The Taboo Trap: Behavioural Detection of Adversarial Samples

Ilia Shumailov*, Yiren Zhao*, Robert Mullins, Ross Anderson
University of Cambridge
UK
name.surname@cl.cam.ac.uk

Abstract

Deep Neural Networks (DNNs) have become a powerful tool for a wide range of problems. Yet recent work has shown an increasing variety of adversarial samples that can fool them. Most existing detection mechanisms 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 a set of behaviors are taboo. If it is exposed to adversarial samples, they will often cause a taboo behavior, which we can detect. As an analogy, we can imagine that we are teaching our robot good manners; if it's ever rude, we know it's come under some bad influence. This defence mechanism is very simple and, although it involves a modest increase in training, has almost zero computation overhead at runtime – making it particularly suitable for use in embedded systems. Taboos can be both subtle and diverse. Just as humans' choice of language can convey a lot of information about location, affiliation, class and much else that can be opaque to outsiders but that enables members of the same group to recognise each other, so also taboo choice can encode and hide information. We can use this to make adversarial attacks much harder. It is a well-established design principle that the security of a system should not depend on the obscurity of its design, but of some variable (the key) which can differ between implementations and be changed as necessary. We explain how taboos can be used to equip a classifier with just such a key, and 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 work well in practice.

1. Introduction

Deep Neural Networks (DNNs) are being built into ever more systems because of their outstanding performance on a wide variety of classification tasks. Applications such as vision [11], language processing [3] and fraud detection [21] are now using them. As these applications start to include safety-critical missions such as vehicle guidance, we need to know whether they are reliable and secure. Unfortunately, recent research has discovered that surprisingly small changes to inputs, such as images and sounds, can cause DNNs to misclassify them. 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 a variety of 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 unmanned drones read them differently [4].

Current adversarial sample detection mechanisms have two main approaches. The first is to build additional classifiers that distinguish between clean and adversarial samples [16, 2]. The second is to restructure or augment the original network topology [7]. These mechanisms can detect a range of attacks, but are not available off-the-shelf and are not infallible. Both introduce extra design complexity and computational cost – which can be particularly difficult to bear in the case of IoT devices.

Here we introduce a conceptually different method of detecting adversarial samples, which we call the *Taboo Trap*. As in previous approaches, we detect them by observing the behaviour of the classifiers. However, while most previous approaches learned the expected behaviour of the classifier, we restrict its behaviour and detect unexpected reactions caused by adversarial samples.

This work was motivated by the observation made by Zhao et al. that adversarial attacks are less transferable with low-bit integer quantisation [28]. An intuitive explanation is that when an activation representation is limited in mag-

* These authors contributed equally.

nitude, the adversary will struggle to over-drive activations to cause a misclassification.

An intuitive analogy is that we train our robots to behave well, and if one of them is suddenly starts being rude to us, we know it has fallen under some bad influence – as when we educate children to be polite, and yet they have been influenced by inappropriate words in the playground and use them at the dinner table. By 'inappropriate' we do not just mean anatomical swearwords or discriminatory epithets; everyday vocabulary choice coveys a lot of information about social group membership, political views and so on [23]. (The analogy is slightly stretched, as in a DNN we can measure not just the output but also the activations of intermediate layers; it is as if we can detect a subverted child not just when they speak an inappropriate word, but even when they think one.)

To set a taboo trap, we first profile the activation values and decide what transform function to apply on the activations. This function is then used to restrict the activations to specific ranges based on the profiled information. We then re-train the network and get a network where activations adhere to the resulting limitations. 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.

In this paper we make the following contributions:

- We observe that one can restrict DNN behaviour in such a way as to facilitate detection of adversarial inputs.
- We evaluate the proposed detection framework using a range of adversarial attacks and different DNNs.
- We show that the changes made to the training process do not impair the convergence and performance of the tested networks.

In section 2, we describe existing attack and defense mechanisms. We describe our Taboo Trap and the experimental setups in section 3 on the next page and evaluate its performance in section 4 on page 4.

2. Related Work

Adversarial machine learning studies how an adversary can exploit machine learning models and what can be done to block or mitigate such attacks. This is a fast-moving field with a rapidly-growing literature, and of necessity our summary here is somewhat telegraphic.

The seminal paper was by Szegedy et al. [24] who first pointed out that DNNs are vulnerable to adversarial samples and that, in many cases, humans cannot distinguish between clean and adversarial samples – the generated perturbations are imperceptible.

That paper sparked off an arms race between adversarial attack and defense. Defence mechanisms now come in a wide variety of flavours. The earliest, *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 to them [24]. They 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, until in 2017 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; another is a *black-box attack* [26]. 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 defence mechanism, noting that gradient-based attacks need to measure a gradient and making this hard to do accurately stops the attacks from working [26]. Those defences, however, were soon shown to be vulnerable to black-box attacks.

The new approach is adversarial sample detection. The idea is that, in addition to the original neural network, another mechanism is constructed to detect whether an input image is adversarial. Conceptually, such mechanisms are like intrusion detection systems; they provide situational awareness, which in turn can enable a flexible defence. For example, if one can detect that a black-box approximation attack is underway then one can start injecting false data to defeat it.

Metzen et al. were among the first to study adversarial sample detection [18]. They augmented the original classifier with an auxiliary network which they trained to classify the inputs as either clean or adversarial. They evaluated this against dynamic and static adversaries, and concluded that it could detect adversarial samples.

Grosse et al. proposed two other approaches [7]. First, they explored whether a statistical test could distinguish adversarial from legitimate sample distribution data distribution; they found it was feasible but not efficient in practice. Second, they found that instrumenting the network with an additional class designed for adversarial sample detection is able to work more reliably.

Meng and Chen focus on detecting adversarial samples using an additional classifier [17]. Their MagNet system typically uses two or more different detector networks and a reformer network; the detector networks learn to distinguish between normal and adversarial inputs by measuring their distance from an expected manifold. The reformer network

uses auto-encoders to move adversarial samples towards the manifold of legitimate ones, so as to stop them fooling the detectors. Although MagNet defeats black-box attacks, it was found to be vulnerable to white-box attacks (where the attacker knows the model's parameters); but its creators claim that using multiple different autoencoders can defeat white-box (or at least grey-box) attacks since the defense behaviour becomes too unpredictable. This appears to be the first paper to invoke the concept of cryptographic diversity as a defence.

Lu et al. presented SafetyNet – another way of instrumenting a classifier with a adversarial sample detector [16]. 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. They found that this allows them to detect adversarial samples reliably and note that there might exist even better objective functions.

Li et al. invented another detection method based on observation of the last-layer outputs of convolutional neural networks [15]. They built a cascade classifier to detect unexpected behaviours in this last layer.

There has also been research on using Bayesian methods to estimate the level of uncertainty at dropouts and thus detect adversarial samples [5] or applying transformations to inputs to make detection easier [25].

All of the above-mentioned defences, however, have at least one of two shortcomings. First, some require additional classifiers [18, 15], which means extra computation when deploying the models. Second, some require restructuring the original neural network [16, 17], which means extra design complexity and a possible degradation of functionality.

Our goal is to design a detection mechanism 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. Adversarial Attacks

We decided to use the following attacks to evaluate the taboo trap:

- Fast Gradient Sign Method (FGSM) [6];
- Iterative Method [12];
- Deepfool Method [19].

The fast gradient sign method (FGSM), introduced by Goodfellow et al., creates perturbations using the sign of the gradients of a DNN [6]. The iterative method (BIM) of Kurakin et al. is an iterative version of this: adversarial samples are used as inputs to FGSM and accumulate perturbations iteratively [12]. Deepfool, by Moosavi et al., is also an iterative method. It aims to minimize the changes required to push a DNN across decision boundaries, by iteratively perturbing an input image and linearlizing the classification space to move it to the closest decision boundary [19]. We used the implementations of the attacks above from Foolbox [22] and implemented our defense mechanism in Pytorch [20].

At the time of writing, FGSM is generally considered to be a weak adversary, BIM a slightly stronger one and DeepFool the strongest attack.

3.2. The Taboo Trap

3.2.1 Behavioural Detection

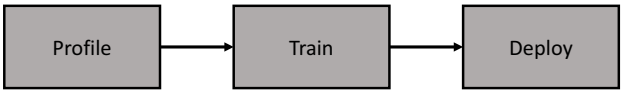


Figure 1: A 3-stage detection pipeline.

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 unexpected behaviour thereafter as an adversarial sample.

We illustrate this as a three-stage process in Figure 1. 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):

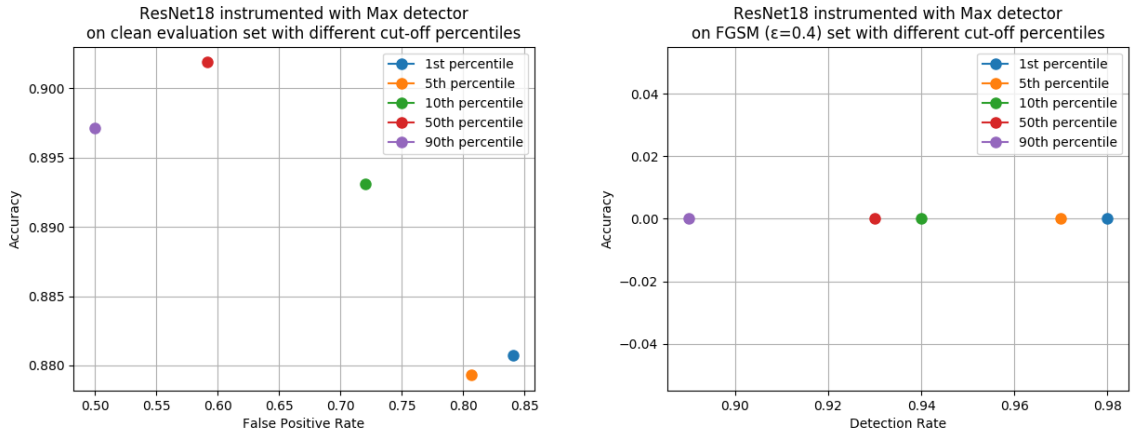
$$L = L_{SGD} + \lambda \sum_{n=1}^{B} f_t(A_n) \tag{1}$$

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:

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

3.2.2 Using nth Percentile as Transform Function

In this section, we study one of the many possible transform functions – the maximum percentile function. We will



(a) Clean data

(b) FGSM ($\epsilon = 0.4$) data

Figure 2: Predictive accuracy with different percentiles used with Max detector of ResNet18 on CIFAR10 dataset.

return to the discussion of what makes a good transform function, and how to diversify them, in Section 5 on page 6.

Considering the activation values of each layer, we have a collection of values, and 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 $maxA \subset \mathbb{R}^{N \times L}$ to represent the profiled maximum activations of N samples on L layers; effectively, $maxA_{n,l}$ is a single value that is the maximum of $A_{n,l}$. For each layer l, the profiled maximum activations of all samples on the training dataset are represented as $maxA_{[1:N],l} \subset \mathbb{R}^N$. The profiled maximum activation ($maxA_{[1:N],l}$) is a vector of length N, where N is the number of samples. We first calculate threshold value for a particular layer:

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

We design a function $\alpha_l = g(\max A_{[1:N],l}) = \text{Percentile}_n(\max A_{[1:N],l})$, where 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:

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

In the second stage, we find the number of activations in each layer that are over-excited and 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 hyper-parameter used to control how hard we are penalizing the neural network for having large activation values.

In the third phase, 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.

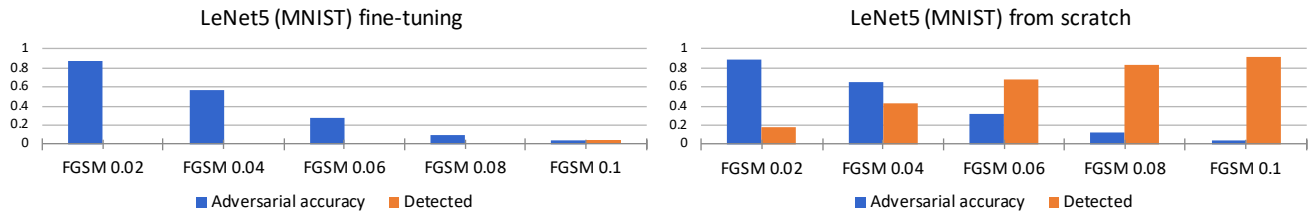
3.3. Networks and Datasets

We use LeNet5 [14] on MNIST [13]. The LeNet5 model has 431K parameters and classifies MNIST hand-written digits with an accuracy of 99.17%. For the CIFAR10 [10] datasets, we use M-CifarNet [27] and ResNet18 [8] to perform classifications. The M-CifarNet classifier [27] has 1.3M parameters and achieves 89.48% classification accuracy, and our ResNet18 model achieves 93.83% accuracy on CIFAR10 and 76.76% accuracy on CIFAR100.

4. Evaluation

4.1. Training, Detection and Classification trade-off

The additional loss introduced through the regularization term makes training more challenging. First, even though the network usually converges to the same accuracy in the same number of epochs compared to the original network without such regularization, the false positive rate remains high. By this we refer to the number of images that are detected as adversarial but are actually clean inputs. In order to remove them, we need to train the network for a few more additional epochs with a small learning rate and a large λ. This effectively forces the weights to make sure they fall in



(a) LeNet5 fine-tuned from a pretrained model.

(b) LeNet5 trained from scratch.

Figure 3: LeNet5 Accuracy and detection rates on adversarial samples with regularization but different training strategies.

the expected range. Second, detection rates depend heavily on how aggressive the thresholding values are; normally more aggressive thresholds also indicate more fine-tuning epochs. Intuitively, aggressive thresholds require weights to adjust more to provide good detection performance.

To explore the performance / defence trade-off we trained the networks from scratch (why, will be explained in section 4.2) for the same number of epochs. Note, we did not optimise the networks until their respective best performance so we could explore the relative performance. Figure 2 on the previous page shows the predictive accuracy of ResNet18 with different max-percentile thresholds each trained for:

- Epochs 20, learning rate 0.1, alarm rate 5
- Epochs 20, learning rate 0.01, alarm rate 5
- Epochs 20, learning rate 0.01, alarm rate 100

We used the models that reported the lowest false positive rate and highest training accuracy between training iterations.

Figure 2a on the preceding page shows the performance of the classifiers on clean data (y-axis, Accuracy) and percentage of the clean samples that were classified correctly but actually raised the alarm (x-axis, False Positive Rate). It can be seen that the networks with less aggressive thresholds ended up having relatively large accuracy. More aggressive thresholds indicate large regularization loss which impacts training quality and produces larger false positive rates. Similar behavior is observed for all of the networks we trained.

Note that this section only focuses on analyzing the trade-offs, and only has a limited number of epochs spent on fine-tuning. For all of the networks used, we could train them to near original accuracy and near zero false positive rate.

4.2. Training From Scratch versus Fine-tuning

When training with a regularizer, there are two methodologies to consider. One is simply to train the network again from scratch, and the other one is using the existing pre-trained model as a starting point for fine-tuning. We find that there is a big difference in detection rates between the two. Figure 3a and Figure 3b show the performance of LeNet5 when trained with the same hyperparameters, fine-tuned and from scratch respectively.

It can be seen that the detection rates are significantly larger with small FGSM ϵ values.

Intuitively, when training the network with a large regularizer, the weights have to optimize two losses – the regularization loss and the classification loss. For fine-tuning, the former dominates the cost function thus the model struggles to balance between the two losses and makes convergence challenging. In contrast, when training from scratch, the regulariser loss is so large that it provides heavy restrictions on the initialized weights. So we first see a quick drop in regularization loss and the classification loss then starts to decrease slowly. Ultimately, both processes can provide similar performance, however, fine-tuning approaches are restricted to a smaller set of regularization hyper-parameters and it almost always reduce the large regularzation loss first which completely destroys the classification accuracy and gives no advantage over training from scratch.

In summary, we provide the training methodology of the taboo trap in Algorithm 1.

Algorithm 1: Taboo trap instrumented training process

Data: α = Alarm Rate, ϵ = 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 α and ϵ and collect loss l for a number of epochs.
  if l *is not decreasing* **then**
    Increase α
    Decrease ϵ
  end
end

	θ	Classifier	Taboo Trap		
		R	R	D	RE
None		0.94	0.95	0	0.02
FGSM	$\epsilon = 0.02$	0.15	0.6	0.01	0.08
	$\epsilon = 0.04$	0.07	0.35	0.16	0.18
	$\epsilon = 0.06$	0.03	0.14	0.68	0.74
	$\epsilon = 0.08$	0.01	0.08	0.91	1
	$\epsilon = 0.1$	0.01	0.06	0.92	1
	$\epsilon = 0.2$	0.01	0.06	0.91	1
	$\epsilon = 0.4$	0	0.05	0.92	1
	$\epsilon = 0.6$	0	0.03	0.91	1
	$\epsilon = 0.8$	0	0.02	0.92	1
	$\epsilon = 1.0$	0	0.03	0.92	1
BIM	$\epsilon = 0.07$	0	0.01	0.94	0.97
	$\epsilon = 0.5$	0	0.01	0.94	0.9
	$\epsilon = 1.0$	0	0.01	0.94	0.9
DeepFool	$i = 1$	0.7	0.71	0.02	0.15
	$i = 3$	0.15	0.4	0.03	0.46
	$i = 5$	0.01	0.26	0.05	0.71

Table 1: The max nth percentile detector evaluation of ResNet18 on CIFAR10 dataset next to an uninstrumented ResNet18. R means the recovered accuracy on adversarial samples, D means detection ratio on all adversarial samples and RE refers to the percentage of recovered adversarial samples that raised the alarm.

4.3. Regularization Effect

Table 1 shows the performance of ResNet18 on the CIFAR10 dataset with and without taboo trap. By restricting the distribution of activation values, one also gets better regularization and the base classifier becomes more resistant to adversarial samples. The difference becomes apparent with a strong DeepFool attack with 3 steps – 15% accuracy without versus 40% with Taboo Trap. Additionally, taboo trap now recognizes 25% more samples on the adversarial samples, and detected 44% of the adversarial samples that were misclassified.

4.4. Max nth percentile detector evaluation

Table 2 shows the performance of the taboo trap against three attackers with different levels of capabilities. The first thing that becomes apparent is that all of the networks have shown a good detection rate for FGSMs with a relatively high ϵ. Interestingly, LeNet5 on MNIST and ResNet18 on CIFAR10 have shown good recovered accuracy for small values of ϵ, whereas M-CifarNet and ResNet18 on CIFAR100 have not. Both models have in common that they try to solve a rather complex task with a much smaller architecture capability. This suggests that finding reported in Section 4.3 can be made even stronger given a more capable architecture. The taboo trap shows a relatively weak performance on a relatively strong attack (BIM) 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.

5. Discussion

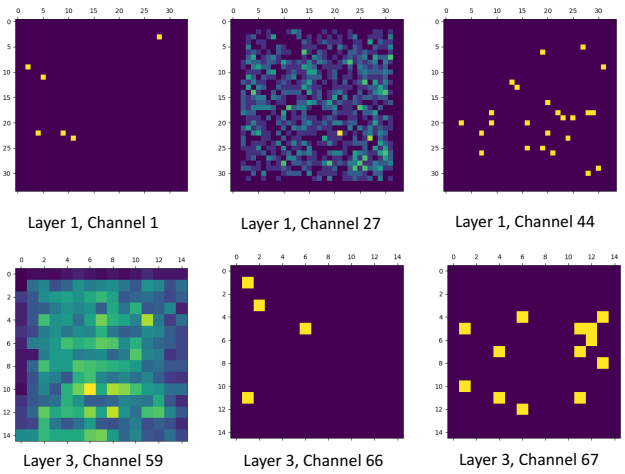


Figure 4: Visualization of pixels that the Taboo Trap reported as over-excited. The underlying neural network is M-CifarNet with FGSM ($\epsilon = 0.8$) attack.

What makes our work particularly important is that we have presented a simple, low-cost way to defend networks against GreyBox attacks, and a combination of transforms and behaviour-level constraints may extend this protection to BlackBox/WhiteBox attacks.

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 everyhing about it, except the value of a key, becomes public knowledge [9]. Here, we have studied the performance of using the nth maximum percentile as a simple transform function. However, transform functions could be any transformations on the activation values. This gives plenty of opportunity to add the equivalent of a cryptographic key. In the GreyBox setup, attackers might be aware of the deployment of the taboo trap classifier but remain ignorant of the chosen transform function. In a WhiteBox scenario, attackers might construct

		LeNet5 – MNIST			M-CifarNet – CIFAR10			ResNet18 – CIFAR10			ResNet18 – CIFAR100		
	θ	R	D	RE	R	D	RE	R	D	RE	R	D	RE
No attack		0.99	0	0.02	0.89	0	0.02	0.95	0	0.02	0.75	0.03	0.06
FGSM	$\epsilon = 0.02$	0.9	0.07	0.06	0.04	0.04	0.06	0.6	0.01	0.08	0.09	0.01	0.02
	$\epsilon = 0.04$	0.64	0.27	0.21	0.02	0.06	0.09	0.35	0.16	0.18	0.04	0.01	0.01
	$\epsilon = 0.06$	0.32	0.52	0.45	0.01	0.12	0.17	0.14	0.68	0.74	0.02	0.01	0
	$\epsilon = 0.08$	0.12	0.73	0.68	0.01	0.3	0.3	0.08	0.91	1	0.01	0.01	0.02
	$\epsilon = 0.1$	0.05	0.88	0.84	0.01	0.64	0.72	0.06	0.92	1	0.01	0.01	0.01
	$\epsilon = 0.2$	0	0.99	0.53	0	0.81	0.68	0.06	0.91	1	0	0.74	0.7
	$\epsilon = 0.4$	0	0.99	0.1	0	0.81	0.53	0.05	0.92	1	0	0.75	0.2
	$\epsilon = 0.6$	0	0.99	0.07	0	0.81	0.53	0.03	0.91	1	0	0.75	0.1
	$\epsilon = 0.8$	0	0.99	0.07	0.01	0.81	0.65	0.02	0.92	1	0	0.75	0.1
	$\epsilon = 1.0$	0	0.99	0.05	0.01	0.82	0.75	0.03	0.92	1	0	0.75	0.1
BIM	$\epsilon = 0.07$	0	0.63	0	0	0.79	0	0.01	0.94	0.97	0	0.18	0
	$\epsilon = 0.5$	0	0.64	0	0	0.79	0	0.01	0.94	0.9	0	0.18	0
	$\epsilon = 1.0$	0	0.64	0	0	0.79	0	0.01	0.94	0.9	0	0.18	0
DeepFool	$i = 1$	0.87	0.07	0.11	0.45	0.03	0.03	0.71	0.02	0.15	0.64	0.01	0.01
	$i = 3$	0.1	0.13	0.21	0.02	0.03	0.04	0.4	0.03	0.46	0.18	0.01	0.01
	$i = 5$	0	0.14	0.05	0	0.03	0	0.26	0.05	0.71	0.02	0.01	0
	$i = 7$	0	0.14	0	0	0.03	0	0.2	0.07	0.82	0	0.01	0
	$i = 9$	0	0.14	0	0	0.03	0	0.16	0.08	0.89	0	0.01	0

Table 2: The max n-th percentile detector evaluation of LeNet5 (MNIST), M-CifarNet (CIFAR10), ResNet18 (CIFAR10 and CIFAR100). R means the recovered accuracy on adversarial samples, D means detection ratio on all adversarial samples and RE refers to the percentage of recovered adversarial samples that raised the alarm.

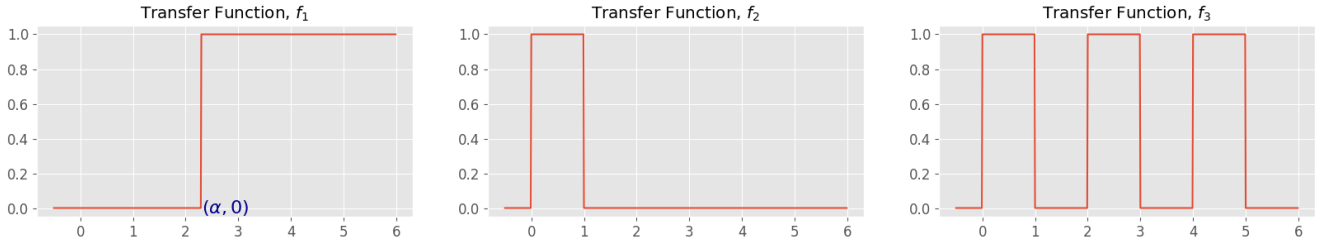


Figure 5: Different transform functions for activations. f_1 is the maximum 1st percentile, where the threshold α is different for each layer since is per-layer profiling dependent. f_2 is restricted to $([0:1])$ and applied only on the first layer. f_2 is restricted to $([0:1],[2:3],[4:5])$ and applied on all layers.

samples that trick the defence employed – they can try to feed in various inputs to figure out the transform function. One potential defence against WhiteBox attacks is changing the key – for example, to combine a series of transform functions that work on conflicting ranges and rotate them with their corresponding models at run-time. The MagNet discussed the possibility of this; the Taboo Trap gives much more scope to do it.

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.

In Figure 4 on the preceding page, we show a heatmap

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

Table 3: Performance of detector on LeNet5 (MNIST) trained with three different transform functions: f_1 and f_3 apply on all layers; f_2 only applies on the first layer.

on the over-threshold pixels on different adversarial samples. This helps us to evaluate the effectiveness of a trans-

form function. We apply the attack on the evaluation dataset to generate adversarial samples and record which pixels are over-driven. The brighter the pixel is, the more often this happened. On a particular transform function (max-percentile), some channels show a large number of sensitive pixels but some channels only show one or two. The majority of the filters remain silent as the detector actually mostly works on particular layers and even particular channels. However, those channels and layers can be hard to identify unless the defense has been deployed and tested against real adversarial samples.

To further demonstrate that behavioural defense can work with different transform functions, Table 3 on the previous page shows the defensive capabilities of LeNet5 (MNIST) trained with three different transform functions, while Figure 5 on the preceding page shows the selected three transform functions.

We conducted three experiments.

1. In the first, we apply the maximum percentile function f_1 on all layers.
2. The second applies f_2 only on the first layer.
3. The third experiment is to apply f_3 on all layers.

The functions, f_1, f_2, f_3, are respectively max threshold with 1st percentile of activation maximum; first layer only restricted with an aggressive range of $([0, 1])$; and all layers restricted to $([0, 1], [2, 3], [4, 5])$. All of the networks were trained to have an accuracy of around 99% with a false positive rate of around 1%. As can be seen in the table, the performance of DeepFool against f_1 is six times that of f_3 and ten times that of f_2.

Yet despite bad performance with f_1, it has the best detector performance against BIM. It is only marginally better than f_2, but is 1.5x better than f_3. Poor performance against BIM does not stop f_3 from scoring best against FGSM. All the functions present relatively good detection rates against weak attacks, but show different levels of capability against stronger ones. 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.

As noted above, the heterogeneity of the potential defensive transform functions offers a potential solution to the white-box attack. A combination of diversity and heterogeneity makes defence unpredictable, in ways that the defender can tune in response to both the expectation and the experience of attacks. Using a key to select from among different transform functions could also bring enough non-determinism to grey-box attacks, when the transform functions are known, but the parameters are not.

Detection can also be randomised. In our evaluation we used the canonical binary detector – the sample is either adversarial or not. One can make use a continuous function instead, measure how confident we are(e.g. how far away it is from the threshold in case of nth percentile function) that a particular sample is adversarial, and respond in an appropriate but non-deterministic way.

We have designed our taboo trap with a view to its being deployed on low-cost hardware where computation may be severely limited, for example by battery life, in such cases the implementation needs a certain amount of care. One way to save energy is use the detector per-layer and abort computation when the detector fires. However, this might introduce a timing side-channel where the implementation detail makes this relevant.

Finally, nothing stops the defender from using classifiers with taboo traps in conjunction with other strategies. 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 [16].

6. Conclusion

In this paper we presented the Taboo Trap, a radically new way to detect adversarial samples that is both simple and cheap. Instead of learning a network's behaviour, we restrict it during training so that it avoid certain taboo values, whether taboo outputs or taboo activations of intermediate layers, and use taboo violation to detect adversarial inputs. We explored the trade-offs between the detectability and aggressiveness of the resulting restrictions, and found that our models show consistently dependable behaviour on a set of common computer-vision benchmarks. We explained how to train the instrumented networks and showed that the resulting classifiers are still able to provide comparable performance to the baseline models regardless of how complex the models are.

The run-time compute overhead 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 can perform much of 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-Forschungsstiftung im 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] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

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

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

[8] 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.

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

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

[11] 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.

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

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

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

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

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

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] A. S. C. Ross. Linguistic class-indicators in present-day english. In *Neuphilosophische Mitteilungen*.

[24] 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.

[25] S. Tian, G. Yang, and Y. Cai. Detecting adversarial examples through image transformation, 2018.

[26] 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.

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

[28] 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.