MRobust: A Method for Robustness against Adversarial Attacks on Deep Neural Networks

Yi-Ling Liu
Imperial College London, United Kingdom

Alessio Lomuscio
Imperial College London, United Kingdom

***Abstract*—We present a novel black-box adversarial training algorithm to defend against state-of-the-art attack methods in machine learning. In order to search for an adversarial attack, the algorithm analyses small regions around the input that are likely to make significant contributions for the generation of adversarial samples. Unlike some of the literature in the area, the proposed method does not require access to the internal layers of the model and is therefore applicable to applications such as security. We report the experimental results obtained on models of different sizes built for the MNIST and CIFAR10 datasets. The results suggest that known attacks on the resulting models are less transferable than those models trained by state-of-the art attack algorithms.**

I. Introduction

Conventional machine learning models are known to be vulnerable to adversarial examples [1]–[4]. A gradient-based approach to generate adversarial examples for linear classifiers, support vector machines (SVM), and neural networks in the context of MNIST models was first developed in [5]. This was then extended to proactive and reactive defences to improve the security of machine learning models [6]. It has also been observed that adversarial examples may have imperceptible differences compared to the original input [7]. [8] suggested that adversarial examples are inherently caused by the linear behaviour of deep neural networks (DNNs) when operating in high-dimensional spaces. The topology and geometry of adversarial examples were analysed in [9], while the local intrinsic dimensionality of adversarial regions for adversarial examples were characterised in [10]. More recently, it has been observed that adversarial examples are transferable, i.e., an example which is adversarial for a DNN can often be used to mislead the prediction of other DNNs [7], [11], [12]. Moreover, adversarial examples could be universal in the sense that a single example may be used against several different models created from the same dataset [13].

Adversarial examples on DNNs have raised serious security issues as DNNs have become increasingly popular in applications such as computer vision systems for autonomous vehicles, face recognition software, and malware detection [14]–[16]. Several defence methods against adversarial attacks have been proposed [17]–[19]; but these were broken shortly [20]–[22]. More powerful defence techniques are required to make DNNs robust and usable in safety-critical applications. This paper makes a contribution towards this aim.

To increase the models' robustness against attacks, several adversarial training methods have been developed. [8] first injected adversarial examples during the training stage and demonstrated that in this way the robustness of DNNs can be increased significantly. However, this comes at a high cost as the resistance to attacks increases only after large numbers of adversarial examples have been added to the training data. This raises the key research question of learning models that are robust to adversarial examples in a computationally effective manner.

In this research, we propose an adversarial attack method to generate adversarial examples for adversarial training. Given an arbitrary input to a DNN, the algorithm here proposed searches small regions around the input that have significant potential to generate adversarial samples. The algorithm does not require access to the internal layers of the DNN and thus falls in the realm of black-box adversarial attack. Similarly to other approaches, these attacks are then used for adversarial training. By comparing against FGSM [8] and PGD [21], two state-of-the-art methods in adversarial training, our results show that the resulting DNNs synthesised via our method are less susceptible to attack transferability. We also show that the method reduces significantly the number of adversarial examples required for adversarial training.

The remainder of this paper is organised as follows. Section II introduces some preliminaries regarding adversarial examples and their formulation under different attacks on DNNs. Section III describes the main black-box algorithm here proposed aimed at generating adversarial examples and how adversarial examples are employed for adversarial training. Section IV defines robustness transferability and reports quantitative results on the MNIST and CIFAR10 datasets; Section V concludes the paper.

Related work. State-of-the-art defence methods can be categorised into two groups. The first is commonly referred to as approaches dealing with reactive countermeasures. These defend the model after an attack has been performed. They attempt to detect adversarial examples in the inputs. These include methods of adversarial detecting [23] and input reconstruction [24]. The second consists of proactive methods which aim to train robust DNNs before attacks are performed on them. These include network distillation [17] and adversarial training [25]. Although all these countermeasures improve the robustness of DNNs to some level, they still have significant limitations. This paper overcomes some of them as we explain in the following.

Subnetworks to detect adversarial examples for a given net-

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

work were explored in [23]. This deep subnetwork is trained to distinguish genuine data from data containing adversarial perturbations. Similar ideas are employed in approaches aimed at detecting adversarial examples in the testing stage [26]–[29]. For instance, SafetyNet [29] extracts each ReLU layer's output as the features for the adversarial detector and detects adversarial examples using the RBF-SVM algorithm. In [28], adversarial examples are shown to have different coefficients in low-ranked components after whitening by Principal Component Analysis (PCA). While all of these methods can help to mitigate against some attacks, [30] demonstrate that most of these adversarial detecting methods, including [23], [26]–[28], cannot defend against the C&W attack that they develop. In terms of adversarial training, PGD [21] is presently the most performing method and was shown to be more powerful than C&W. When tested against it, the method we present achieves higher accuracies than PGD.

A related concept in the literature is the one of input reconstruction which aims to transform adversarial examples to clean data and then using these to assist DNNs in predicting correct classifications via denoising autoencoders (DAEs). A variant of DAEs, called deep contractive networks (DCNs), is proposed to increase the robustness of neural networks without a significant performance penalty [24]. However, calculating partial derivatives at each layer in the back-propagation stage has a high computational cost. In addition, the proposed layered based approach does not guarantee global optimality. Compared to this approach, at least for the datasets that we experimented with, our method does not suffer from local optimality problems.

[17] proposed network distillation, which was originally designed to reduce the size of DNNs whilst maintaining classification characteristics, for defence purposes. However, network distillation was unable to defend against the PGD attack developed in [21]. Reducing the size of the model may also have implications on the overall accuracy. In contrast, we do not suffer from any accuracy degradation and we show that the approach is resilient against the PGD attack. In summary, our contribution is intended to improve on the state of the art by providing a technique that is comparably computationally attractive, can defend against the C&W and PGD attacks, and has good accuracy, and does not appear to be susceptible to local optima.

II. Preliminaries

In this section we define some background notions used in the rest of the paper. Specifically, in Subsection II-A we formalise deep neural networks. In Subsection II-B we introduce the problem of finding adversary symbols formally. In Subsection II-C we introduce two state-of-the-art adversarial attack methods, which are also used later in experimental comparisons. Lastly, in Subsection II-D, we give some details of adversarial training strategy.

A. Deep Neural Networks

We focus on the most common architecture of k-layer fully-connected feed-forward neural network with ReLU activation function. For an input vector x_t, each hidden layer transforms its input vector from the previous layer to the next layer by applying an affine transform as follows:

$$\begin{aligned} z^0 &= x_t, \\ a_i^{(l+1)} &= \sum_{j=1}^{N^{(l)}} w_{ij}^{(l)} z_j^{(l)} + b_i^{(l)}, \\ z_i^{(l+1)} &= \sigma(a_i^{(l+1)}), \end{aligned} \tag{1}$$

where $N^{(l)}$, $W^{(l)}$ and $b^{(l)}$ are the number of nodes, a weight matrix, and a bias vector of the l-th layer, respectively, and $\sigma(a_i^{(l+1)})$ is a nonlinear ReLU activation function. The Rectified Linear Unit (ReLU) is defined as $ReLU(x) = max(0, x)$, where the output is the maximum between 0 and the input positive value. In the last layer, the $softmax$ function is used to obtain the probability of the class y_t, which is formulated as:

$$softmax(x_t) = \frac{exp(w_{y_t} a^L)}{\sum_{n=1}^{N^{(L)}} exp(w_n a^{(L)})}, \tag{2}$$

where L is the number of hidden layers and $N^{(L)}$ is the number of output units. In summary, a DNN model θ is defined by the set of weight matrices W, bias vectors b, and a nonlinear activation function $\sigma(x)$ as follows:

$$\theta = \{W, b, \sigma(x)\}, \tag{3}$$

where $W = \{W^{(1)}, ..., W^{(L)}\}$ and $b = \{b^{(1)}, ..., b^{(L)}\}$. We refer to [31], [32] for more details of DNNs.

B. Adversarial Symbols

We denote a labelled training dataset of size n with $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is the input data in d dimensions and $y_i \in \{1, ..., k\}$ is a label in k classes. Given a deep neural network $\mathcal{N}$ with associated function $f(\cdot) : \mathbb{R}^d \rightarrow y_i$, the loss function of the network with parameters θ on (x_i, y_i) is expressed as $L(x_i, \theta, y_i)$. The gradient of the associated function $f(\cdot)$ with respect to the vector x_i is denoted as $\bigtriangledown_x f(\cdot)$.

Definition II.1. Let δ be a perturbation for a given input x_i. The perturbation between the original input data x_i and the adversarial example x_i' is represented as $\delta = x_i' - x_i \in \mathbb{R}^d$.

Definition II.2. Let ℓ be a norm of some p, where:

- for $p \in \mathbb{R}$ and $1 \leq p < \infty$, the ℓ_p norm of x_i is defined as $\|x_i\|_p = (\sum_{j=1}^d |x_i(j)|^p)^{1/p}$;
- for $p \in \infty$, the ℓ_∞ norm of x_i is defined as $\|x_i\|_\infty = max_j |x_i(j)|$.

Given a trained neural network $\mathcal{N}$ and an input data sample x_i, generating an adversarial example $x_i' \in \mathbb{R}^d$ with a label

$y'_i \in \{1, ..., k\}$ under some norm p can be formulated as a box-constrained optimisation problem:

$$\min \quad \|\delta\|_p \quad s.t. \quad f(x') \neq f(x), \quad x' \in [0, 1], \tag{4}$$

where $\|\cdot\|_p$ denotes the distance between two data samples under some norm p. This optimisation problem emphases on minimising the perturbation δ while misclassifying the target prediction given a constrained input data.

We here use another definition for adversarial example by minimising the loss value of the target class, where the perturbation δ is constrained in a subtle range ϵ. This is formulated as:

$$\min_{x'} \quad L(f(x'), f(x)) \quad s.t. \quad \|\delta\|_p \leq \epsilon, \quad f(x') \neq f(x), \tag{5}$$

where ϵ denotes the value of the perturbation constraint.

C. Adversarial Attack Methods

In this subsection we illustrate two (attack) methods for generating adversarial examples. We will evaluate the robustness and transferability of our proposed method against these two attack methods in the experimental Section IV.

Fast Gradient Sign Methods (FGSM) [8]. The method involves searching for adversarial attacks by performing one step gradient update along the direction of the sign of the gradient at each pixel of the input data. Specifically, given an adversarial input $x' = x + \delta$, where x is original input image and δ is the perturbation, the loss function of the model for adversarial input can be express as Equation (6):

$$L(x', \theta, y') = L(x, \theta, y) + (x' - x)\nabla_x \, L(x, \theta, y), \tag{6}$$

where θ is the hyperparameters of a model. Through minimising $L(x', \theta, y')$ subjecting to $\|x' - x\|_\infty \leq \epsilon$, the required perturbation is derived as Equation (7):

$$\delta = \epsilon \cdot sign(\nabla_x \, L_\theta(x)), \tag{7}$$

where ϵ is the magnitude of the perturbation constraint. By increasing ϵ the likelihood of x' being misclassified by the classifier $f(\cdot)$ increases; by increasing ϵ the attack may be more easily detected by humans.

Projected Gradient Descent (PGD) [21]. The PGD attack is widely believed to be of the most powerful attack methods. This method adopts the multi-step variant FGSM, i.e., projected gradient descent (PGD) on the negative loss function [25]:

$$x'_{t+1} = Clip_{x+\delta}(x'_t + \alpha \cdot sign(\nabla_x \, L(x', \theta, y))), \tag{8}$$

where α is the variant step size at step t and the $Clip$ function ensures that the output falls in the valid input value. PGD iteratively re-starts from many points in the ℓ_∞ balls around data points from the respective evaluation sets.

Algorithm 1: Black-box Adversarial Attack: MATTACK

1 **function** BLACK-BOXADVERSARIALATTACK ;
  Input : Clean image dataset x
  Initialise perturbation constraint setting for ϵ
  Initialise search trees T
  Output: Effective adversarial examples x'
2 Execute Scale Invariant Feature Transform to obtain potential candidate descriptor;
3 **while** *effective adversarial example not found* **do**
4 Pick up one most promising point as root node N_R from the descriptor;
5 Trace from the root node;
6 Selection: select nodes with greatest confidence value and trace till the leaf node;
7 Expansion: expand one node from leaf node;
8 Simulation: do simulation with the expanded node and check win or loss;
9 Backpropagation: update associated information for each node along the traversing path back to the root node
10 **end**

D. Adversarial Training

Adversarial training [8] is the process of explicitly training a model with adversarial examples to enhance model robustness. First, adversarial examples are generated in every step of the training stage and then they are injected into the training dataset. This method significantly contributes to robustness, as it can provide regularisation and improve precisions for DNNs under adversarial attacks.

In this training process, the initial setting for neural network is a random uniform distribution. The training process starts from choosing a minibatch set B of size m from training dataset and then generating k adversarial examples $\{x'_1, ..., x'_k\}$ from corresponding clean images $\{x_1, ..., x_k\}$ using current state of network $\mathcal{N}$. The new minibatch B' is expressed as $\{x'_1, ..., x'_k, x_{k+1}, x_m\}$. One training step of network $\mathcal{N}$ is implemented with the new minibatch B' and the previous steps are repeated until the training loss is converged.

III. THE MROBUST DEFENCE METHOD

In this section we present a black-box adversarial attack method relaying on the Scale Invariant Feature Transform (SIFT) algorithm [33]. We present how this algorithm uses Monte-Carlo tree search (MCTS) [34] to generate effective adversarial examples in Subsection III-A. In Subsection III-B we summarise the full black-box adversarial training algorithm, called MROBUST, including a robust optimisation function.

A. Black-box Adversarial Attack Method

In this subsection we introduce Scale Invariant Feature Transform [33] to search potential invariant feature candidates and Monte Carlo Tree Search [34] to find an adversarial example based on the outcomes of Scale Invariant Feature Transform. We summarise the full algorithm in Algorithm 1.

Scale Invariant Feature Transform [33]. The algorithm begins by executing SIFT on the clean image, given a perturbation ϵ, received as input (line 2). SIFT was first proposed

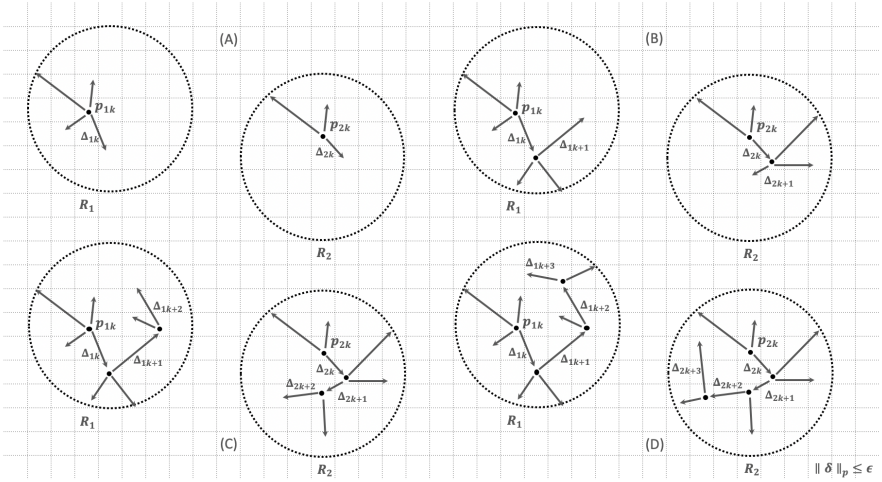

Fig. 1: An Effective Adversarial example in a region R_i with $\|\delta\|_p \leq \epsilon$.

to extract image features in object recognition systems [33]. It can be employed to extract features for any object in an image and provide a feature description of this object. This description can then be used to identify and locate objects in other images. In order to perform accurate recognition, these extracted features should be detectable under variations of image scale, orientation and illumination, and these extracted features are also invariant to image scaling, translation and rotation.

For the purpose of searching for features, Laplacian of Gaussian (LoG) are normally used to detect areas of rapid changes (edges) in images. Due to the significant computation cost of LoG operations, the Difference of Gaussians (DoG) is instead employed for simplification. Once the DoG is obtained, each pixel in the DoG is compared with its eight neighbours at the same scale and the process is then repeated at different scales. If a local maximum is identified, this is considered as a potential invariant feature. The points that are low-contrast and those that are edge response points are discarded. By discarding these points we are left with potential candidates for feature invariance over the parameters above. These points are collected into a descriptor also containing the location and orientation of these points.

Monte Carlo Tree Search (MCTS) [34].

We now explain how Monte Carlo Tree Search [34] can be employed to identify potential adversary symbols by searching in the neighbourhood of the candidates obtained by SIFT as above (line 3-10). MCTS is a heuristic search algorithm for decision making; its key feature consists of selecting the most promising moves on the basis of random sampling of the search space. In each iteration, MCTS consists of four steps, that are selection (line 6), expansion (line 7), simulation (line 8) and backpropagation (line 9). We consider each node of the tree as a specific pixel in an image. We first traverse from the root node N_R and choose a child node N_C with the highest confidence value down to a leaf node N_L. Then we expand one or more children nodes N_E and simulate from one of them to get a win or loss. A win of the game represents the fact that an adversarial example can be found once a perturbation is applied to this pixel. In the last step we update the tree structure with the new confidence information for each node according to the simulation result whether the search was a win or a loss.

We use the confidence value C_i, representing the associated information in each node N_C of the game tree, given by Equation (9):

$$C_i = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}, \tag{9}$$

where w_i stands for the inversion of Euclidean distances for the node considered after the i-th move, n_i stands for the number of visits for the node considered after the i-th move, and N_i stands for the total number of visits after the i-th move. The exploration parameter c is theoretically equal to $\sqrt{2}$.

Intuitively the higher probability to derive an adversarial symbol have the higher confidence value for each node. On the contrary, the lower probability to obtain an adversarial one will have a lower confidence value. The confidence measure above is inspired from the game of Go [35] where is widely used to make decisions in different games and applications for deep neural networks.

Given the above, we select one effective adversary symbol from the candidates above by imposing the constraint $\|\delta\|_p \leq \epsilon$. For each region R_i in an image, each path of the game tree can be considered as a ladder-like structure. Each region R_i contains several nodes $N_{C_{ik}}$ of the game tree and a specific perturbation Δ_{ik} is applied after a node $N_{C_{ik}}$ is selected. The total perturbations applied to a region is then presented as $\sum_{k=0}^{K} \Delta_{ik}$, where K is the total number of nodes along a path. An effective adversarial example $\mathcal{E}_A$ with $\|\delta\|_p \leq \epsilon$ is then formulated as Equation (10):

$$\mathcal{E}_A = \| \sum_{i=0}^{R} \sum_{k=0}^{K} \Delta_{ik} \|_p \leq \epsilon, \tag{10}$$

Algorithm 2: Black-box Adversarial Training: MROBUST

Deep neural network M

Size of the training minibatch is m

function BLACK-BOXADVERSARIALTRAINING ;
  Input : Deep neural network M
    Training dataset x
    Size of the training minibatch is m
  Output: Deep neural network MROBUST
    Adversarial training accuracy and loss values
Randomly initialize deep neural network M
while *training not converged* **do**
  Read minibatch $B = \{x_1, ..., x_m\}$ from training dataset;
  Generate m adversarial examples $B_{adv} = \{x'_1, ..., x'_m\}$ from corresponding clean examples $\{x_1, ..., x_m\}$ with black-box adversarial attack method in Algorithm 1:
    function BLACK-BOXADVERSARIALATTACK;
  Do one training step of network M using minibatch B';
  Update model loss with robust optimisation: $\min_\theta \mathcal{L} = \min_\theta \sum_{i=1}^{m} \max_\delta L(x'_i, \theta, y_i)$;
end

where R is the total number of regions in an image.

We explain more details in Figure 1 (A-D). In this figure, there are two regions $R_{i=1..2}$ and each region R_i contains several nodes $N_{C_{ik}}$ of the game tree. For region R_1, a specific perturbation Δ_{1k} is applied after one node $N_{C_{1k}}$ is selected. The total perturbations applied in a region R_1 is then presented as $\sum_{k=0}^{K} \Delta_{1k}$. An effective adversarial example $\mathcal{E}_A$ with constraint $\|\delta\|_p \leq \epsilon$ is obtained once all perturbations in these two regions satisfied with the constraint.

B. Black-box Adversarial Training Algorithm: MROBUST

In the previous subsection we obtained effective adversarial examples by using the black-box adversarial attack method. In this subsection we introduce a black-box adversarial training algorithm relaying on these effective adversarial examples. We summarise the full algorithm in Algorithm 2, including the robust optimisation step.

We now explain Algorithm 2. We first randomly initialised the neural network M. The loss value $\mathcal{L}$ of the neural network M is updated according to the softmax result of each adversarial training iteration. The training loop continues for as many epochs as required until the required accuracy is converged. During each training loop, we randomly select a minibatch B of size m from the training dataset, and generate corresponding minibatch B_{adv} consisted of size m adversarial examples using the black-box adversarial attack method. The minibatch B_{adv} is then applied into the robust model M for adversarial training. The convergence criteria ensures that resulting model M is robust in small neighbourhoods of every training point around x. We call these neighbourhoods the perturbations δ and we represent them as $x' = x + \delta$. The overall process can be regarded as a solution to the robust optimisation problem against adversarial examples and formulated as:

$$\min_\theta \mathcal{L} = \min_\theta \sum_{i=1}^{m} \max_{\|\delta\| \leq \epsilon} L(x'_i, \theta, y_i), \tag{11}$$

where δ is the perturbation set under the constraint ϵ corresponding to the adversarial example x'_i. This involves optimising the model parameter θ with respect to a worst-case data (x'_i, y_i), rather than against the original training data, which is related to the black-box attack method previously; the i-th worst-case data point is selected from the perturbation set δ.

IV. EXPERIMENTAL RESULTS

In this section we evaluate the robustness of the black-box adversarial training algorithm (MROBUST) presented in the previous section, and report the results obtained on the MNIST [36] and CIFAR-10 [37] datasets. We evaluate the transferability between different adversarial training models using FGSM, PGD, and MROBUST methods. The evaluation basis for transferability is defined in Subsection IV-A and the experimental setup and results are shown in Subsections IV-B, IV-C and IV-D.

A. Robustness Transferability

Attacks transferability is problematic in applications [11] as attacks identified in one domain may be easily transferable to another. Transferability can be analysed in terms of intra-technique and cross-technique transferability. Intra-technique transferability concerns the misclassifications (caused by a set of attacks) on different models trained on the same learning method. Cross-technique transferability concerns misclassifications (caused by a set of attacks) on models trained on different learning methods. Here we focus on cross-technique transferability against attacks on models built with FGSM [8], PGD [21], and MROBUST methods. Specifically, we use the black box adversarial attack method of the previous section as an adversary and evaluate the robustness of different adversarially trained models against this adversary. We also study the robustness of different model architectures and evaluate how the capacity of the network impacts for transferability.

In the following we focus on robustness transferability, defined as $RT = 1 - attack\_success\_rate$, which measures the percentage of adversarial samples produced using the black-box attack adversary that do not cause a misclassification on the trained model. In other words, a higher robustness transferability represents a situation in which the trained model is more robust under the attack of the black-box attack adversary.

B. Experimental Setup

We evaluated the method on the MNIST and CIFAR10 datasets. The MNIST database of handwritten digits contains a training set of 60,000 examples, and a test set of 10,000 examples. The digits were size-normalized and centred in a fixed-size image of 28×28. We generated adversarial examples under the perturbation constraints of size $\epsilon = 0.1$ in the l_∞ norm. To investigate model capacity, we considered two training networks of simple and wide architectures, respectively. The simple network consisted of two convolution layers of sizes 32 and 64 filters, and a fully connected layer of size 1024. The wide network consisted of two convolution layers of

Target \ Source	S. (Nature Training)	S. (FGSM Training)	S. (PGD Training)	S. (M Training)	W. (Nature Training)	W. (FGSM Training)	W. (PGD Training)	W. (M Training)
S. (Nature Training)	12.3	85.6	85.8	86.2	6.4	78.4	86.6	87.3
S. (FGSM Training)	78.3	64.3	68.4	74.7	76.4	64.8	76.2	82.1
S. (PGD Training)	80.2	78.4	81.4	80.1	79.2	76.5	80.8	82.7
S. (M Training)	83.8	84.2	84.7	74.9	83.2	84.5	86.8	79.7
W. (Nature Training)	15.7	89.7	88.6	90.1	5.2	77.2	85.3	90.7
W. (FGSM Training)	79.2	72.7	79.6	82.6	70.5	64.2	81.2	84.8
W. (PGD Training)	81.4	80.1	82.7	83.5	81.4	79.5	82.4	82.9
W. (M Training)	85.5	86.3	86.2	78.6	85.1	85.7	88.4	80.6

TABLE I: The robustness transferability comparison of nature training, FGSM, PGD and MROBUST methods using black-box adversarial attack from the source network on MNIST.

sizes 64 and 128 filters, and also a fully connected layer of size 1024. Both networks were adversarially trained with FGSM, PGD and MROBUST methods. The robustness transferability for black-box attack adversary between different adversarial trained methods and architectures are shown in Table I. More explanations about the experimental results are described in Subsection IV-C.

The CIFAR10 dataset contains a training set of 50,000 examples, and a test set of 10,000 examples of 32×32 colour images in 10 different classes. The value of each pixel in the input images were normalised in the interval $[0, 1]$. As before, we generated adversarial examples under the perturbation constraints of size $\epsilon = 0.1$ in the l_∞ norm. For the CIFAR10 dataset, we used Resnet model [38] as the baseline model and constructed a variant with layers wider by a factor of 10, resulting in a wide network with 5 residual units with (16, 160, 320, 640) filters each. This network achieved up to 95.2% accuracy on clean test dataset. We also performed adversarial training with FGSM, PGD and MROBUST methods on these two networks and investigated the robustness transferability for black-box attack adversary between different adversarially trained methods. The resulting robustness transferability measures are shown in Table II. We will explain more experimental results in Subsection IV-D.

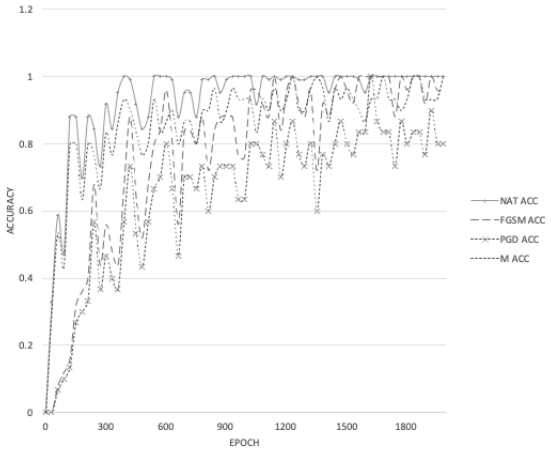

Fig. 2: Adversarial Accuracy on MNIST dataset with $\epsilon = 0.1$.

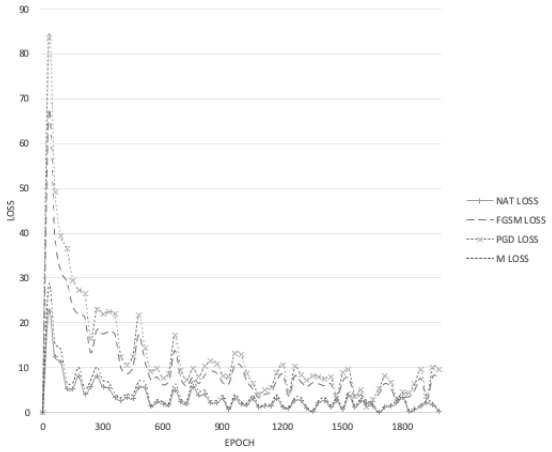

Fig. 3: Adversarial Loss on MNIST dataset with $\epsilon = 0.1$.

C. MNIST

Table I summarises the results for robustness transferability obtained for the MNIST dataset. We first trained DNNs with FGSM, PGD and MROBUST methods as source and target models and generated attacks with a MATTACK adversary based on the query result from each of the trained source models. The generated attack was then transferred into each of the target trained models and a check was carried to determine whether the target trained models can successfully defend against this attack. The results obtained show that a strong adversary generally reduces transferability and increases robustness transferability. For example, the robustness transferability with source simple trained model S. (M Training) paired with target S. (FGSM Training) is 74.7; this is higher than 68.4, which is the value for the same target but source S. (PGD Training). In addition, the MROBUST method generally augments robustness transferability for different models, except for those that have MROBUST itself as a source. For instance, the robustness transferability with source simple trained model S. (M Training) paired with target W. (PGD Training) is 83.5; this is higher than 82.7, which is the value for the same target but source S. (PGD Training). This happens in most of the cases. On the contrary, the robustness transferability with source simple trained model S. (M Training) paired with

Target \ Source	S. (Nature Training)	S. (FGSM Training)	S. (PGD Training)	S. (M Training)	W. (Nature Training)	W. (FGSM Training)	W. (PGD Training)	W. (M Training)
S. (Nature Training)	9.1	75.4	76.4	76.9	3.4	71.9	77.6	78.3
S. (FGSM Training)	68.3	53.4	58.2	64.9	66.1	55.5	66.9	72.5
S. (PGD Training)	69.6	68.1	70.7	69.4	69.3	65.9	70.1	73.2
S. (M Training)	73.3	74.1	75.2	64.5	72.1	74.9	76.4	69.3
W. (Nature Training)	10.3	79.1	78.3	79.6	8.3	67.6	75.9	79.8
W. (FGSM Training)	69.1	63.2	69.7	71.5	69.4	54.9	70.1	74.6
W. (PGD Training)	70.6	69.8	71.3	72.9	70.8	69.2	71.9	72.8
W. (M Training)	75.7	77.1	76.8	69.4	75.2	76.6	79.2	68.3

TABLE II: The robustness transferability comparison between nature training, FGSM, PGD and MROBUST methods using black-box adversarial attack from the source network on CIFAR10.

target W. (M Training) itself is 78.6; this is lower than 86.2, which is the value for the same target but source S. (PGD Training). The reason is that we generate attacks based on the source model using the similar method. Moreover, changing the architecture from simple to wide networks generally has a positive effect to robustness transferability; so the value with source wide model W. (M Training) paired with target W. (M Training) is higher than source S. (M Training) paired with target W. (M Training).

We report the accuracy and loss for different adversarial training methods over the first 2,000 epochs in Figure 2 and Figure 3. The results show that the convergence is highest for nature training with no adversarial examples, followed by MROBUST, then PGD, and finally FGSM. The results show that MROBUST converges faster than the competing methods with 13.2 % because it searches for only potential candidates of the perturbations for adversarial examples. This can also help to reduce the required adversarial examples for adversarial training and thus save the training efforts. In summary, the adversarial training with MROBUST converges faster than other two methods.

D. CIFAR10

Table II summarises the robustness transferability on the CIFAR10 dataset. As above the results show that a strong adversary reduces transferability and helps the robustness transferability. For instance, the robustness transferability with source simple trained model S. (M Training) paired with target W. (FGSM Training) is 71.5; this is higher than 69.7, which is the value for the same target but source S. (PGD Training). Furthermore, the MROBUST contributes to robustness transferability for different models, except for those with a source MROBUST themselves; see, e.g., source S. (M Training)/target W. (M Training) is lower than source S. (PGD Training)/target W. (M Training), whilst the value of source S. (M Training)/target W. (PGD Training) is higher than source S. (PGD Training)/target W. (PGD Training). Moreover, changing the architecture from simple to wide networks improves robustness transferability in average, e.g., source W. (M Training)/target W. (M Training) is higher than source S. (M Training)/target W. (M Training).

We plot the accuracy and loss for different adversarial training methods over the first 25,000 epochs in Figure 4 and Figure 5. As above we found that nature training method converged faster than all, followed by MROBUST, PGD and FGSM. The results also show that MROBUST requires fewer adversarial examples during adversarial training and thus

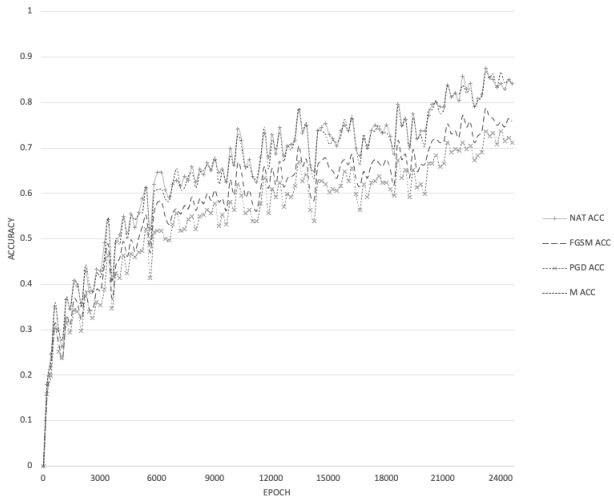

Fig. 4: Adversarial Accuracy on CIFAR10 dataset with $\epsilon = 0.1$.

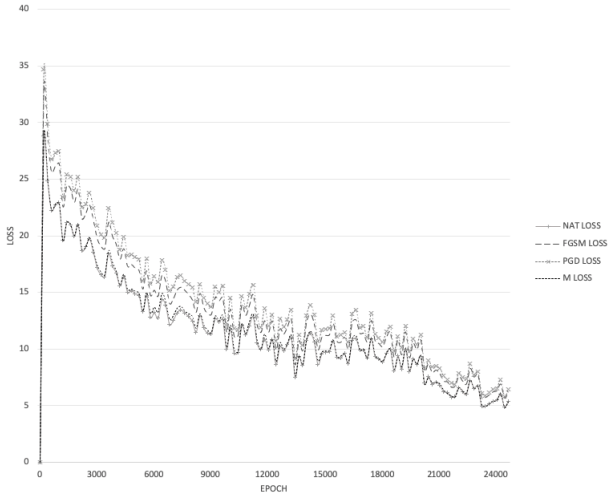

Fig. 5: Adversarial Loss on CIFAR10 dataset with $\epsilon = 0.1$.

improves the training efforts with 5.2%. As in the MNIST case, we can see from Figure 4 and Figure 5, the adversarial training with MROBUST converges more efficient than other two methods.

V. CONCLUSIONS

In this paper we proposed the MROBUST defence method with MCTS and evaluated the robustness transferability results on MNIST and CIFAR10 datasets. We focused on small perturbations from potential candidates that are capable to generate adversarial examples as this can save time complexity for adversarial training and increase the robustness against adversarial attacks. In real applications some pre-processing components like denoising elements are normally included prior to neural network models; so we here only focus on potential perturbations. The results show: i) that the method is computationally attractive, ii) it can defend against the FGSM and PGD attacks, and iii) does not appear to be susceptible to local optima. In future work, we will continue to improve robustness for different datasets like German Traffic Signs against different attacks.

REFERENCES

[1] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[2] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *arXiv: 1607.02533*, 2016.

[3] S. 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 (CVPR)*. IEEE, 2016, pp. 2574–2582.

[4] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 427–436.

[5] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," *ECML PKDD*, vol. 8190, no. 3, pp. 387–402, 2013.

[6] F. Roli, B. Biggio, and G. Fumera, "Pattern recognition systems under attack," *of the 18th Iberoamerican Congress on Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, vol. 8258, pp. 1–8, 2013.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[8] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[9] A. Fawzi, S. Moosavi-Dezfooli, P. Frossard, and S. Soatto, "Empirical study of the topology and geometry of deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 3762–3770.

[10] X. Ma, B. Li, Y. Wang, S. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *arXiv:1801.02613*, 2018.

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

[12] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," in *arXiv:1704.03453*, 2017.

[13] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1765–1773.

[14] I. Apple, "Face id security https://images.apple.com/business/docs/FaceID_Security_Guide.pdf," 2017.

[15] M. Bojarski, D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," in *arXiv: 1604.07316*, 2016.

[16] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proceedings of 10th International Conference on Malicious and Unwanted Software (MALWARE)*, 2015, pp. 11–20.

[17] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*. IEEE, 2016, pp. 582–597.

[18] A. Rozsa, M. Gunther, and T. Boult, "Towards robust deep neural networks with bang," in *arXiv:1612.00138*, 2016.

[19] F. Tramer, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *arXiv:1705.07204*, 2017.

[20] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," in *arXiv:1706.04701*, 2017.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[22] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *arXiv:1704.01155*, 2017.

[23] V. F. J.H. Metzen, T. Genewein and B. Bischoff, "On detecting adversarial perturbations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[24] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[26] Z. Gong, W. Wang, and W. Ku, "Adversarial and clean data are not twins," in *arXiv:1704.04960*, 2017.

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

[28] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[29] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 446–454.

[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 38th IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 39–57.

[31] C. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006.

[32] S. Haykin, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2009.

[33] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999, pp. 1150–1157.

[34] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proceedings of the 5th International Conference on Computers and Games*, 2006, pp. 72–83.

[35] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G.Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[36] Y. LeCun and C. Cortes, "Mnist handwritten digit database http://yann.lecun.com/exdb/mnist/," 1998.

[37] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10 https://www.cs.toronto.edu/~kriz/cifar.html," 2010.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.