THE TABOO TRAP: BEHAVIOURAL DETECTION OF ADVERSARIAL SAMPLES

A PREPRINT

Ilia Shumailov[*]
Computer Laboratory
University of Cambridge
ilia.shumailov@cl.cam.ac.uk

Yiren Zhao*
Computer Laboratory
University of Cambridge
yiren.zhao@cl.cam.ac.uk

Robert Mullins
Computer Laboratory
University of Cambridge
robert.mullins@cl.cam.ac.uk

Ross Anderson
Computer Laboratory
University of Cambridge
ross.anderson@cl.cam.ac.uk

November 26, 2019

ABSTRACT

Deep Neural Networks (DNNs) have become a powerful tool for a wide range of problems. Yet recent work has found an increasing variety of adversarial samples that can fool them. Most existing detection mechanisms against adversarial attacks impose significant costs, either by using additional classifiers to spot adversarial samples, or by requiring the DNN to be restructured. In this paper, we introduce a novel defence. We train our DNN so that, as long as it is working as intended on the kind of inputs we expect, its behavior is constrained, in that some set of behaviors are taboo. If it is exposed to adversarial samples, they will often cause a taboo behavior, which we can detect. Taboos can be both subtle and diverse, so their choice can encode and hide information. It is a well-established design principle that the security of a system should not depend on the obscurity of its design, but on some variable (the key) which can differ between implementations and be changed as necessary. We discuss how taboos can be used to equip a classifier with just such a key, and how to tune the keying mechanism to adversaries of various capabilities. We evaluate the performance of a prototype against a wide range of attacks and show how our simple defense can defend against cheap attacks at scale with zero run-time computation overhead, making it a suitable defense method for IoT devices.

1 Introduction

Deep Neural Networks (DNNs) are being built into ever more systems. Applications such as vision [10], language processing [3] and fraud detection [20] are now using them. Some of these applications are starting to support safety-critical missions. Unfortunately, recent research has discovered that surprisingly small changes to inputs, such as images and sounds, can cause DNNs to misclassify. In the new field of adversarial machine learning, attackers craft small perturbations that are not perceptible by humans but are effective at tricking DNNs. Such adversarial samples can pose a serious threat to systems that use machine learning techniques: examples range from tricking a voice or face recognition system to break into stolen smart phones [1], to changing road signs and street scenes so that human drivers will interpret them one way while autonomous cars or drones read them differently [4].

Current adversarial sample detection has two main approaches. The first is to train additional classifiers to distinguish between clean and adversarial samples [2, 15]. The second is to restructure or augment the original network topology [6]. These mechanisms detect a range of attacks, but are not available off-the-shelf and are not infallible. Both

[*]Equal contribution

introduce extra design complexity and computational cost – which can be particularly difficult for low-cost or energy-limited IoT devices. Low-cost attacks can be surprisingly portable [25], so expensive defense mechanisms would impose additional costs on a wide range of products.

In this paper, we introduce a novel method of detecting adversarial samples, named the *Taboo Trap*. As before, we detect them by observing the classifiers' behaviour. However, rather than learning the expected behaviour of a classifier we wish to protect, and then trying to harden it against adversarial samples, we restrict its behaviour during training and detect the unexpected reactions that such samples cause.

To set a Taboo Trap, we first profile the activation values and choose a transform function to apply to them so they can be limited to specific ranges. We then re-train the network to get one with the restricted activations we chose. Finally, we use a detection function that rings the alarm if any activation breaks a taboo by falling outside the expected ranges. In a simple implementation, the transform function can be simply clipping activation values, which adds almost zero computation overhead, and is effective at rejecting simple attacks.

In this paper we make the following contributions:

- We propose a conceptually different defense strategy, namely restricting DNN behaviour during training so as to detect adversarial inputs later;
- We evaluate our proposed detection mechanism using a range of adversarial attacks and DNNs, demonstrating that it has a low run-time overhead;
- We show that the changes made to the training process do not impair the convergence and performance of the tested networks.

2 Related Work

Szegedy et al. [22] first pointed out that DNNs are vulnerable to adversarial samples and that, in many cases, humans cannot distinguish between clean and adversarial samples. That paper sparked off an arms race between adversarial attack and defense. Defense mechanisms come in a wide variety of flavours. The earliest defense, *adversarial training*, was proposed by Szegedy et al. themselves; they noticed that, when they fed adversarial samples into the training process, the classifier became more robust [22]. Szegedy et al. found however that adversarial samples often transfer between models, enabling *black-box attacks* where a sample trained on one model is used to attack another. Variations on this theme were then explored, Tramer et al. found simple attacks that defeat the basic, single-step, version of adversarial training. One type involves a random perturbation to escape local minima, followed by a linearisation step [23]. They proposed *ensemble adversarial training* in which a model is trained against adversarial samples generated on a variety of networks pre-trained from the same data. They also described *gradient masking* as a defense mechanism, noting that gradient-based attacks need to measure a gradient and making this hard to do accurately stops the attacks from working [23].

The alternative defense approach is adversarial sample detection. Conceptually, such mechanisms are like intrusion detection systems; they provide situational awareness, which in turn can enable a flexible defense. Metzen et al. were among the first to study adversarial sample detection [17]. They augmented the original classifier with an auxiliary network which they trained to classify the inputs as either clean or adversarial, and concluded that it could detect adversarial samples. Grosse et al. proposed two other approaches [6]. First, they found feasible to use statistical test to distinguish adversary from legitimate sample distribution data distribution; but such test was compute demanding in practice. Second, they found that instrumenting the network with an additional class designed for adversarial sample detection is able to work more efficiently. Meng and Chen focused on detecting adversarial samples using an additional classifier [16] and designed the MagNet system that consists of a detector network and a reformer network. Lu et al. presented SafetyNet – another way of instrumenting a classifier with an adversarial sample detector [15]. It replaces the last layers of the classifier with a quantised ReLU activation function and uses an RBF-based SVM to classify the activation patterns. Li et al. invented another detection method based on observation of the last-layer outputs of convolutional neural networks [14]. They built a cascade classifier to detect unexpected behaviours in this last layer.

All of the above defences, however, have at least one of the following shortcomings. First, some require additional classifiers [14, 17], which means extra computation at runtime. Second, some require restructuring the original neural network [15,16], which means extra design complexity and possible degradation of functionality. Our goal is to build a detection mechanism for IoT devices that requires no additional classifiers, does not impair the network's performance in the absence of adversarial samples, and can detect them dependably with almost zero computational overhead.

3 Methodology

3.1 The Taboo Trap

The intuition behind the Taboo Trap is simple. We train the DNN to have a restricted set of behaviours on activations that are hidden from the attackers during training, and report any behaviour that later violates these restrictions as an adversarial sample. We first profile activation values for all samples (N) on the training dataset across different layers (L), $A \subset \mathbb{R}^{N \times L \times X \times Y \times C}$. Each $A_{l,n}$ is a three-dimensional tensor which is a collection of feature maps, $A_{n,l} \subset \mathbb{R}^{X \times Y \times C}$, where X, Y and C are the feature map's width, height and number of channels respectively. To set a Taboo Trap, we need to define a transform function f_t on all activations. During the training phase, we combine the output of the transform function, considering only a batch of data (B):

$$L = L_{SGD} + \lambda \sum_{n=1}^{B} f_t(A_n) \tag{1}$$

λ is a hyper-parameter and L_{SGD} is the loss from Stochastic Gradient Descent (SGD). In the third stage, the output of $f_t(A_n)$ translates to the detection result:

$$\text{Detected} = \begin{cases} \text{True}, & \text{if } f_t(A_n) \geq 0 \\ \text{False}, & \text{otherwise} \end{cases} \tag{2}$$

To give an example, we study one of the many possible transform functions – the maximum percentile function. We will return to the discussion of what makes a good transform function, and how to diversify them, in Section 5 on page 5.

Consider the activation values, for each layer l and sample n, we have $A_{n,l} \subset \mathbb{R}^{X \times Y \times C}$, where X, Y and C are the feature map's width, height and number of channels respectively. We use $\max(A) \subset \mathbb{R}^{N \times L}$ to represent the profiled maximum activations of N samples on L layers; effectively, $\max(A_{n,l})$ is a single value that is the maximum of $A_{n,l}$. The profiled maximum activation $\max(A_{[1:N],l})$ is a vector of length N, where N is the number of samples. We first calculate a threshold value for a particular layer:

$$\alpha_l = g(\max(A_{[1:N],l})) \tag{3}$$

We design a function $\alpha_l = g(\max(A_{[1:N],l})) = \mathsf{Percentile}_n(\max(A_{[1:N],l}))$, where $\mathsf{Percentile}_n$ computes the nth percentile of the function input. We then perform fine-tuning on the DNN with a regularizer that penalizes activations larger than the profiled thresholds. We then design the transform function (f_t) in the following form:

$$f_t(A_n) = \\ \lambda \sum_{l=0}^{L-1} \sum_{x=0}^{X_l-1} \sum_{y=0}^{Y_l-1} \sum_{c=0}^{C_l-1} f_p(A_{n,l,x,y,c}, \alpha_l) \tag{4}$$

We define f_p to be:

$$f_p(a, b) = \begin{cases} 1, & \text{if } a \geq b \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

After finding the number of activations in each layer that are over-excited, we try to minimize their values using SGD. The regularizer provides an extra loss to the loss function of the neural network and λ is a hyperparameter used to control how hard we are penalizing the neural network for having large activation values. We call λ the alarm rate.

Finally, we use a detection function when we deploy the model. For any given input n, we check the activation values that the neural network produces at every layer. If any of them is greater than the threshold value for that particular layer, we recognize that particular input as adversarial.

3.2 Training Strategy

When training with Taboo Trap, we find that it is necessary to gain some model diversity when re-training the models. Fine-tuning from a pretrained model can lead to a suboptimal outcome when balancing the cost from the taboo regularization term against the model cost. We summarize our training strategy in Algorithm 1, which effectively is about balancing the training hyperparameters.

Algorithm 1: Taboo Trap instrumented training process

```
Data: $\lambda$ = Alarm Rate, $\epsilon$ = Learning Rate, $l$ = loss
Result: WTC instrumented network with comparable performance
Pick initial values for Alarm Rate and Learning Rate;
while Detection Rate on test data $> 0$ do
    Retrain with $\lambda$ and $\epsilon$ and collect loss $l$ for a number of epochs.
    if $l$ is not decreasing then
        Increase $\lambda$
        Decrease $\epsilon$
    end
end
```

		LeNet5 – MNIST			**M-CifarNet – CIFAR10**			**ResNet18 – CIFAR10**		
	θ	A	D	AD	A	D	AD	A	D	AD
Baseline		0.99	0	0.02	0.89	0	0.02	0.95	0	0.02
FGSM	$\epsilon = 0.02$	0.9	0.07	0.06	0.04	0.04	0.06	0.6	0.01	0.08
	$\epsilon = 0.04$	0.64	0.27	0.21	0.02	0.06	0.09	0.35	0.16	0.18
	$\epsilon = 0.08$	0.12	0.73	0.68	0.01	0.3	0.3	0.08	0.91	1
	$\epsilon = 0.1$	0.05	0.88	0.84	0.01	0.64	0.72	0.06	0.92	1
PGD	$\epsilon = 0.07$	0	0.63	0	0	0.79	0	0.01	0.94	0.97
	$\epsilon = 0.5$	0	0.64	0	0	0.79	0	0.01	0.94	0.9
	$\epsilon = 1.0$	0	0.64	0	0	0.79	0	0.01	0.94	0.9
DeepFool	$i = 1$	0.87	0.07	0.11	0.45	0.03	0.03	0.71	0.02	0.15
	$i = 3$	0.1	0.13	0.21	0.02	0.03	0.04	0.4	0.03	0.46
	$i = 5$	0	0.14	0.05	0	0.03	0	0.26	0.05	0.71

Table 1: The max n-th percentile detector evaluation of LeNet5 (MNIST), M-CifarNet (CIFAR10), ResNet18 (CIFAR10). A means the accuracy on generated adversarial samples, D means the detection rate on adversarial samples that have fooled the models and AD represents the detection rates on adversarial samples that are correctly classified by the models. Detection ratios on clean evaluation data (false positives) in all cases are controlled less than 1%.

4 Evaluation

4.1 Attacks, Networks and Datasets

We consider the Fast Gradient Sign Method (FGSM) [5], Projected Gradient Descent [11] and Deepfool [18] attacks to evaluate the Taboo Trap. We used the implementations of the attacks above from Foolbox [21] and implemented our defense mechanism in Pytorch [19]. At the time of writing, FGSM is generally considered to be a weak adversary, PGD a slightly stronger one and DeepFool the strongest attack.

We use LeNet5 [13] on MNIST [12]. The LeNet5 model has 431K parameters and classifies MNIST hand-written digits with an accuracy of 99.17%. For the CIFAR10 [9] datasets, we use M-CifarNet [24] and ResNet18 [7] to perform classifications. The M-CifarNet classifier [24] has 1.3M parameters and achieves 89.48% classification accuracy, and our ResNet18 model achieves 93.83% accuracy on CIFAR10.

4.2 Max *n*th Percentile Detector

Table 1 shows the performance of the Taboo Trap against three different attackers. All of the networks have shown a good detection rate (D and AD) for FGSMs with a relatively high ϵ. Interestingly, LeNet5 on MNIST and ResNet18 on CIFAR10 have shown good accuracy for small values of ϵ, whereas M-CifarNet have not. Both models have in common that they try to solve a rather complex task with a much smaller architecture capability. The Taboo Trap shows a relatively weak performance on a relatively strong attack (PGD) and a lot worse performance against a strong attack (DeepFool). Similar to FGSM, the performance of the detector was better in models with higher capacity for a given dataset.

To summarize, the nth percentile detector has shown relatively good detection rates against weak attackers with the rates decreasing as the attacks become stronger. Finally, the detector's performance and adversarial recovery effect observed seems to be connected with the general capacity of the chosen DNN to solve a given task.

FGSM with a relatively large epsilon value can be seen as a simple attack that is easy to execute at scale in practice. It is a single-step method and a large epsilon makes the adversarial samples transferable through different networks. Iterative methods, such as DeepFool and PGD with binary search, can break our defense. However, these attacks are usually performed with a strong assumption that attackers can iteratively optimize the adversarial samples, so detecting these expensive attacks is not the aim of our detection method. As we have demonstrated in our results, we can efficiently reject adversarial samples generated by cheap attacks with high detection ratios, especially on adversarial samples that have successfully fooled our models (the D measurement in Table 1 on the previous page).

4.3 Efficiency Comparison

As mentioned previously, we designed Taboo Trap with efficiency in mind, aiming at rejecting cheap attacks at scale. Table 2 shows the computational performance and also defense efficiency when compared to MagNet [16] and SafetyNet [15]. Since Taboo Trap with maximum percentile as a transfer function is only a value comparison in computation, it does not either introduce extra compute or extra parameters. Existing adversarial detection mechanisms have surprisingly large overheads, some of them even have the defense networks to be multiple times larger than the original classification network. In the case of edge devices, when compute resources are limited and the running networks are small, a large increase in compute and memory footprint caused by defense is infeasible. We realize Taboo Trap performs worse on DeepFool compared to other methods. However, we used a simple transfer function, a more advanced transfer function might aid detection ratios. Also, we highlight the purpose of designing Taboo Trap is to defeat simple attacks at scale and Deepfool as an iterative attacking method should not be considered as a cheap attack. Taboo Trap detects simple attack like single-step FGSM as well as other advanced detection methods (Table 2).

Technique	θ	FGSM	DeepFool	New MACs	as %	New Params
Taboo Trap	P_1	0.99	0.14	**0**	**0**	**0**
MagNet [16]		1.0	1.0	190K	20	297
SafetyNet [15]	1 layer	0.92	0.82	34.81M	3622	5,95M
	3 layers	0.98	0.92	21.77M	2265	4.86M

Table 2: LeNet5 has 961K multiply-accumulate operations (MACs) and 17.75K parameters without defenses. SafetyNet is trained on the DeepFool adversarial samples. P_i refers to the ith-percentile. FGSM $\epsilon = 0.8$ is used. Overhead for MagNet is only calculated over the detector.

5 Discussion

In 1883, the cryptographer Auguste Kerckhoffs enunciated a design principle that has stood the test of time: a system should withstand enemy capture, and in particular it should remain secure if everything about it, except the value of a key, becomes public knowledge [8]. Here, we have studied the performance of using the nth maximum percentile as a simple transform function. This can be applied to any randomly chosen subset of the activation values, and gives plenty of opportunity to add the equivalent of a cryptographic key.

In the grey-box setup, attackers might be aware of the deployment of the taboo trap classifier but remain ignorant of the chosen transform function. In a white-box scenario, attackers might construct samples that trick the defence employed – they can try to feed in various inputs to figure out the transform function. The defence against white-box attacks is changing the key – for example, to combine a series of transform functions that work on different activation values or different ranges. That way, a vendor can produce two different versions of his product, one for the mass market and one for industrial use; an attacker who develops adversarial samples for the former will not be able to use them on the latter.

So how do we choose good keys? We have experimented with a number of different transform functions and have observed them exhibiting a variety of defensive capabilities.

To demonstrate that behavioral defense can work with different transform functions, Table 3 shows the defensive capabilities of LeNet5 (MNIST) trained with three different transform functions, f_1 is the maximum 1st percentile ($[0 : \alpha]$), where the threshold α is different for each layer since it is per-layer profiling dependent. f_2 is restricted to

Attack	θ	f_1	f_2	f_3
FGSM	$\epsilon = 0.4$	0.94	0.97	**0.98**
PGD	$\epsilon = 0.07, i = 5$	**0.94**	0.92	0.61
DeepFool	$i = 5$	0.01	**0.12**	0.06

Table 3: Taboo Trap on LeNet5 with three different transform functions: f_1 and f_3 on all layers; f_2 on the first.

$(0 : 1]$ and applied only on the first layer. f_2 is restricted to $[0 : 1], [2 : 3], [4 : 5]$ and applied on all layers. All of the networks were trained to have an accuracy of around 99% with a false positive rate of less than 1%. As can be seen in the table, the performance of different attacks on various keys differs from each other. This suggests that each of the attacks focuses on exploiting a particular layer within a particular range, and so a defender can choose different families of transform functions to block different adversaries.

The diversity and heterogeneity of the potential defensive transform functions gives the classic solution to the white-box attack. They make defense unpredictable, and the defender can respond to both the expectation and the experience of attacks by changing a key. The key selects from among different transform functions and will also add non-determinism to grey-box attacks, where the transform functions are known but the parameters are not.

We designed Taboo Trap for deployment on low-cost hardware where computation may be severely limited, for example by battery life. However, nothing stops the defender from using classifiers with Taboo Traps in conjunction with other strategies where the computation budget permits this. As our approach does not change the network structure or add any other additional components to the pipeline it is easily combined with other, more expensive, defensive systems like MagNet [2] or SafetyNet [15].

6 Conclusion

In this paper we presented the Taboo Trap, a radically new way to detect adversarial samples that is both simple and cheap. The run-time compute overhead of Taboo Trap is close to zero, with the only additional cost being a slight increase in the time taken to train the network. We evaluated our simple mechanism and showed that it performs well against a range of popular attacks. The simple nth percentile transfer function that we tested did not perform as well as more complex detection mechanisms such as MagNet but still provides a very useful tool in the defender's armoury.

In addition to simplicity and low cost, the Taboo Trap offers diversity in defence. It can be used with a wide variety of transfer functions, which perform much the same function that key material does in cryptographic systems: the security of a system need not reside in keeping the basic design secret, but in the secrecy of parameters that can be chosen at random for each implementation. This opens the prospect of extending defences against adversarial machine-learning attacks from black-box applications to grey-box and even white-box adversaries.

Acknowledgements

Partially supported with funds from Bosch-Forschungsstiftungim Stifterverband

References

[1] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou. Hidden Voice Commands. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016.

[2] N. Carlini and D. A. Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017.

[3] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[4] 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.

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

[6] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017.

[7] 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*, pages 770–778, 2016.

[8] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires, vol. IX*, pages 161–191, 1883.

[9] A. Krizhevsky, V. Nair, and G. Hinton. The CIFAR-10 dataset. 2014.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[12] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. 2, 2010.

[13] Y. LeCun et al. LeNet-5, convolutional neural networks. page 20, 2015.

[14] X. Li and F. Li. Adversarial examples detection in deep networks with convolutional filter statistics. *CoRR*, abs/1612.07767, 2016.

[15] J. Lu, T. Issaranon, and D. A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly.

[16] D. Meng and H. Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 135–147, New York, NY, USA, 2017. ACM.

[17] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[18] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks. 2016.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[20] R. Patidar, L. Sharma, et al. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2011.

[21] J. Rauber, W. Brendel, and M. Bethge. Foolbox: a python toolbox to benchmark the robustness of machine learning models (2017). *URL http://arxiv. org/abs/1707.04131*, 2017.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[23] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[24] Y. Zhao, X. Gao, R. Mullins, and C. Xu. Mayo: A framework for auto-generating hardware friendly deep neural networks. 2018.

[25] Y. Zhao, I. Shumailov, R. Mullins, and R. Anderson. To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression. *arXiv preprint arXiv:1810.00208*, 2018.