ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence
and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Attacking Model Sets with Adversarial Examples

István Megyeri[1], István Hegedűs[1] and Márk Jelasity[1,2] *

1- University of Szeged, Hungary
2- MTA-SZTE Research Group on Artificial Intelligence, Hungary

Abstract. Adversarial input perturbation is a well-studied problem in machine learning. Here, we introduce a generalized variant of this problem, where we look for adversarial examples that satisfy multiple constraints simultaneously over a set of multi-class models. For example, we might want to force an entire set of models to make the same mistake over the same example, in order to create transferable attacks. Or we might want to fool just a single model, without fooling the rest of the models, in order to target only a specific manufacturer. Known attacks are not directly suitable for addressing this problem. The generated example has to satisfy multiple constraints and no feasible solution may exist for any amount of perturbation. We introduce an iterative heuristic algorithm inspired by the DeepFool attack. We evaluate our method over the MNIST and CIFAR-10 data sets. We show that it can find feasible multi-model adversarial perturbations, and that the magnitude of these perturbations is similar to the single model case.

1 Introduction

It has been known for many years that most machine learning models are surprisingly sensitive to very small adversarial perturbations of the input [1, 2]. In the original formulation of the problem, we are given a fixed model and a correctly classified example. The attacker wishes to find a minimal perturbation of the example such that the model predicts any wrong label (untargeted attack) or a given desired label (targeted attack). Since the seminal papers on this topic, a large number of methods have been proposed to create better adversarial examples [3, 4] and to provide defense mechanisms [5, 6]. See [7] for a relatively recent overview.

Here, we propose and study a more general version of this problem, where we are given more than one model and an example. For each model, we specify whether the given model should correctly classify the example or predict any wrong label or predict a fixed specific label. This way, all the models specify a constraint on the desired perturbation. We assume a white box scenario where all the models are fully known.

This formulation allows for a wide variety of adversarial constraints on a given model set. Here, we focus on two patterns of constraints with interesting applications. In the first case, we wish to find an adversarial example such that all the models predict the same wrong label. This is related to the problem

*This study was supported by the National Research, Development and Innovation Office of Hungary through the Artificial Intelligence National Excellence Program (grant 2018-1.2.1-NKP-2018-00008), by grant TUDFO/47138-1/2019-ITM of the Ministry for Innovation and Technology, Hungary, and by the project “Integrated program for training new generation of scientists in the fields of computer science”, no EFOP-3.6.3-VEKOP-16-2017-0002, funded by the European Union and co-funded by the European Social Fund.

of finding adversarial examples for ensembles of models in the hope that these examples would also fool additional unseen black box models [8, 9, 10]. Known methods rely on the traditional setup where the ensemble is treated as a single model that has to be fooled. Unlike our approach, it always allows for multiple models in the ensemble to predict the correct label or even other inconsistent labels, as long as the ensemble decision is fooled.

The second pattern we study involves fooling a single model from the set, while making sure the rest of the models keep predicting the correct label. To the best of our knowledge, this is a novel scenario. In a sense, this is the inverse of the transferability problem, where we are looking for adversarial perturbations that do *not* transfer to other models, in a well-controlled manner. An interesting application is when an attacker wants to fool the product of a specific manufacturer, while making sure all the other products work correctly.

Our multi-constraint adversarial problem cannot be tackled with existing attack approaches directly. We propose an iterative optimization algorithm inspired by the DeepFool method [3]. We evaluate our method over the MNIST and CIFAR-10 data sets. We show our approach can find feasible multi-model adversarial perturbations, and that the magnitude of these perturbations is similar to the single model case.

2 Algorithm

Let us first introduce our notations. We assume a set of multi-class models $f_1, \ldots, f_m$ where $f_i : \mathbb{R}^d \to \mathbb{R}^C$. The models have C outputs that correspond to the possible class labels. The classification of a given input x by a model f_i is given by $k_i(x) = \arg\max_j f_{i,j}(x)$, where $f_{i,j}$ is the jth output dimension of f_i. We are looking for adversarial examples such that a given subset of the models is fooled while the rest of the models are not.

The basic idea behind the algorithm comes from the DeepFool method [3], where we also implement a heuristic iterative optimization algorithm based on the first order approximations of the decision boundaries. However, unlike DeepFool, we deal with several models targeted simultaneously by several different attack patterns. Our algorithm is shown in Algorithm 1. We assume that we are given an example x that is classified correctly by all the models. The models in $\mathcal{F}_t$ have to be fooled while those in $\mathcal{F}_p$ must not be. The loop runs until this goal is met. Within the loop, we ask for two perturbation steps: one that fools all the models in $\mathcal{F}_t$ and one that makes sure that all of the models in $\mathcal{F}_p$ predict the correct label. We apply the one with the larger norm. The idea is that this way we first solve the harder problem and then gradually satisfy the rest of the constraints.

A single iteration step is computed by Algorithm 2. The goal is to find a perturbation for x such that all the models in $\mathcal{F}$ predict a common label $c^* \in \mathcal{C}$, where $\mathcal{F}$ and $\mathcal{C}$ are parameters of Algorithm 2. In this version we present the untargeted version of the algorithm where this common label is not given as input, it can be arbitrary. The idea behind the algorithm is that for each label and each model we compute one potential targeted step for the iteration like the DeepFool iteration step [3]. We then pick the class label c^* that minimizes

Algorithm 1 Multi-model adversarial perturbation

```
Input: example $x$, targeted models $\mathcal{F}_t$, protected models $\mathcal{F}_p$
Assumption: $\forall f_j \in \mathcal{F}_t \cup \mathcal{F}_p : k_j(x) = \hat{c}$, where $\hat{c}$ is the correct class of $x$
$x_0 \leftarrow x$
$i \leftarrow 0$
while $i < i_{max}$ and $[\exists f_j \in \mathcal{F}_t : k_j(x_i) = \hat{c}$ or $\exists f_j \in \mathcal{F}_p : k_j(x_i) \neq \hat{c}]$ do
    $r_t \leftarrow$ getStep$(x_i, \{$every class label except $\hat{c}\}, \mathcal{F}_t)$
    $r_p \leftarrow$ getStep$(x_i, \{\hat{c}\}, \mathcal{F}_p)$
    $r \leftarrow r_{\arg\max_{i \in \{t,p\}} \|r_i\|_2}$        ▷ the larger of $r_t$ and $r_p$
    $x_{i+1} \leftarrow x_i + r$
    $i \leftarrow i + 1$
return $x_i$        ▷ the perturbed input
```

Algorithm 2 getStep

```
Input: example $x$, targeted classes $\mathcal{C}$, targeted models $\mathcal{F}$.
for $c \in \mathcal{C}$ do
    for $f_i \in \mathcal{F}$ such that $k_i(x) \neq c$ do        ▷ models predicting other than $c$
        $\hat{w}_{i,c} \leftarrow \nabla f_{i,c}(x) - \nabla f_{i,k_i(x)}(x)$        ▷ $\approx$ direction to class $c$
        $w_{i,c} \leftarrow \hat{w}_{i,c} / \|\hat{w}_{i,c}\|_2$        ▷ $\approx$ normalized direction to class $c$
        $\delta_{i,c} \leftarrow |f_{i,c}(x) - f_{i,k_i(x)}(x)| / \|\hat{w}_{i,c}\|_2$        ▷ $\approx$ distance to class $c$
    $m_c = \arg\max_i \delta_{i,c}$        ▷ index of model with maximal distance to $c$
$c^* \leftarrow \arg\min_c \delta_{m_c,c}$        ▷ class where maximal distance from $k_i(x)$ is minimal
return $\delta_{m_{c^*},c^*} \cdot w_{m_{c^*},c^*}$        ▷ perturbation towards making all $\mathcal{F}$ predict $c^* \in \mathcal{C}$
```

the maximal perturbation size over all the models. The maximal perturbation vector corresponding to this class label (where the maximum is taken over the models) is returned.

Note that there are several cases that we do not elaborate on here, for example, when some of the sets are empty. These can be handled in a natural way.

3 Experiments

We used the MNIST and CIFAR-10 data sets. The MNIST [11] data set consists of grayscale 28×28 images of handwritten digits, from 0 to 9. The CIFAR-10 [12] data set contains 32×32 RGB color images representing 10 classes of objects. The main properties are shown in Table 1. The column "Consistently Classified" is explained later on. As preprocessing, the features were normalized in both data sets to the range $[0, 1]$.

We created four model sets to test our multi-model attack method. Three sets were created on MNIST. For each set, we fixed a network structure and used eight different regularization parameters to train eight different weight sets for the network. A fourth set was created on CIFAR-10, where we used one network structure and eight different regularization parameters.

The three networks for the MNIST data set had one hidden layer of sig-

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Table 1: Properties of data sets

	Training Set	Test Set	#features (d)	Consistently Classified
MNIST	60 000	10 000	784	7860/9180/9443
CIFAR-10	50 000	10 000	3072	4335

Table 2: Regularization coefficients used to create model set

	0	1	2	3	4	5	6	7
MNIST 10	0	1e-8	1e-7	1e-6	1e-5	1e-4	1e-3	1e-2
MNIST 100	0	1e-9	1e-8	1e-7	1e-6	1e-5	1e-4	1e-3
MNIST 1000	0	1e-9	1e-8	1e-7	1e-6	1e-5	1e-4	5e-4
CIFAR-10	0	1e-5	1e-4	1e-3	2e-3	3e-3	4e-3	5e-3

moid neurons of size 10, 100 and 1000, respectively, and a softmax output layer. On CIFAR-10 we trained a convolutional network with a shallow LeNet-like architecture. It uses two blocks of two convolutional layers followed by max-pooling, followed by two dense layers. Every layer has ReLU activation except the last one, which has a softmax activation. The dimensions of the first four convolutional layers of 3x3 filters, and the last two dense layers are (32x32x32), (30x30x32), (15x15x64), (13,13,64), 512, and 10. This results in 1,250,858 parameters.

We used ADAM [13] as our optimizer with a minibatch size of 128 and a stopping threshold of 10^{-10}. The eight regularization parameters were different for each network, as seen in Table 2. The reason is that we calibrated the range so that the last setting is overly regularized.

The properties of the individual models in the model sets are shown in Figure 1 (left). We define robustness as the L_2 norm of the untargeted adversarial perturbation found by DeepFool, normalized by $\sqrt{d}$, where d is the input dimension. We normalize with $\sqrt{d}$ because, in the case of image data, this way we characterize the sensitivity of each pixel irrespective of the resolution of the image, which is a more natural measure. Recall, that each input feature has a value in the range $[0, 1]$. We can see that robustness is increasing with regularization in all the cases, as expected.

The last column of Table 1 shows the number of test examples that were correctly classified by all the models in the respective model set, in the case of MNIST in the order of the 10, 100 and 1000 neuron hidden layers. Our multi-model attack method was evaluated on these consistent examples only. For each model set, we computed the adversarial perturbation for all these test examples in 9 different scenarios. This includes targeting each of the 8 different models in the model set individually while protecting the rest of the models (that is, $\mathcal{F}_t = \{f_i\}$ and $\mathcal{F}_p = \{f_0, \dots, f_7\} \setminus \mathcal{F}_t$ for $i = 0, \dots, 7$) as well as targeting all the models simultaneously (that is, $\mathcal{F}_t = \{f_0, \dots, f_7\}$ and $\mathcal{F}_p = \{\}$). The number of iterations was limited by $i_{max} = 1000$. We used the models without the softmax activation, as was done in [3].

All the attacks were successful for all the examples, except in the case of MNIST with 10 hidden neurons, where the number of examples on which the

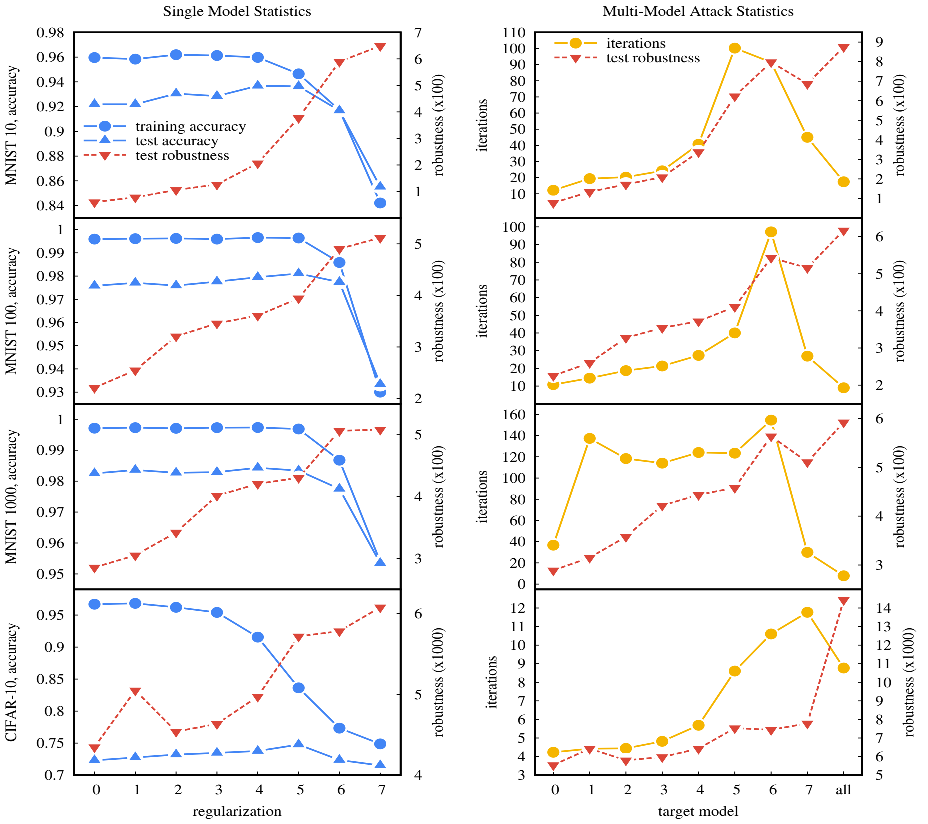

Fig. 1: Properties of the individual models (left), and multi-model attack statistics, where the horizontal axis indicates the targeted models $\mathcal{F}_t$ (right).

attack was not successful ranged from 16 to 278 out of the 7860 consistent examples, which amounts to an 0.2% to 3.5% error rate. The results are shown in Figure 1 (right). The required number of iterations of our attack method is rather small. Surprisingly, the CIFAR-10 model set requires much fewer iterations than the MNIST sets despite it containing larger models.

Quite surprisingly, the multi-model perturbations are very similar in size to those of the single model (DeepFool) perturbations shown in the left column. This result was not anticipated, because the models differ only in the applied regularization coefficient, so they are fairly correlated, which would suggest that finding an adversarial example that fools one model but not the others is hard. However, in all the model sets, even for the most robust model (with large regularization) we can easily find an adversarial example with very small perturbation that does not fool the rest of the models. In the CIFAR-10 dataset, the perturbation found for the “all targeted” case has a somewhat larger magnitude than that for the other attacks. However, it is still an extremely small (normalized) perturbation - amounting to less than 1.5% per input feature.

4 Conclusions

We proposed an iterative algorithm to find small adversarial perturbations that fool a given set of models simultaneously in a given pattern. This problem formulation has several applications including the generation of transferable adversarial examples, as well as *non-transferable* examples that target only a specific model and ensure that the other models are safe.

The algorithm applies the first-order approximation of the decision boundaries used in the DeepFool method. We evaluated the algorithm on a number of model sets over MNIST and CIFAR-10. We found that the algorithm consistently produces small perturbations in all the cases we examined. Perhaps the most interesting result is that small adversarial perturbations are present even when a non-transferable adversarial example was generated for the most robust model in the set, despite the fact that the models differed only in the regularization coefficient. The generalization of the method and making improvements to its convergence speed are under way.

References

[1] Ian J. Goodfellow and Jonathon Shlens Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd Intl. Conf. on Learning Representations (ICLR)*, 2015.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd Intl. Conf. on Learning Representations (ICLR)*, 2014.

[3] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, June 2016.

[4] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th Intl. Conf. on Learning Representations (ICLR)*, 2018.

[6] Florian Tramer, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th Intl. Conf. on Learning Representations (ICLR)*, 2018.

[7] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Networks and Learning Syst.*, 30(9):2805–2824, Sep. 2019.

[8] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. 5th Intl. Conf. on Learning Representations (ICLR)*, 2017.

[9] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *Proc. of the 36th Intl. Conf. on Machine Learning, (ICML)*, pages 4970–4979, 2019.

[10] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *Proc. 6th Intl. Conf. on Learning Representations (ICLR)*, 2018.

[11] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, November 1998.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[13] Jimmy Ba and Diederik Kingma. Adam: A method for stochastic optimization. In *3rd Intl. Conf. on Learning Representations (ICLR)*, 2015.