Robustness Learning via Decision Tree Search Robust Optimisation

Yi-Ling Liu
y.liu17@imperial.ac.uk
Alessio Lomuscio
a.lomuscio@imperial.ac.uk

Imperial College London
United Kingdom

Abstract

We present a novel method for robustness training for ReLU-based deep neural networks. The method involves a decision tree search targeting the worst-case data points to generate adversarial examples. We combine the decision tree search method with robust optimisation to train a robust model while maintaining accuracy at comparably lower computational effort than state-of-the-art methods. The efficiency is obtained by focusing on small regions centred around the input that have significant potential to generate adversarial samples. We implemented the resulting method in the framework DTSROBUST, which was evaluated against state-of-the-art defence methods on MNIST and CIFAR10 datasets. In experiments, DTSROBUST achieved a 14.2% gain on efficiency against the state-of-the-art defence methods in MNIST and 10.3% of that in CIFAR10 while maintaining similar accuracy.

1 Introduction

In the recent decade, machine-learning methods based on deep neural networks (DNNs) have achieved breakthroughs in several domains, including computer vision [26] and natural language processing [5]. Increasingly, neural networks are being considered for safety-critical systems. For this to happen in a safe and secure manner, it is essential that the DNNs used in safety-critical applications are robust and trustworthy.

It is now well-known that many DNNs are generally fragile to seemingly imperceptible changes to their input data [28]. Well-documented examples of such fragility to carefully-designed noise can be found in the context of image detection [12], video analysis [31], and traffic sign misclassification [6]. In response to this vulnerability, a growing body of work has focused on improving the robustness of DNNs [22, 25]. In particular, the literature concerning adversarial robustness has sought to improve robustness to small, imperceptible perturbations of data. To this end, robust training algorithms, i.e. adversarial training [8], typically incorporate norm-bounded, adversarial data perturbations in a robust optimisation formulation.

Adversarial training has provided a rigorous framework for analysing, and improving the robustness of DNNs considering norm constrained perturbations. However, adversarial training requires learning via a large number of perturbed examples before robustness may be obtained. This is expensive and time-consuming. Therefore, developing computationally effective and robust training approaches is a topic of interest.

In this research, we propose a robust learning method based on decision tree search and robust optimisation. Given an arbitrary input to a DNN, our algorithm searches in small regions centred around the input that have significant contributions to generate adversarial samples. As we show, the method results to be more robust against different adversarial attacks and is competitive results with Fast Gradient Sign Method (FGSM) [8] and Projected Gradient Descent (PGD) [22]. In the experiments reported the method also reduced significantly the number of adversarial examples required for adversarial training leading to computational advantages when generating robust models.

The remainder of this paper is organised as follows. Section 2 introduces some preliminaries about how adversarial examples can be formulated under different attacks on DNNs. In Section 3 we describe a novel method to generate adversarial examples with decision tree search, while Section 4 introduces how adversarial examples are employed for adversarial training via robust optimisation. Section 5 reports experimental results obtained on the MNIST and CIFAR-10 datasets; section 6 concludes the paper.

Related work. Several defence methods for DNNs have been proposed to improve robustness. A method for network distillation to improve the generalisation capability of DNNs by transferring extracted knowledge from a large network to a small one was proposed in [25]. Although the experiments showed that network distillation can enhance robustness against Jacobian-based Saliency Map Attack (JSMA) [24], network distillation was unable to defend against the PGD attack developed in [22]. Adversarial training [8] concerns explicitly training a model on adversarial examples, in order to make it more robust against attacks or to reduce its test error on clean inputs. A disadvantage of adversarial training is that it takes more training efforts compared with other methods. In [14] a method is put forward to create a deep subnetwork as an auxiliary network to detect adversarial examples. Several related works have also attempted to detect adversarial examples in the testing stage [9, 21], but failed to defend against FGSM and Carlini & Wagner Attack (C&W) [3]. Input reconstruction approaches aim to transform adversarial examples to clean data and then applying these to assist neural networks to predict correct results with denoising autoencoders (DAEs) [10], but it does not guarantee global optimality.

A related stream of research focuses on robustness against black-box attacks and transferability [20, 23]; these are not directly comparable to the present method which focuses on white-box attacks. More generally, a recently emerging line of work focuses on efficient verification methods for neural classifiers [1, 7, 13, 15, 16, 30]. While these efforts can improve adversarial training, they remain focused on model validation.

2 Preliminaries

We here define the notations and symbols used for problem formulation and then recall two state-of-the-art adversarial attack methods, which are also used for experimental comparisons in the experimental section.

Problem Formulation. Suppose the number of object classes k in a 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 $y_i \in \{1, \ldots, k\}$ is a label in k classes. Given a deep neural network $\mathcal{N}$ with associated function $f(\cdot) : \mathbb{R}^d \to y_i$, the loss function of the network with parameters θ on (x_i, y_i) is expressed as $\mathcal{L}(x_i, \theta, y_i)$, for instance the cross-entropy loss for a neural network. An adversarial example and the label of that are defined as $x'_i \in \mathbb{R}^d$ and $y'_i \in \{1,...,k\}$, respectively. A perturbation between the original input data x_i and the adversarial example x'_i is represented as $\delta = x'_i - x_i \in \mathbb{R}^d$. The gradient of the loss function $\mathcal{L}$ with respect to the vector x_i is denoted as $\bigtriangledown_x \mathcal{L}(\cdot)$.

Given a trained neural network $\mathcal{N}$ and an original input data sample x, generating an adversarial example x' under some norm p can be defined as a box-constrained optimisation problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in \mathcal{U}} \mathcal{L}(x', \theta, y) \right], \tag{1}$$

where $\mathcal{U} = \{\delta : \|x - x'\|_p < \varepsilon\}$ denotes the uncertainty set of allowed perturbations between two data samples within l_p distance. This optimisation problem emphasises on searching an optimal solution with minimal cost given the worst-case realisation from $\mathcal{U}$.

Adversarial Attack Methods. We recall two representative attack methods for generating adversarial examples. We start from Fast Gradient Sign Method (FGSM) [8] which consists of a fast method to generate adversarial examples. The second is the PGD attack method [22] which is robust against most of the existing adversarial detecting defences. We will evaluate robustness of our work against these two attack methods in the experimental section 5.

1.) Fast Gradient Sign Methods (FGSM). [8] propose a fast method for generating adversarial examples called Fast Gradient Sign Method. They perform one step gradient update along the direction of the sign of gradient at each pixel of input data. In this approach, given an adversarial input $x' = x + \delta$, where x is the original input image and δ is the perturbation, the loss function of the model for adversarial input can be express as Equation (2):

$$\mathcal{L}(x', \theta, y') = \mathcal{L}(x, \theta, y) + (x' - x) \nabla_x \mathcal{L}(x, \theta, y), \tag{2}$$

where θ is the hyperparameters of a model. Through minimising $\mathcal{L}(x', \theta, y')$ subjecting to $\|x' - x\|_\infty \leq \varepsilon$, the required perturbation is derived as Equation (3):

$$\delta = x' - x = \varepsilon \cdot sign(\nabla_x \mathcal{L}_\theta(x)), \tag{3}$$

where ε is the magnitude of the perturbation constraint. Increasing ε value increases the likelihood of x' being misclassified by the classifier $f(\cdot)$, but larger changes are more easily detected by humans.

2.) Projected Gradient Descent (PGD). The PGD attack [22] is widely believed to be one of the state-of-the-art attack methods. This method adopts the multi-step variant FGSM, which is essentially projected gradient descent on the negative loss function [18]:

$$x'_{t+1} = Clip_{x+\delta}(x'_t + \alpha \cdot sign(\nabla_x \mathcal{L}(x', \theta, y))), \tag{4}$$

where α is the variant step size at the step number t and *Clip* function makes sure 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.

3 Decision Tree Search Attack

We now present a decision tree search adversarial attack DTSATTACK to generate effective adversarial examples. The method is composed of four steps. We first initialise spanning tree for tree traversal; then we traverse such tree according to a confidence value and initialise

Algorithm 1: Decision Tree Search Adversarial Attack: DTSATTACK

```
1  function DTSADVERSARIALATTACK ;
   Input  : Clean image dataset $x$
            Initialise perturbation constraint setting for $\varepsilon$
            Initialise search trees $T$
   Output: Effective adversarial examples $x'$
2  Initialise Spanning Tree to obtain maximal distance values;
3  while not terminalNode and $\mathcal{P} \leq \varepsilon$ and time < TimeOut do
4      while iterationTime < stepSearchTime do
5          Tree Traversal: Traverse the spanning tree according to the confidence value for each
             node $i$;
6          Initialise Exploration Nodes: Explore available expanding nodes $N_e$;
7          for each exploration nodes $N_e$ do
8              Nodes Sampling: Randomly choose one region from available sub-regions;
9              Back Propagation: Update associated information for each node along the path;
10         end
11     end
12     Choose Best Child Node: Choose one of the best path from the root node ;
13     Make One Move: Make one move based on the best exploration node as child node and
         update new root node;
14 end
```

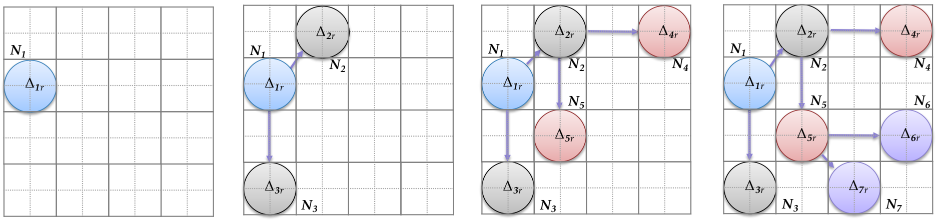
explorable nodes. We then sample values for each node on the tree from the explorable nodes and update the confidence information in the back propagation step. This iteration continues until the termination conditions are satisfied, as summarised in Algorithm 1.

Spanning Tree Initialisation. For each image $x \in \mathbb{R}^d$ in d dimensions, x can be separated into m sub-regions with the number of pixels j in each sub-region. To search the most potential pixels with a higher probability of generating an adversarial example, we first compute each pixel by the distance between the average value x_a and each pixel value as $\mathcal{D} = \|x_i - x_a\|$, where x_i is the value of each pixel i, and the matrix $\mathcal{D}$ is the distance of each pixel x_i from the average value. The larger value in $\mathcal{D}$, it stands for the more potential pixel intuitively. We then sort $\mathcal{D}$ by the value with descent order as $\mathcal{D}_{\mathcal{S}} = Sort(\mathcal{D})$ and divide $\mathcal{D}_{\mathcal{S}}$ into m sub-regions N_m with the number of points j in each sub-region. This initialisation step establishes a starting search step for the root node N_r of the spanning tree, which will be the start point of the next tree traversal step.

Tree Traversal. We now present how to expand nodes from the initialised spanning tree and what criteria are used to make decisions of choosing nodes in the tree traversal step. In this step we consider the most promising moves according to a confidence value of each node in the spanning tree. Each node of the spanning tree is regarded as a sub-region N_m from the previous step. In each move k, we choose a child node N_c with the highest confidence value down to a leaf node N_l. The confidence value $\mathcal{C}_k$ of associated information in each node N_c of the spanning tree is formulated as Equation (5), which is on the basis of Monte Carlo Tree Search [4] algorithm, a decision search algorithm for decision process, and revised accordingly:

$$\mathcal{C}_k = \frac{w_k}{v_k} + c\sqrt{\frac{\ln V_k}{v_k}}, \tag{5}$$

where w_k stands for the Euclidean distances for the node considered after the k-th move, v_k stands for the number of visits for the node considered after the k-th move, and V_k stands for



(a) (b) (c) (d)

Figure 1: The process of the decision tree search attack.

the total number of visits after the k-th move. The exploration parameter c is theoretically equal to $\sqrt{2}$. The higher the confidence value means the more contributions to search adversarial examples. Conversely, the lower probability to obtain an adversarial example the lower is the confidence value. This confidence measure above is inspired by neural approaches to game playing [27].

Node Sampling. In a sampling step, from the leaf node N_l identified in the previous step, we choose among the exploration nodes N_e. The explorable nodes consist of the remaining sub-regions excluding the ancestors of a node N_a in the tree. For example, the explorable nodes of the root node N_r are the remaining sub-regions from the sub-regions in the root node. We first sample nodes from these explorable nodes and simulate whether an adversarial example is generated. This step continues by choosing from the remaining sub-regions randomly and applying perturbations accordingly on the datapoints j in each sub-region until the end of the search time. We then simulate based on these newly perturbed points and examine whether an adversarial example is found. The randomly choosing process is formulated as:

$$\mathcal{RC} = random(\mathcal{S}_{avai}(N_e) - \mathcal{S}_{used}(N_a)), \tag{6}$$

where *random* choose one region from the remaining sub-regions randomly, $\mathcal{S}_{avai} = \{N | N \in N_e, \text{exploration nodes}\}$ is the available set of explorable sub-regions and $\mathcal{S}_{used} = \{N | N \in N_a, \text{ancestor nodes}\}$ is the sub-regions in the ancestors of a node.

Back Propagation. We now present how to update the information for the newly explored nodes under the constraint with $\|\delta\|_p \leq \varepsilon$. We select one path from the previous sampling step with the maximal distance between the newly perturbed and the clean points. We then back propagate and update the corresponding distance value and number of visits for the confidence values of nodes along the expanding path. The perturbation amount $\mathcal{P}$ in each iteration is constrained with $\|\delta\|_p \leq \varepsilon$, which is formulated as Equation (7):

$$\mathcal{P} = \|\sum_{t=0}^{T}\sum_{r=0}^{R} \Delta_{tr}\|_p \leq \varepsilon, \tag{7}$$

where $\sum_{r=0}^{R} \Delta_{tr}$ stands for the number of perturbations in each node, and T is the total number of nodes along the same path.

The whole process is represented in Figure 1a-1d. First, the image is divided into m sub-regions and the root node N_1 is selected as the start point in the spanning tree (Figure 1a). The tree is then expanded to N_2 and N_3 (Figure 1b). These two nodes are then sampled and the updated confidence values are back propagated accordingly. Then path with highest confident values of nodes (N_1, N_2) is selected (Figure 1c). The expanded nodes N_4 and N_5 are then sampled to check if an adversarial example is acquired. The confidence values are updated for each node of the whole path accordingly. The nodes (N_1, N_2, N_5) are selected and the nodes (N_6, N_7) are expanded from N_5 (Figure 1d). The simulation and back-propagation

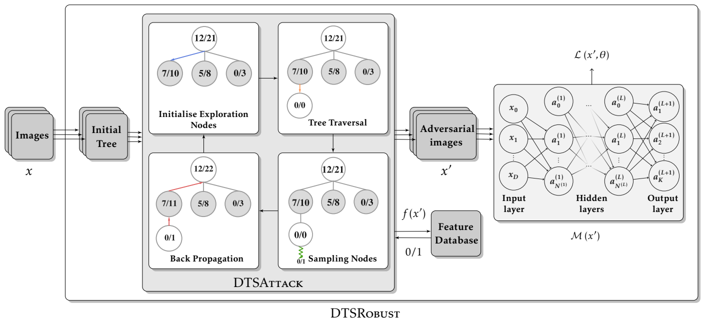


Figure 2: The DTSROBUST Implementation Framework.

steps are applied as previously mentioned. These iterations continue until the termination conditions are satisfied. The parameter Δ_{tr} stands for the number of perturbations in each node. For example, Δ_{1r} in the node N_1 is the perturbations applied when N_1 is selected. The overall perturbations applied in an adversarial example are the summation of perturbations of the nodes along the whole path.

The decision tree search attack is summarised in Algorithm 1. The algorithm starts by initialising the spanning tree (line 2). In the search iteration, the spanning tree is traversed according to the confidence value (line 5) and available expanding nodes are then explored (line 6). For each exploration node, one region is randomly chosen from available sub-regions for sampling (line 8) and the associated information is updated for the nodes along the search path (line 9). These iterations (line 4-13) will continue until the termination conditions are satisfied.

4 The DTS Robust Framework

In this section we introduce a robust optimisation method that interacts with DTSATTACK from the previous section to form the basis of the DTSROBUST framework. To be more specific, we combine robust optimisation with DTSATTACK thereby obtaining a method that is evaluated against other attack methods, which also include robust optimisation.

Robust Optimisation. The robust optimisation method [1] described below aims to obtain stable solutions under uncertainty of the data. The uncertainty has a deterministic and worst-case nature; perturbations to the data are drawn from uncertainty sets $\mathcal{U}$. The objective in robust optimisation is to obtain solutions that are feasible and well-behaved under any realisations of the uncertainty from $\mathcal{U}$. An optimal solution among feasible solutions has minimal cost given the worst-case realisation from $\mathcal{U}$. Robust optimisation thus normally have a min-max formulation, where the objective function is minimised with respect to a worst-case realisation of perturbations. The corresponding robust optimisation is:

$$\min_{x} \sup_{(A,b,c)\in\mathcal{U}} \{c^T x : Ax \leq b\}, \tag{8}$$

where A is a given matrix, (b,c) are the given vectors and the objective is to search a solution x which is robust to perturbations within the uncertainty set $\mathcal{U}$ in the data.

Given this, the problem can be formulated as the search of a stable solution in a small neighbourhood around every training point x_i. This neighbourhood corresponds to the uncertainty set $\mathcal{U}_i$. For example, we may set $\mathcal{U}_i = R_p(x_i, r)$, a region with radius r around x_i with respect to some norm p. To do so, we select from the neighbourhood a representative

$x_i' = x_i + \delta_{x_i}$, the point on which the network output will induce the greatest loss. The network output on x_i' is required to be y_i, the output for x_i. Assuming that many test points are close to training points from the same class, we expect that the training algorithm will have a regularisation effect and consequently will improve the network performance on test data. Moreover, we expect this approach to increase the robustness of the network output to adversarial example. Hence, the training network is optimised with a min-max approach:

$$\min_{\theta} \mathcal{L} = \min_{\theta} \sum_{i=1}^{m} \max_{\delta \leq \|\varepsilon\|} J(x_i', \theta, y_i), \tag{9}$$

where δ is the uncertainty 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) with entropy J, rather than against the original training data; the i-th worst-case data point is selected from the uncertainty set δ. The uncertainty sets are determined by the problem at hand; adversarial training [8] can be understood as one such problem.

DTS Implementation Framework. We now introduce the decision tree search robust optimisation framework in Figure 2, including the robust optimisation function applied in a training model $\mathcal{M}$. This is divided into two steps: the first consists of an attack generation step with DTSATTACK; the second is a robust optimisation step. For the attack generation part, first, a model $f(\cdot)$ is trained as an assistant classifier from a training dataset of images x. Then, the procedure follows a training loop. In each training loop, corresponding to a training epoch, a minibatch of size m of images is randomly selected from the training dataset x. This minibatch is then analysed by the DTSATTACK to generate adversarial examples x'. Whether or not the images generated by DTSATTACK are proper adversarial examples is determined by a test on the assistant classifier $f(\cdot)$. The resulting adversarial examples x' form the basis for training the robust model $\mathcal{M}(\cdot)$, which is initially untrained. The loss value $\mathcal{L}$ is updated according to the softmax result of the robust model $\mathcal{M}$. The training loop continues for as many epochs as required until the required accuracy is converged.

Following the framework mentioned above, the DTSROBUST randomly selects a minibatch in each epoch until the loss of the robust model converges to a desired value. The convergence criteria ensures that resulting model $\mathcal{M}$ 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.

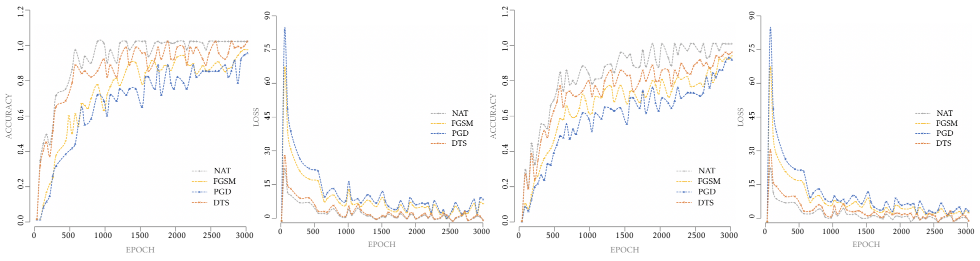
5 Experimental Results

In this section, we evaluate the DTS robust optimisation algorithm presented in the previous section and report the results obtained with the MNIST [19] and CIFAR10 [17] datasets. We evaluate the robustness obtained against different attack methods, namely FGSM, PGD, and DTS. More details about experiments can be found in the supplementary material.

Experimental Setup. 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-normalised and centred in a fixed-size image of 28×28. We generated adversarial examples under the perturbation constraints of size $\varepsilon_{\text{DIFF}} = 0.02$ and 0.03 respectively. The parameter $\varepsilon_{\text{DIFF}}$ means that the modification rates of pixels for adversarial examples are no more than 2% and 3%. To investigate model capacity, we consider two training networks of simple and wide architectures, respectively. The simple network consists of two convolution layers of

Target Model \ Adversary	Nature	FGSM	FGSM^R	PGD^2	PGD^5	PGD^{20}	DTS^1	DTS^2	l_2	$\varepsilon_{\text{DIFF}}$
S. (FGSM Training)	96.2	95.1	94.8	92.7	91.4	89.7	90.8	89.6	3.87	0.02
S. (PGD Training)	95.8	95.2	94.2	93.6	92.7	90.6	91.5	90.8	3.71	
S. (DTS Training)	**97.7**	**96.3**	**95.1**	**93.8**	**93.3**	**91.9**	**91.9**	**91.6**	3.27	
W. (FGSM Training)	97.4	96.7	96.4	94.6	94.1	92.7	92.8	92.3	3.95	
W. (PGD Training)	96.9	95.7	95.3	95.1	94.8	93.4	93.5	93.1	3.86	
W. (DTS Training)	**97.8**	**96.8**	**96.3**	**95.8**	**95.2**	**94.1**	**93.9**	**93.6**	3.79	
S. (PGD Training)	94.8	92.7	91.9	90.7	89.9	88.7	89.5	88.6	4.17	0.03
S. (DTS Training)	**96.5**	**94.1**	**93.2**	**92.7**	**92.3**	**90.6**	**90.5**	**90.1**	4.04	
W. (PGD Training)	96.1	92.8	92.1	91.3	90.3	89.7	90.1	89.3	4.28	
W. (DTS Training)	**97.3**	**94.3**	**93.7**	**93.1**	**92.5**	**91.7**	**91.8**	**91.2**	4.21	

Table 1: The resulting accuracy of nature training, FGSM, PGD and DTS methods against white-box adversarial attacks with $\varepsilon_{\text{DIFF}} = 0.02$ and 0.03 on MNIST dataset.



(a) Accuracy ($\varepsilon_{0.02}$). (b) Loss ($\varepsilon_{0.02}$). (c) Accuracy ($\varepsilon_{0.03}$). (d) Loss ($\varepsilon_{0.03}$).

Figure 3: MNIST accuracy and loss with $\varepsilon_{\text{DIFF}}$ = 0.02 ((a) & (b)) and 0.03 ((c) & (d)).

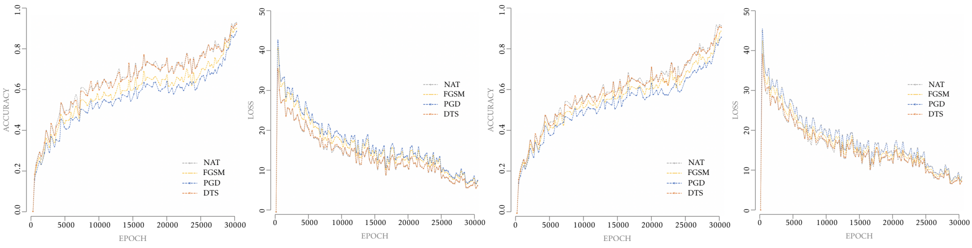
sizes 32 and 64 filters, and a fully connected layer of size 1024. The wide network consists of two convolution layers of sizes 64 and 128 filters, and also a fully connected layer of size 1024. Both networks are adversarially trained with FGSM, PGD and DTS methods; Table 1 reports the resulting accuracies against white-box attack adversaries on different adversarial trained methods and architectures.

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 values of input images were also normalised in the interval $[0,1]$. As before, we generated adversarial examples under the perturbation constraints of size $\varepsilon_{\text{DIFF}} = 0.02$ and 0.03. For the CIFAR10 dataset, we used the Resnet model [11] as the simple network and modified the network via using wider layers by a factor of 10, resulting in a network with 5 residual units with (16, 160, 320, 640) filters each. The experimental results are shown in Table 2.

MNIST. Table 1 summarises the resulting accuracies obtained on the MNIST dataset. We generated adversarial examples using the white-box attack method of FGSM, PGD and DTS with $\varepsilon_{\text{DIFF}} = 0.02$ and 0.03, and then evaluated them on a target network, that was adversarially trained independently, with different methods. The target models consist of two different architectures, which are simple and wide networks. For example, S.(FGSM Training) means the target model trained with FGSM adversarial training using a simple network. The first column (Nature) stands for the accuracy of each adversarially trained target network without attacks. The FGSM random attack (FGSM^R) was implemented according to [29], whereby small random perturbations are performed before applying FGSM. The PGD attack considered 10 random restarts uniformly distributed under ε per input and settings of 2, 5 and 20 steps with step size 0.01. The search time for DTS attack is constrained in 1 second and 2 seconds respectively; once an adversarial example is found, the process

Adversary / Target Model	Nature	FGSM	FGSM^R	PGD^2	PGD^5	DTS^1	DTS^2	l_0	l_1	l_2	$\varepsilon_{\text{DIFF}}$
S. (FGSM Training)	93.8	93.0	92.5	89.5	89.1	87.9	87.0	14.5	13.6	3.93	0.02
S. (PGD Training)	93.3	92.4	92.0	91.6	91.2	90.1	88.9	13.8	12.9	3.82	
S. (DTS Training)	**94.8**	**93.2**	**92.8**	**91.9**	**90.7**	**90.5**	**89.8**	12.3	11.3	3.74	
W. (FGSM Training)	94.2	93.9	93.3	91.4	90.8	90.3	89.5	14.2	13.9	3.91	
W. (PGD Training)	94.0	93.7	93.4	92.9	92.3	91.6	91.0	13.9	13.5	3.84	
W. (DTS Training)	**94.9**	**94.1**	**93.7**	**93.2**	**92.5**	**91.7**	**91.5**	12.4	11.9	3.79	
S. (PGD Training)	92.4	89.8	88.4	87.6	86.9	86.4	85.9	20.8	18.9	4.22	0.03
S. (DTS Training)	**93.2**	**92.5**	**91.8**	**90.7**	**89.7**	**88.6**	**87.4**	19.2	18.3	4.04	
W. (PGD Training)	93.4	90.1	88.9	88.1	87.4	87.1	86.2	20.3	19.2	4.34	
W. (DTS Training)	**93.8**	**92.5**	**92.1**	**91.4**	**90.2**	**89.3**	**88.6**	19.5	18.6	4.21	

Table 2: The resulting accuracy of nature training, FGSM, PGD and DTS methods against white-box adversarial attacks with $\varepsilon_{\text{DIFF}} = 0.02$ and 0.03 on CIFAR10 dataset.



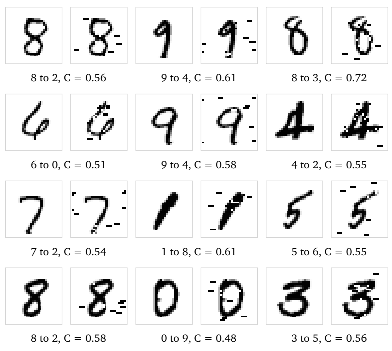
(a) Accuracy ($\varepsilon_{0.02}$). (b) Loss ($\varepsilon_{0.02}$). (c) Accuracy ($\varepsilon_{0.03}$). (d) Loss ($\varepsilon_{0.03}$).

Figure 4: CIFAR10 accuracy and loss with $\varepsilon_{\text{DIFF}}$ = 0.02 ((a) & (b)) and 0.03 ((c) & (d)).

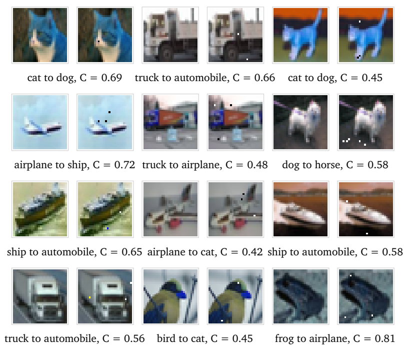
will be terminated. From the results, the attack strength from strong to weak is DTS, PGD and then FGSM as the accuracies against DTS attack (DTS^X columns) are lower than PGD and FGSM columns. This means that DTSATTACK is a stronger attack, and thus the resulting accuracies among different trained networks are lower. In addition, the results show that DTS contributes to improved accuracies against different adversaries, and maintain robustness even under the DTS attack itself. For example, the accuracy of S. (DTS training networks) is 93.8, which is higher than other training networks 92.7 and 93.6 under the PGD attack (column PGD^2). Moreover, changing the architecture from simple to wide networks also contributes to accuracies overall. The accuracies decrease only few percentages even increasing $\varepsilon_{\text{DIFF}}$ to 0.03 and the W. (DTS training) is more robust than W (PGD Training) in general. The average distances required for DTS are smaller than the others as DTS searches mainly the most potential features. Figure 3 reports the accuracy and loss trends for different adversarial training methods over the first 3,000 epochs with $\varepsilon_{\text{DIFF}}$ = 0.02 and 0.03. The average memory usage for FGSM, PGD and DTS are 1.5, 1.7, and 1.8GB respectively. We find that robust optimisation with DTS converges faster than the other two state-of-the-art methods with 14.2% on average when compared with PGD method.

CIFAR10. Table 2 summarises the resulting accuracies obtained on the CIFAR10 dataset with similar setting as MNIST. The results obtained also demonstrate that a strong adversary can help to improve model accuracies. In addition, DTS contributes to improving the accuracies against different adversaries while retaining robustness against the DTS attack itself. For example, the accuracy of W. (DTS training network) under the DTS attack itself (column DTS^2) is 91.5, which is higher than other networks with 91.0 or 89.5 (e.g., W. (PGD training network) or W. (FGSM training network)) with $\varepsilon_{\text{DIFF}} = 0.02$. Comparing the results against different network architectures reveals that changing the architecture from simple to wide networks can contribute to accuracies generally. The average distances required for DTS are smaller than FGSM and PGD as DTS only searches for the most potential features. Figure 4 reports the accuracy and loss trends for different adversarial training methods over the first 30,000 epochs. The average memory usage for FGSM, PGD and DTS are 1.9, 2.2, and 2.3GB respectively. The data obtained support the conclusion that the adversarial training with DTS converges faster with 10.3% on average than the other state-of-the-art methods.

Adversarial Examples with DTS We present some adversarial examples obtained for MNIST and CIFAR10 in Figure 5 with different confidence values under $\varepsilon_{\text{DIFF}} = 0.02$. Through DTSATTACK, only minor perturbations, where $\varepsilon_{\text{DIFF}} = 0.02$, are required to generate attacks. The odd columns are the original images with correct classes and the even columns are their corresponding adversarial examples. In the figure, most confidence values are over 0.5 while generating adversarial examples. From the results, the adversarial ones are clear to distinguish from the correct classes. Some of these attacks are not easy to detect by humans; see for example the ones reported for class cat to class dog with confidence values of 0.69 and 0.45 in CIFAR10.



(a) MNIST adversarial images ($\varepsilon_{0.02}$).


cat to dog, C = 0.69
truck to automobile, C = 0.66
cat to dog, C = 0.45
airplane to ship, C = 0.72
truck to airplane, C = 0.48
dog to horse, C = 0.58
ship to automobile, C = 0.65
airplane to cat, C = 0.42
ship to automobile, C = 0.58
truck to automobile, C = 0.56
bird to cat, C = 0.45
frog to airplane, C = 0.81

(b) CIFAR10 adversarial images ($\varepsilon_{0.02}$).

Figure 5: Some adversarial examples for MNIST and CIFAR10 under $\varepsilon_{\text{DIFF}} = 0.02$.

6 Conclusions

In this work we proposed a decision tree search robust optimisation framework and presented the experimental results obtained on MNIST and CIFAR10 datasets. We evaluated the proposed approach against small perturbations since pre-processing components like denoising elements are normally included in real applications. We believe that performance advantage obtained was due to the fact that the decision tree search method considers the most promising features during the robust learning process, and only searches over the parameterised manifold to worst-case perturbations of data. Further, DTS benefits from recent advances in Monte Carlo Tree Search. For the reasons summarised above, the results we obtained show that: i) the method can be deployed in different problems and datasets, e.g., MNIST and CIFAR10, ii) DTS is computationally attractive compared to the present state-of-the-art, iii) differently from other methods, it can defend against the FGSM and PGD attacks, and iv) it achieves global optima while maintaining robustness.

Acknowledgements. A. Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies.

References

[1] A. Bental, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

[2] E. Botoeva, P. Kouvaros, J. Kronqvist, A. Lomuscio, and R. Misener. Efficient verification of neural networks via dependency analysis. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 3291–3299, 2020.

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

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

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv:1810.04805*, 2018.

[6] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[7] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. AI^2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018.

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

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

[10] 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.

[11] 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)*, pages 770–778, 2016.

[12] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *arXiv:1903.12261*, 2019.

[13] P. Henriksen and A. Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, pages 2513–2520, 2020.

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

[15] G. Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. Dill, M. Kochenderfer, and C. Barrett. The marabou framework for verification and analysis of deep neural networks. In *Proceedings of the 31st International Conference on Computer Aided Verification (CAV)*, pages 443–452, 2019.

[16] P. Kouvaros and A. Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2643–2650, 2021.

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

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

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

[20] Y. Liu and A. Lomuscio. Mrobust: A method for robustness against adversarial attacks on deep neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

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

[22] 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.

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

[24] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 37th IEEE Symposium on Security and Privacy*, pages 372–387, 2016.

[25] 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*, pages 582–597, 2016.

[26] S. Sabour, N. Frosst, and G. Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[27] D. Silver, A. Huang, C.J. 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*, 529:484–503, 2016.

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

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

[30] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6367–6377, 2018.

[31] X. Wei, J. Zhu, S. Yuan, and H. Su. Sparse adversarial perturbations for videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8973–8980, 2019.