class as the class associated with the normal example in the test set.

\paragraph*{Adversarial accuracy:} the accuracy of a classifier on an adversarial example generated by an attack approach. The defense \( D \) makes a correct decision on an adversarial example if and only if \( D \) outputs the same class as the class associated with the normal example used to generate the adversarial example. This assessment is reasonable because we make sure that each attack approach generates only a similar example to the given normal example.

We also evaluate our defense approach on the two attack scenarios.

\section{The MulDef Approach}

We design our \textsc{MulDef} approach for improving a neural network model to make it more robust against adversarial examples generated by a given attack approach. The design of the \textsc{MulDef} approach is based on the design principle of diversity.

\textsc{MulDef} consists of two components: (1) the model generator and (2) the runtime model selector. Given a target model (seed model), and an attack approach to be defended against, the model generator constructs additional models (from the seed model) together with the seed model to form a family of models, such that the models are complementary to each other. We leverage the ideas from adversarial training to help train addition models so that an adversarial example for one model can be defended by many other models in the family. At runtime, the runtime model selector randomly selects a model from the family to be applied on a given example. Because the attackers do not know in advance which model will be selected, the attackers cannot compute the model’s gradient and can hardly perform an attack that requires computing gradients, such as FGSM and C&W.

In particular, \textsc{MulDef} illustrated in Figure 1 takes as input a target model, training set, and an attack approach to be defended against. Then \textsc{MulDef} constructs a family of models and randomly selects a model to be applied on a given input example at runtime. We design \textsc{MulDef} to be easy to use, low cost, and scalable. \textsc{MulDef} does not modify or analyze the target model’s implementation but only improves the training set.

We next illustrate how the two components in our \textsc{MulDef} approach address two major challenges, respectively: (1) how can we construct other additional models in the family so that they are complementary to each other? (2) how can we combine the use of these models together at runtime? We also discuss our empirical exploration on design choices for our approach.
with the union of the original training set and adversarial observation to explain such results is that in the first solution, Solution 1 performs better than Solution 2. Let the 

\[ \text{Solution 1: } M_i \text{ is constructed as the union of the original training set and the adversarial examples generated for the previously constructed model } \text{Adv}_{M_{i-1}}. \]

\[ \text{Solution 2: } M_i \text{ is constructed as the union of the original training set and the adversarial examples generated for each of the previously constructed models } \text{Adv}_{T}, \text{Adv}_{M_1}, \text{Adv}_{M_2}, \ldots, \text{Adv}_{M_{i-1}}. \]

Our evaluation results (Section VI) show that the second solution of augmenting the training set with all the adversarial examples generated for each of the previously constructed models (Solution 2) performs better than Solution 1. One observation to explain such results is that in the first solution, \text{Adv}_{M_2} may not be representative for \text{Adv}_{M_1}. So \text{M}_3 (trained with the union of the original training set and \text{Adv}_{M_2}) can still be vulnerable to \text{Adv}_{M_1}. In the second solution, we train \text{M}_3 with the union of the original training set and \text{Adv}_{T}, \text{Adv}_{M_1}, \text{Adv}_{M_2} to make \text{M}_3 more robust against all the previously constructed models’ adversarial examples. Thus, we implement MULDEF by following the second solution. It is worth noting that the last model, which is trained with all the other models’ adversarial examples, seems to be more robust than other models. However, the reason that we still include other models in our defense is that the last model is vulnerable to its own adversarial set (\text{Adv}_M), and all the models in the family could be complementary to each other.

Figure 2 illustrates the idea of having multiple models and why Solution 2 performs better than Solution 1. Let the rectangle in each solution represent the set of all adversarial examples for the target model (\text{Adv}_T \subset S) under a given attack approach. This set can be infinite. MULDEF incrementally constructs additional models one by one. First, MULDEF constructs \text{M}_1 aiming to defend against a subset of \text{Adv}_T by training \text{M}_1 with some adversarial examples for \text{T}. The circle for \text{M}_1 covers the subset of \text{Adv}_T that \text{M}_1 can defend. MULDEF keeps constructing more models to cover more space in the rectangle, indicating that MULDEF is getting more robust to adversarial examples. Intuitively, MULDEF performs well when many of the constructed models cover a large portion of the rectangle. The difference between Solution 1 and Solution 2 is that MULDEF trains \text{M}_3 with the union of the original training set and \text{Adv}_T, \text{Adv}_{M_1}, \text{Adv}_{M_2} in Solution 1, but trains \text{M}_3 with the union of the original training set and \text{Adv}_T, \text{Adv}_{M_1}, \text{Adv}_{M_2} in Solution 2. Thus, \text{M}_3 in Solution 2 is likely able to defend against adversarial examples for \text{M}_1, resulting in having a higher chance that a given adversarial example can be defended by all the three models (\text{M}_1, \text{M}_2, \text{M}_3). According to Figure 2, a given adversarial example \text{x} can be defended by only \text{M}_1 and \text{M}_2 in Solution 1. However, \text{x} can be defended by all the three models in Solution 2. A higher number of additional models that are robust to \text{x} results in a higher chance that MULDEF selects a right model at runtime.

**B. Runtime Model Selector**

To combine multiple models together, MULDEF randomly selects a model from the family of models \text{T}, \text{M}_1, \text{M}_2, \text{M}_3, \ldots, \text{M}_p to compute the class label for each given input example. The intuition of this strategy is to be able to introduce uncertainty in the target model (by the design principle of diversity) within the family of models so that it is hard for the attackers to generate adversarial examples that can
attack all the models. Moreover, this runtime model selection acts as a wall to hide the gradient of a single model, because the attackers do not know in advance which model \textsc{MulDef} ends up selecting at runtime. This random strategy shares the same spirit of defensive distillation \cite{defensive_distillation}, which attempts to hide the gradient of the target model.

\subsection*{C. Empirical Exploration on Design Choices}

Before settling down on our approach, we also conduct some experiments to see whether using only adversarial training can make the target model more robust against adversarial examples. We create two convolutional neural network models that can achieve about 99.13%/80.4% test accuracy (on the test set) for both MNIST and CIFAR-10. Then we use FGSM to attack the model in the white-box attack scenario. Without adversarial training, the model has about 8.87%/13.79% adversarial accuracy against FGSM for MNIST/CIFAR-10. Then we try augmenting adversarial examples (generated by FGSM) to the training set to see whether the model is more robust. Note that we run the experiment three times and report the average accuracy. Figure \ref{fig:adv_acc} shows that augmenting adversarial examples for both datasets can even worsen the model: the more adversarial examples we augment, the lower adversarial accuracy the model achieves. According to Figure \ref{fig:adv_acc}, the adversarial accuracy of target model $T$ for MNIST/CIFAR-10 surprisingly goes down to under 2.50%/13.00% when we augment the training set with more than 50% of the original training set’s size.

Thus, we believe that we should try to construct another model $D$ that is robust to the target model’s adversarial examples (Adv$_T$). We then generate another set of adversarial examples for $T$ (Adv$_T'$) and use it along with the original training set to train model $D$ with the same architecture as $T$ to see how $D$ performs on Adv$_T$. The results in Figure \ref{fig:model_acc} show that the accuracy of model $D$ on Adv$_T$ is higher when we augment more adversarial examples especially in the beginning for both datasets. Then the accuracy converges to around 97%/74% for MNIST/CIFAR-10. Notice that the adversarial accuracy does not significantly change when we augment more than around 15% and 25% of adversarial examples for MNIST and CIFAR-10. So we decide that in our \textsc{MulDef} approach, we construct other models by using 15% and 25% of adversarial examples to augment the training set for MNIST and CIFAR-10, respectively.

One may wonder why we decide to augment the training set with adversarial examples. We can also use only adversarial examples for $T$ to train other additional models. If we use only adversarial examples to train other additional models, the additional models will perform worse on normal examples (in the test set), not being desirable. Our random strategy in runtime model selection relies on the fact that almost all models in the family must perform as good as $T$ on normal examples to ensure that \textsc{MulDef} reaches about the same test accuracy as $T$’s as before.

Based on the preceding observation, we hypothesize that simply retraining the target model with adversarial examples cannot significantly make the model more robust against future adversarial examples as the attack approach also knows everything about the retrained target model. However, having multiple models helps as an adversarial example for one model may not be able to fool another model. Runtime model selection helps combine multiple models together. \textsc{MulDef} always randomly selects a model to classify any given input at runtime. One intuition for using this random strategy is that we want to make sure that any attack approach cannot deterministically know in advance which model will be selected for any adversarial example. This random strategy is simple and low cost but can also provide some probabilistic guarantee that \textsc{MulDef} will not likely select a model that is vulnerable to a given adversarial example when we have many models. We also conduct experiments to investigate on how many additional models \textsc{MulDef} should contain, given that having too many models can make the approach not scalable.

\section*{V. Evaluation Setup and Implementation}

\subsection*{A. Evaluation Setup}

In this section, we discuss the datasets that we use to evaluate our \textsc{MulDef} defense approach, against two existing attack approaches, and how we perform our evaluations in the two attack scenarios (white-box and black-box attacks). Note that we run all the evaluations three times and report...
C&W adversarial examples are stronger than FGSM. Note that FGSM generates more perturbation than C&W. But adversarial examples generated by different attack approaches in the original training sets, whereas the other images are samples. The left-most images in Figure 4 are normal examples for the visual comparison of adversarial examples and normal samples. Table I shows the parameter setting, and Figure 4 shows the visual comparison of adversarial examples and normal samples. The left-most images in Figure 4 are normal examples in the original training sets, whereas the other images are adversarial examples generated by different attack approaches. Note that FGSM generates more perturbation than C&W. But C&W adversarial examples are stronger than FGSM.

### Datasets
We conduct our evaluations on two public datasets: MNIST and CIFAR-10 (see Section I).

### Target model
We create two different convolutional neural network models for the two datasets (MNIST and CIFAR-10) as follows:

- **MNIST:** We follow the previous study on FGSM [5] and use the same model in the study to evaluate FGSM with MNIST. The model mainly consists of three convolutional layers with 64 neurons for each layer and ReLu as the activation function, and a densely-connected layer of 10 neurons for each digit.

- **CIFAR-10:** We slightly adjust the model for CIFAR-10 in the study on C&W [2]. The model mainly consists of four convolutional layers with 64 neurons for the first two layers, 128 neurons for the next two layers, and ReLu as their activation function, two densely-connected layers of 256 neurons, and a densely-connected layer of 10 neurons for each class. The only difference is that we add dropouts in both the convolutional layers and densely-connected layers. We also add l2-norm regularization in the first two densely-connected layers.

We set the max epoch equal to 10 for the MNIST model and 50 for the CIFAR-10 model. Two training processes both adopt the early stopping technique used to avoid overfitting. The technique stops the training process when the validation loss fails to reduce by at least 0.001 for 5 epochs.

The target model for MNIST achieves 98–99% test accuracy and the target model for CIFAR-10 achieves 76–81% test accuracy.

### Attack approaches
We evaluate MULDEF on two attack approaches: FGSM and C&W (see Section I). To check whether a model outputs a correct label for an adversarial example generated by each attack approach, we need to set the parameters of the attack approach to constrain the amount of perturbation on examples. In other words, the correct label for an adversarial example generated from a normal example should be the same as the original label for the normal example. Table I shows the parameter setting, and Figure 4 shows the visual comparison of adversarial examples and normal samples. The left-most images in Figure 4 are normal examples in the original training sets, whereas the other images are adversarial examples generated by different attack approaches. Note that FGSM generates more perturbation than C&W. But C&W adversarial examples are stronger than FGSM.

In FGSM, the degree of perturbation is controlled by the parameter \( \epsilon \). As shown in Table I, we first take the default value of \( \epsilon \) for MNIST. However, this value creates large perturbation in CIFAR-10, resulting in unrecognizable images for human eyes. So we repeatedly reduce the value by 0.05 until we confirm that the images are human recognizable. We set the same \( \epsilon \) for both white-box and black-box scenarios.

In C&W, the degree of perturbation is controlled by the parameter \( \text{confidence} \). We set this value to 0.01 for both MNIST and CIFAR-10 in the white-box attack scenario. In order to enhance the black-box attack, we increase this value to 10 for MNIST and 30 for CIFAR-10. Another parameter named \( \text{max~iterations} \) is used to control the max number of iterations for generating adversarial examples. We set \( \text{max~iterations} \) to 300 for MNIST and 100 for CIFAR-10 because the values are high enough to generate adversarial examples reducing the target model to 0%. Other parameters in C&W are set as their default values.

### White-box and black-box scenarios
For the white-box attack scenario, the attackers have complete knowledge about the target model, and the attackers can leverage the target model to freely generate adversarial examples from initial test examples. The attackers have the freedom to read the target model’s internal weights or compute its gradients, depending on the attackers’ technique. We evaluate the target model on the adversarial examples by comparing its outputs to the original labels.

For the black-box attack scenario where the attackers can access only the target model output. We first train a substitute model with synthetic inputs selected by a Jacobian-based heuristic [3] to approximate the target model’s decision boundaries. We use 150 hold-out images from the test set and run 5 Jacobian-based augmentation epochs, and set the augmentation parameter \( \lambda = 0.1 \). All of these parameters are default values. Then we apply white-box attacks on the substitute model to generate adversarial examples. Finally, we evaluate the target model on the adversarial examples.

### MULDEF setup
There are two main parameters to configure MULDEF: (1) the percentage of adversarial examples augmented to the training set, and (2) the number of additional models to be constructed. As discussed in Section IV, we select 15%/25% as the percentage of augmented adversarial examples for MNIST/CIFAR-10. To explore an optimal number of additional models, we set the number of additional models to be 1, 2, 3, and 4. Thus in total, MULDEF has at most 5 models including the target model.

### Implementation
We implement MULDEF in a tool and conduct all the evaluations in Python 3.0. We use TensorFlow 1.8.0 [21] for the machine learning computation and Keras (the Python Deep Learning library version 2.1.6) [22] for neural networks. All the implementations can be found on the project website.

### VI. Evaluation Results
We measure the performance of MULDEF against FGSM and C&W in both white-box and black-box attack scenarios.

---

1Project website: URL anonymized due to double blind submission
using two datasets: MNIST and CIFAR-10.

A. Performance of MulDEF in White-box Attack Scenario

To evaluate our approach in the white-box attack scenario, we perform two types of attacks to MulDEF:

1) **Indirect Attack:** Because MulDEF constructs a family of models: $T, M_1, M_2, \ldots$, an attacker can attack MulDEF in a divide-and-conquer fashion by attacking each model separately. For white-box attacks, we use FGSM and C&W to generate adversarial examples for each model.

2) **Direct Attack:** We combine multiple models by using the Merge layer in the Keras library so that MulDEF can be regarded as a single model. We can directly attack the combined model as the target model. Only C&W is used in this attack because FGSM crashes and fails to work on MulDEF (due to the runtime model selection, MulDEF cannot compute the gradients that FGSM relies on).

**Performance of MulDEF under Indirect Attack.**

Figures 5, 6, 7, and 8 show the adversarial accuracy of the target model $T$ (MulDEF with only one model – the target model) and the adversarial accuracy of MulDEF (with 2, 3, 4 and 5 models) on different sets of adversarial examples ($Adv_T, Adv_{M_1}, \ldots, Adv_{M_4}$ denoted as different colored bars). Those four figures lead to the same conclusion that MulDEF can successfully defend Indirect Attack: when using 5 models in MulDEF, the lowest adversarial accuracy in FGSM+MNIST, C&W+MNIST, FGSM+CIFAR-10, and C&W+CIFAR-10 are 64.12%, 64.34%, 49.99%, and 60.45% respectively, whereas the target model (without any defense) has 11.46%, 0.00%, 14.47%, and 0.00% adversarial accuracy for those setups.

**Performance of MulDEF under Direct Attack.**

Figure 9 shows the adversarial accuracy of MulDEF against C&W when we use different numbers of additional models for both MNIST and CIFAR-10 datasets. The figure shows that MulDEF with additional models is much more robust than the target model for both datasets as MulDEF achieves 63.95% and 49.44% adversarial accuracy for MNIST and CIFAR-10, respectively when using 5 models, whereas the target model (without any defense) has 0.00% adversarial accuracy for both datasets. Unexpectedly, the figure shows that the adversarial accuracy of MulDEF against C&W is decreasing when we add the fourth and the fifth models for
MNIST. We further investigate the effect of the number of additional models in later discussion.

In summary, MULDEF provides substantial defense for the target model against C&W in the white-box scenario under both Indirect Attack and Direct Attack. Furthermore, we are interested to see the comparison of adversarial accuracy between Indirect Attack (IndirectAtk) and Direct Attack (DirectAtk). We allow attackers to choose the better one from these two types of attacks, because the attackers have complete knowledge of all the models in MULDEF in the white-box scenario.

Table II shows the comparison of adversarial accuracy when MULDEF has 2, 3, 4, or 5 models on both datasets using C&W. For Indirect Attack, more than one group of adversarial
Fig. 9: DIRECT ATTACK adversarial accuracy of MULDEF against C&W with different numbers of multiple models for both MNIST and CIFAR-10.

| #Models | MNIST     | CIFAR-10  |
|---------|-----------|-----------|
|         | IndirectAtk | DirectAtk | IndirectAtk | DirectAtk |
| 2       | 36.86%     | 37.85%    | 36.51%      | 43.95%     |
| 3       | 51.74%     | 51.39%    | 50.94%      | 56.61%     |
| 4       | 57.91%     | 50.57%    | 56.24%      | 63.13%     |
| 5       | 64.34%     | 49.44%    | 60.45%      | 65.95%     |

TABLE II: Comparison of INDIRECT ATTACK and DIRECT ATTACK adversarial accuracy of MULDEF against C&W. Lower adversarial accuracy between INDIRECT ATTACK and DIRECT ATTACK is in bold, indicating a better attack.

examples are obtained. Therefore, we refer to the adversarial accuracy in INDIRECT ATTACK as the lowest adversarial accuracy among groups of adversarial examples.

The results show that the two types of white-box attacks have their own advantages. For MNIST, when we use 2 models in MULDEF, INDIRECT ATTACK performs better than DIRECT ATTACK (36.86% vs 37.85%). However, when the number of models grows from 3 to 5, DIRECT ATTACK outperforms INDIRECT ATTACK. For CIFAR-10, INDIRECT ATTACK always outperforms DIRECT ATTACK.

In later discussion, we refer to C&W white-box adversarial accuracy as the minimum of INDIRECT ATTACK and DIRECT ATTACK adversarial accuracy with C&W. Because FGSM white-box attack does not have DIRECT ATTACK adversarial accuracy, we refer to FGSM white-box adversarial accuracy as the INDIRECT ATTACK adversarial accuracy with FGSM.

B. PERFORMANCE OF MULDEF ON ORIGINAL TESTING DATASET

While achieving higher adversarial accuracy, MULDEF also maintains about the same test accuracy as the target model. Figure 10 confirms that diversity does not decrease much of test accuracy as each additional model has about the same test accuracy as the target model. We suspect that the quality of training set would be a major factor for the test accuracy of a classifier while model diversity plays only a minor role.

Overall, for MNIST, our defense against FGSM/C&W has 98.91%/98.79% test accuracy compared to 99.03%/98.58% (the target model’s test accuracy). For CIFAR-10, our defense against FGSM/C&W has 77.13%/80.05% test accuracy compared to 79.78%/80.41% (the target model’s test accuracy).

C. IMPACT OF DIFFERENT NUMBERS OF MODELS

According to Figures 5-8 and Table II, we can infer that having more models in MULDEF can increase adversarial accuracies. And surprisingly, each model can maintain about the same test accuracy as the target model’s. One may have the concern that when we generate more models, the last model
may have lower test accuracy than the others, because the last model is constructed by the most adversarial examples augmented to its training set. It turns out that those adversarial examples do not have a negative impact or confuse the model when the model faces against normal examples. This result suggests that increasing diversity of a classifier by including more models constructed by our MulDef approach could make the classifier more robust against adversarial examples.

In the preceding evaluations, we show the performance of MulDef with only 2 to 5 models. However, in the real defense scenario, we have to determine when to stop running the model generator in MulDef. Moreover, we hypothesize that the adversarial accuracy will converge at some point. To confirm this hypothesis, we additionally evaluate our approach when using 6, 7, 10 models.

Figure 11 shows the white-box adversarial accuracy of MulDef against FGSM and C&W on both datasets. The adversarial accuracy of FGSM+MNIST, FGSM+CIFAR-10, and C&W+CIFAR-10 converges when using 10 models. Furthermore, the adversarial accuracy of C&W+MNIST gradually declines after reaching the highest point (when using 3 models) and also converges when using 10 models.

From these results, we can confirm our hypothesis that the adversarial accuracy will converge. This conclusion suggests that the model generator in MulDef can stop constructing more models when the adversarial accuracy of MulDef converges.

D. Impact of Adversarial Training and Randomization in MulDef

Based on the idea of adversarial training, the model generator in MulDef constructs each additional model one after another, where $M_i$ is trained with the union of the original training set and $Adv_{M_1}, Adv_{M_2}, ..., Adv_{M_{i-1}}$. In other words, for $Adv_{M_i}$, all the other models constructed after $M_i$ are trained with the training set that includes $Adv_{M_i}$. Therefore, most models constructed after $M_i$ should be robust to $Adv_{M_i}$. To test this hypothesis, we measure the average accuracy of all the models constructed after $M_i$, denoted as the third accuracy in Table III. We can see that the third accuracy is the highest as expected except for the last column. The cells in the last column in Table III are missing the third accuracy, because $M_4$ is the last model in the family. In addition, in Figures 5, 6, 7, and 8 the impact of adversarial training also reflects that MulDef tends to be most vulnerable against the adversarial examples generated for the last model, because no model in the family is exposed to those adversarial examples.

To further demonstrate that our random strategy in runtime model selection performs well, we test two hypotheses: (I) most models constructed after $M_i$ achieve higher accuracy on $Adv_{M_i}$ than $M_i$, (II) most models constructed before $M_i$ achieve higher accuracy on $Adv_{M_i}$ than $M_i$. As Hypothesis I has already been demonstrated, we measure the average accuracy of all the models constructed before $M_i$, denoted as the second accuracy in Table III. We denote accuracy of $M_i$ on $Adv_{M_i}$ as the first accuracy. Our result shows that the second accuracy is higher than the first accuracy. This result raises a question on why most models constructed before $M_i$ can achieve higher accuracy on $Adv_{M_i}$, even though they are not exposed to $Adv_{M_i}$ at all during their training process. We hypothesize the reason to be that $Adv_{M_i}$ is very specific to $M_i$. In the process of constructing $M_i$ to $M_t$, every new model constructed is trained with more adversarial examples, causing the later constructed models to be different from previously constructed models. Although adversarial examples have the transferability property, they are not effective compared to the adversarial examples specifically generated for their own model. For example, Table IV shows MulDef against FGSM with 5 models for MNIST. Although no model in MulDef is exposed to the adversarial example for $M_5$, the other models ($T, M_1, M_2, M_3, M_4$) still achieve high accuracy on $Adv_{M_5}$. In addition, Tables VII, VIII and IX in the Appendix show the accuracy of each model in MulDef against FGSM for CIFAR-10 and against C&W for MNIST/CIFAR-10.

In summary, adversarial training helps construct the family of models that are complementary to each other, enabling our random selection used in the runtime model selection.

E. Cross-Attack Scenarios

In the preceding white-box attack scenarios, we use adversarial examples generated by FGSM to construct models in MulDef. Then we evaluate MulDef under FGSM white-box attack. We evaluate MulDef on C&W white-box attack similarly. Realistically, we cannot know in advance which white-box attack approach the attackers will use in reality, indicating that we cannot construct models in MulDef based on the adversarial examples generated by the approach to be used by the attacker. Therefore, we further investigate the robustness of our approach in cross-attack scenarios, in which we use C&W to attack MulDef built on top of FGSM adversarial examples and vice versa.

Table V shows the results. Compared with the adversarial accuracy of the target model, no matter which group of adversarial examples MulDef uses to construct models, MulDef performs better than the target model. Surprisingly, we find that when being attacked by C&W, MulDef built on FGSM adversarial examples is more robust than it built on C&W adversarial examples. We also notice that MulDef built on FGSM adversarial examples is more robust than it built on C&W adversarial examples in general. Note that it does not necessarily suggest that FGSM attack is more powerful than C&W but instead suggests that FGSM adversarial examples are more suitable than C&W ones to construct diversified models in MulDef.

F. Performance of MulDef in Black-box Attack Scenario

In the black-box attack scenario, we train a substitute model as discussed in Section V. Table VI shows that MulDef still achieves higher adversarial accuracy than the target model in the black-box attack scenario except when C&W and CIFAR-10 are used. Reasons that our defense does not substantially improve the target model in the black-box attack scenario might be the transferability property of adversarial examples or the fact that black-box attack approaches are not powerful enough to reveal weakness in the target model.

Table VI summarizes the performance of MulDef in both white-box and black-box attack scenarios. The results shown in the table suggest that our MulDef approach is effective in both black-box and white-box scenarios.
TABLE III: Three adversarial accuracies on a set of adversarial examples generated for each model in MULDEF with 5 models \((T, M_1, M_2, M_3, M_4)\) against an attack approach and a dataset. For the set of adversarial examples for model \(x\) \((Adv_x)\), the first accuracy denotes the accuracy of model \(x\) on \(Adv_x\), the second accuracy denotes the average accuracy of all the models constructed (by MULDEF) before model \(x\) on \(Adv_x\), the third accuracy denotes the average accuracy of all the models constructed (by MULDEF) after model \(x\) on \(Adv_x\). Note that MULDEF constructs \(M_1, M_2, M_3,\) and \(M_4\) in order. The highest accuracy among the three is in bold.

| Attack / dataset | \(Adv_T\) | \(Adv_{M_1}\) | \(Adv_{M_2}\) | \(Adv_{M_3}\) | \(Adv_{M_4}\) |
|------------------|-----------|----------------|----------------|----------------|----------------|
| FGSM / MNIST     | 11.46%    | 54.42%         | 63.59%         | 77.29%         | 81.50%         |
| FGSM / CIFAR-10  | 14.47%    | 61.08%         | 68.43%         | 72.82%         | 77.35%         |
| C&W / MNIST      | 00.00%    | 97.73%         | 96.73%         | 97.73%         | 97.73%         |
| C&W / CIFAR-10   | 00.00%    | 97.41%         | 77.38%         | 77.72%         | 77.72%         |

TABLE IV: Accuracy of each model in MULDEF against FGSM with 5 models on each adversarial example set for MNIST. The highest accuracy for each adversarial example set is marked in bold.

| Models | Accuracy on adversarial examples | \(Adv_T\) | \(Adv_{M_1}\) | \(Adv_{M_2}\) | \(Adv_{M_3}\) | \(Adv_{M_4}\) |
|--------|----------------------------------|-----------|----------------|----------------|----------------|----------------|
| \(T\)  | 11.46%                           | 54.42%    | 63.59%         | 77.29%         | 81.50%         |
| \(M_1\) | 96.66%                           | 61.08%    | 68.43%         | 72.82%         | 77.35%         |
| \(M_2\) | 95.36%                           | 96.52%    | 96.01%         | 97.73%         | 97.73%         |
| \(M_3\) | 94.62%                           | 95.69%    | 96.01%         | 97.73%         | 97.73%         |
| \(M_4\) | 94.94%                           | 95.59%    | 95.46%         | 96.48%         | 96.48%         |

TABLE V: Adversarial accuracies of MULDEF built on FGSM and C&W adversarial examples against FGSM and C&W white-box attack. Cross-attack results are marked in bold.

| Attack approaches | Tgt model | MULDEF | FGSM adv exps | C&W adv exps |
|-------------------|-----------|--------|---------------|--------------|
| FGSM              | MNIST     | 11.46% | 64.12%        | 14.96%       |
|                   | CIFAR-10  | 14.47% | 49.99%        | 26.47%       |
| C&W               | MNIST     | 00.00% | 69.19%        | 49.44%       |
|                   | CIFAR-10  | 00.00% | 65.28%        | 60.45%       |

TABLE VI: Comparison of adversarial accuracy of the target model and MULDEF in the white-box and black-box scenarios.

| Attack approaches | Black-box | White-box |
|-------------------|----------|-----------|
| FGSM              | MNIST    | 70.19%    | 81.76%    |
|                   | CIFAR-10 | 70.32%    | 72.82%    |
| C&W               | MNIST    | 72.10%    | 78.89%    |
|                   | CIFAR-10 | 70.71%    | 65.28%    |

VII. DISCUSSION AND FUTURE WORK

We have shown the effectiveness of MULDEF, which can substantially improve the robustness of the target model in both black-box and white-box scenarios. MULDEF is attack-dependent. It requires the security analysts to provide sample attack approaches as inputs to train additional models. Enabling MULDEF to be attack-independent would be a worthwhile direction for future work. Here we discuss how MULDEF can construct additional models if no attack approach is given. In particular, can we come up with universal adversarial examples, which can be used to train additional models so that our defense can defend against any kind of attack approaches? We have shown some results on cross-attack scenarios: surprisingly MULDEF built with FGSM adversarial examples can effectively defend against C&W. Moreover, MULDEF against C&W built with FGSM achieves higher adversarial accuracy than MULDEF against C&W built with C&W itself. One potential direction could be to build MULDEF with adversarial examples generated by a few different attack approaches, and to check whether MULDEF can substantially defend against any kinds of attack approaches.

MULDEF incrementally constructs additional models one by one. One may wonder when MULDEF should stop constructing more models. According to Figure 9, we hypothesize that the adversarial accuracy of MULDEF should keep increasing as it has more models, and then eventually stabilize. Thus, we can implement MULDEF in such a way that every time it constructs a new model, MULDEF measures its adversarial accuracy on the adversarial examples generated for the new model. If its adversarial accuracy does not change much compared to MULDEF without this new model, MULDEF stops.

Our evaluations including only two attacks and two datasets show that augmenting about 15%/25% (for MNIST/CIFAR-10) of the training set with adversarial examples seems to be enough. However, for some other attacks or datasets, we may need to augment more than 15% or 25%. Moreover, one needs to ensure that the test accuracy of each additional model does not decrease a lot when augmenting many adversarial examples to its training set. The step of choosing this percentage parameter may require tuning to achieve a satisfactory result.

According to our evaluation results, MULDEF slightly improves the target model’s adversarial accuracy by about 2–10% in the black-box attack scenario. We suspect the reason why we do get much improvement to be that the adversarial examples augmented to the training set to construct additional models are generated in the white-box attack scenario; however, we evaluate MULDEF in the black-box attack scenario. MULDEF may achieve higher adversarial accuracy in the black-box attack scenario if the adversarial examples for training additional models are generated in the same attack scenario.

We choose the random strategy for runtime model selection to achieve low cost and introduce uncertainty in the target model. Nevertheless, there can be another strategy based on multi-model/implementation majority voting [14, 23], which we plan to explore in future work. In particular, Srivasta et al. [14] use multiple-implementation testing to test an
implementation of a machine learning algorithm, where the majority output across multiple implementations of the same algorithm is used as a test oracle. MULDEF also contains multiple models that are robust to a given adversarial example, so we may be able to use the majority label across multiple models as an output, instead of randomly selecting one model to be applied on the given example at runtime. However, as discussed in Section I, such strategy can be costly and not scalable, needing to apply all models in the family to the given example. In addition, with a candidate adversarial example, the attackers can also simulate such strategy of multi-model majority voting ahead of time and know beforehand which model in the family to (re-)compute the gradient to improve the candidate adversarial example if needed.

VIII. RELATED WORK

Defense against Adversarial Examples. A variety of approaches have been proposed for defending against adversarial examples. Meng et al. [17] propose a defense approach named MagNet against adversarial examples. MagNet consists of two main steps: detect and reform. Like MULDEF, MagNet does not modify the target model. MagNet first detects whether a given input example is adversarial by measuring the distance between the input example and the manifold of normal examples in the training set. If the input example is farther to the manifold of the normal examples, the input example is marked as an adversarial example and then gets reformed/reconstructed to be close to normal examples. Finally, the example is passed to the target model to classify. Their ideas are quite different from our MULDEF approach, because MULDEF does not reform the given example. Instead, MULDEF trains more models to handle any example. MagNet does not require the knowledge of the attack approaches, and can perform well in the gray-box attack scenario (where the attackers know about the target model and the defense, but not the parameters of the defense), but cannot handle the white-box attack scenario at all. Another approach similar to MagNet is APE-GAN [24], which generates images similar to the given adversarial example by using Generative Adversarial Net (GAN) [25]. Both MagNet and APE-GAN still provide ineffective defense against C&W.

Song et al. [26] propose an approach of image purification named PixelDefend to defend against adversarial examples. PixelDefend requires no knowledge of the attack nor the target model, but uses the PixelCNN model for its state-of-the-art performance in modeling image distributions [27] to detect an adversarial example. Then PixelDefend purifies the adversarial example by searching for more probable images within a small distance of the adversarial example. Because PixelDefend only purifies the given adversarial example, we can combine this approach with MULDEF to proceed with the purified example.

Tramer et al. [28] propose another version of adversarial training, named Ensemble Adversarial Training, which augments the training set with adversarial examples crafted from pre-trained models. Our model generator also constructs each additional model with the original training set augmented with adversarial examples generated from other previously constructed models in the family. However, all the models in the family have the same architecture and parameter settings.

Santhanam et al. [29] propose an attack-independent defense approach, similar to both MagNet and APE-GAN. The main idea is based on using GAN to project the input examples back onto the data manifold. Their results show that their approach is effective across different attacks and datasets; however, their evaluation does not include C&W.

Zantedeschi et al. [30] propose an approach to make the target model more robust against adversarial examples by reinforcing the model architecture so that its prediction becomes more stable. Their approach uses the Bounded ReLU activation function for hedging against the forward propagation of adversarial perturbation and Gaussian data augmentation during training. Their approach is mainly for making the attack visually detectable. However, it still does not perform well against C&W.

Improving Machine Learning Models. Our work falls into the general research domain of improving machine learning models. DeepXplore [23] and DeepTest [31] introduce and leverage the metric of neuron coverage for a neural network with rectified linear units (ReLus) as the activation functions. Neuron coverage measures the percentage of hidden units that can have positive value (for at least one of the test inputs) in the neural network. DeepGauge [32] further generalizes neuron coverage by dividing the range of values of each neuron (obtained during training) into $k$ chunks, and measures whether each of the $k$ chunks can be covered by the test cases. DeepGauge also measures whether each activation has been made to go above and below a certain bound. In addition to neuron coverage, a feature-guided approach [33] and concolic execution [34] approach are also proposed to perform black box testing of image classifiers using image-specific operations.

Applications of Adversarial Machine Learning. Adversarial machine learning has a wide application in other areas of security (e.g., malware detection). MalGAN [35] leverages the transferability property [36,38] to train a substitute model and use such gradient computation in a modified GAN to produce evasive malware variants. GADGET [39] also leverages the transferability property to attack against dynamic machine learning models that leverage recursive neural networks as a basis. The model under attack is known to use sequences of API calls (as observed in a sandbox, for example) to determine whether a sample is malicious. Both MalGAN and GADGET require the attackers to have the knowledge of the complete feature space of the target model, severely limiting the application of the approaches. EvadeHC [40] produces evasive PDF malware relying only on final classification decision (e.g., reject or accept an input sample). The idea is measure the number of morphing steps required to change the detector’s label to reflect evasion progress for a given sample. EvadeHC algorithmically models the evasion progress via a hill-climbing approach. MalConv [41] is proposed to produce evasive malware binaries. MalConv considers only the manipulation of padding bytes appended at the end of the file. Malware detectors can leverage MULDEF to easily defend such an attack by treating the padding bytes as continuous feature values (same as image pixels) in the features set.

IX. CONCLUSION

In this paper, based on the design principle of diversity, we have proposed MULDEF, a defense approach against adversarial examples for neural networks. MULDEF first constructs
a family of models (from the given target model) such that the models are complementary to each other to accomplish robustness diversity. Our approach is based on the insights that many attack approaches rely on the target model to generate adversarial examples that work well specifically on the target model, and are not intentionally built to attack multiple models at the same time. MULDEF is simple, scalable, and easy to be applied, because it does not modify the target model. We evaluate our approach on two attack strategies (FGSM and C&W) for two datasets (MNIST and CIFAR-10). The evaluation results show that our defense approach substantially improves the target model’s adversarial accuracy by 35–50% in the white-box attack scenario against both attack strategies and both datasets. For the black-box attack scenario, our defense approach also improves the target model’s adversarial accuracy against FGSM by 2–10% and against C&W by around 5%. While making the target model more robust against adversarial examples, MULDEF still maintains about the same accuracy as the target model on normal examples.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. CVPR, 2016, pp. 770–778.
[2] D. W. Nicholas Carlini, “Towards evaluating the robustness of neural networks,” in Proc. Security and Privacy (SP), 2017, pp. 39–57.
[3] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in Proc. ASIA CCS, 2017, pp. 506–519.
[4] N. Narodytska and S. P. Kasiviswanathan, “Simple black-box adversarial perturbations for deep networks,” in CVPRW, 2017, pp. 1310–1318.
[5] N. Papernot, N. Carlini, I. Goodfellow, R. Fennman, F. Faghi, A. Matysko, K. Hambardzumyan, Y.-L. Juang, A. Kurakin, R. Sheatsley, A. Garg, and Y.-C. Lin, “cleverhans v2.0.0: an adversarial machine learning library,” arXiv preprint arXiv:1610.00768, 2017.
[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.
[7] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in Proc. SP, 2016, pp. 582–597.
[8] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in Proc. EuroSP, 2016, pp. 372–387.
[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” CoRR, vol. abs/1312.6199, 2013. [Online]. Available: http://arxiv.org/abs/1312.6199
[10] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples,” CoRR, vol. abs/1605.07277, 2016. [Online]. Available: http://arxiv.org/abs/1605.07277
[11] N. Carlini and D. A. Wagner, “Defensive distillation is not robust to adversarial examples,” CoRR, vol. abs/1607.04311, 2016. [Online]. Available: http://arxiv.org/abs/1607.04311
[12] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz, “SoK: Automated software diversity,” in SP. Washington, DC, USA: IEEE Computer Society, 2014, pp. 276–291. [Online]. Available: https://doi.org/10.1109/SP.2014.25
[13] K. Pei, Y. Cao, J. Yang, and S. Jana, “DeepXplore: Automated whitebox testing of deep learning systems,” in Proc. SOSP, 2017, pp. 1–18.
[14] S. Sriskakaokul, Z. Wu, A. Astorga, O. Alebiouso, and T. Xie, “Multiple-Implementation Testing of Supervised Learning Software,” in Proc. EDSMLS, 2018.
[15] Y. LeCun, C. Cortes, and C. J. Burges. (1998) The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
[16] A. Krizhevsky, V. Nair, and G. Hinton. (2009) The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html
[17] D. Meng and H. Chen, “MagNet: A two-pronged defense against adversarial examples,” in Proc. CCS, 2017, pp. 135–147.
[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in Proc. ICLR, 2013.
[19] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.
[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in Proc. ICLR, 2017.
[21] Google. (2018) Tensorflow. https://www.tensorflow.org/
[22] F. Chollet et al., “Keras,” https://keras.io 2015.
[23] K. Pei, Y. Cao, J. Yang, and S. Jana, “DeepXplore: Automated whitebox testing of deep learning systems,” in Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017, pp. 1–18.
[24] S. Shen, G. Jin, K. Gao, and Y. Zhang, “AE-GAN: adversarial eliminating with GAN,” CoRR, vol. abs/1705.05474, 2017. [Online]. Available: http://arxiv.org/abs/1705.05474
[25] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” in Proc. NIPS, 2014, pp. 2672–2680.
[26] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, “Pixeldefend: Leveraging generative models to understand and defend against adversarial examples,” in Proc. ICLR, 2018. [Online]. Available: https://openreview.net/forum?id=UUYGkxCW
[27] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, “Conditional image generation with PixelCNN decoders,” in Proc. NIPS, 2016, pp. 4797–4805. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157382.3157633
[28] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=rykZvSe-RZ
[29] G. Krishnan Santhanam and P. Gnararova, “Defending Against Adversarial Attacks by Leveraging an Entire GAN,” ArXiv e-prints, May 2018.
[30] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, “Efficient defenses against adversarial attacks,” in Proc. AISec, 2017.
[31] Y. Tian, K. Pei, S. Jana, and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in Proceedings of the 40th International Conference on Software Engineering. ACM, 2018, pp. 303–314.
[32] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu et al., “Deepgauche: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems,” arXiv preprint arXiv:1803.07519, 2018.
[33] M. Wicker, X. Huang, and M. Kwiatkowska, “Feature-guided black-box safety testing of deep neural networks,” in International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2018, pp. 408–426.
[34] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, “Concolic testing for deep neural networks,” arXiv preprint arXiv:1805.00089, 2018.
[35] W. Hu and Y. Tan, “Generating adversarial malware examples for black-box attacks based on gan,” arXiv preprint arXiv:1702.05983, 2017.
[36] 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.
[37] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” in Proc. ICLR, 2017.
[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” arXiv preprint arXiv:1312.6199, 2013.
[39] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, “Generic black-box end-to-end attack against nnns and other api calls based malware classifiers,” arXiv preprint arXiv:1701.05970, 2017.
[40] H. Dang, Y. Huang, and E.-C. Chang, “Evaerﬁng classifiers by morphing
in the dark,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 119–133.

[41] B. Kolosnjaji, A. Demonis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, “Adversarial malware binaries: Evading deep learning for malware detection in executables,” arXiv preprint arXiv:1803.04173, 2018.

APPENDIX

| Models | Accuracy on adversarial examples |
|--------|----------------------------------|
|        | AdvT | AdvM1 | AdvM2 | AdvM3 | AdvM4 |
| T      | 14.47% | 57.10% | 52.89% | 50.98% | 49.91% |
| M1     | 65.22% | 14.62% | 73.14% | 60.53% | 58.04% |
| M2     | 58.30% | 72.82% | 13.71% | 67.72% | 57.01% |
| M3     | 59.17% | 60.97% | 71.07% | 19.27% | 64.86% |
| M4     | 61.64% | 62.10% | 65.79% | 68.46% | 20.85% |

TABLE VII: Accuracy of each model in MULDEF against FGSM with 5 models on each adversarial example set for CIFAR-10. The highest accuracy for each adversarial example set is marked in bold.

| Models | Accuracy on adversarial examples |
|--------|----------------------------------|
|        | AdvT | AdvM1 | AdvM2 | AdvM3 | AdvM4 |
| T      | 00.00% | 74.95% | 62.48% | 59.13% | 66.71% |
| M1     | 97.10% | 00.00% | 90.94% | 76.51% | 79.54% |
| M2     | 97.51% | 96.70% | 00.00% | 91.81% | 86.67% |
| M3     | 97.94% | 96.80% | 96.63% | 00.00% | 93.94% |
| M4     | 98.37% | 97.58% | 96.83% | 96.50% | 00.00% |

TABLE VIII: Accuracy of each model in MULDEF against C&W with 5 models on each adversarial example set for MNIST. The highest accuracy for each adversarial example set is marked in bold.

| Models | Accuracy on adversarial examples |
|--------|----------------------------------|
|        | AdvT | AdvM1 | AdvM2 | AdvM3 | AdvM4 |
| T      | 00.00% | 72.43% | 74.90% | 74.99% | 76.16% |
| M1     | 75.12% | 00.00% | 74.63% | 74.25% | 74.77% |
| M2     | 78.00% | 77.57% | 00.00% | 77.70% | 77.88% |
| M3     | 77.30% | 76.21% | 76.84% | 00.00% | 76.95% |
| M4     | 79.24% | 78.29% | 78.62% | 78.60% | 00.02% |

TABLE IX: Accuracy of each model in MULDEF against C&W with 5 models on each adversarial example set for CIFAR-10. The highest accuracy for each adversarial example set is marked in bold.

Tables VII, VIII, and IX (along with Table [IV]) show the accuracy of each model in MULDEF under different settings. For FGSM, each of the four models, M1, M2, M3, M4, is robust on different sets of adversarial examples. Their diversity provides robustness of our overall defense approach. For C&W, the last model (M4), which is trained with all the adversarial examples generated for the other models, is the most robust against AdvT, AdvM1, AdvM2, AdvM3. However, MULDEF still needs M3 to help defend the adversarial examples for M4 (AdvM4).