Adversarial Examples Detection Using Random Pixel-Space Perturbation

Griveau Jordan[1]
griveau.jordan@gmail.com
Jiang Bin[1]
jiangbin@hnu.edu.cn
Melnyk Pavlo[2]
pavlo.melnyk@liu.se

[1] Hunan University
Changsha, China
[2] Linköping University
Linköping, Sweden

Abstract

Similar to other machine learning algorithms, neural networks have been shown to be vulnerable to adversarial examples, *i.e.* inputs containing specifically crafted perturbations whose only objective is to fool a network into misclassification. On the other hand, neural networks are known to be much more robust to *random* perturbations in the input. This motivated us to propose an easy-to-deploy adversarial example detection method based on measuring prediction inconsistencies before and after applying random noise to an input image. We evaluate our method on subsets of three popular benchmarks (Dogs vs Cats, CIFAR-10, and ImageNet) and show that it achieves high adversarial example detection performance for various attacks on higher resolution images. The main advantages of our approach are its simplicity, low computational cost, and the fact that it does not require any prior knowledge of the attack used, which makes it easy to integrate our method into other defence frameworks.

1 Introduction

Deep neural networks (DNNs) have achieved success on numerous tasks, from image and speech recognition [1, 18] to self-driving cars [2] and beating the world champion at the game of Go [19]. However, despite achieving state-of-the-art performance in various domains, DNNs have recently been shown to be vulnerable to adversarial examples [23], *i.e.* inputs containing a carefully crafted perturbation that causes an image classification model to make wrong predictions. Researchers demonstrated this phenomenon to be observable not only in computer vision, but also in speech recognition, by showing that a perturbed audio waveform could make a speech-to-text model drastically change the transcription [5].

A surprising aspect of these specifically perturbed samples is that the amount of perturbation required to make a model misclassify the input can be unnoticeable by a human observer. On the other hand, NNs have been shown to be more robust against random perturbations such as Gaussian noise [9]. This motivated us to consider prediction inconsistencies before and after applying random noise to the input as an approach for detecting adversarial examples.

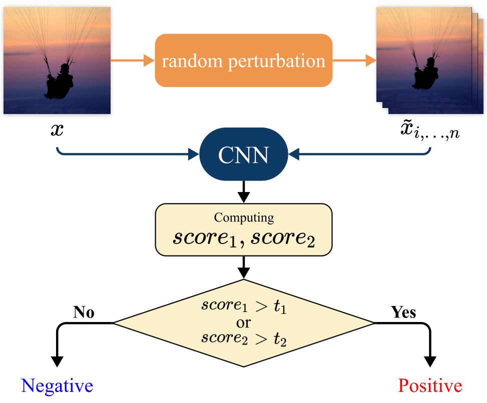


Figure 1: We transform the input image by adding random noise of varying intensity and feed both the original input and the transformed versions into a pre-trained CNN. Two scores are then computed to evaluate the difference between the outputs. The input image is identified as positive (*i.e.* adversarial) if at least one of the two scores exceeds a threshold.

The contributions of our work are as follows:

- We study the effects of applying additive random noise to normal and adversarial examples (Section 4), and observe disparity that, to the best of our knowledge, has not been discussed in prior work.
- We introduce a novel method for detecting adversarial examples (Section 5) that is based on applying random noise with varying intensity to the input and computing two scores. Our method does not require prior knowledge of the attack used, has a low computational cost compared to state-of-the-art techniques [8, 13], and requires no optimisation of the model parameters, and thus, potentially can be easily integrated into other defence frameworks.
- Unlike other approaches, our method can successfully detect both small- and large-perturbation adversarial examples (Section 5).

2 Background

In this section, we provide a brief introduction to NNs and methods to generate adversarial examples.

2.1 Neural networks

A neural network can be written as a function $F(x) = y$, where x represents the input and y represents the output. The final layer of a neural network performing classification is often the softmax activation function. Therefore, $F(x)$ outputs a probability distribution, where $F(x)_i$ represents the probability that input x belongs to class i. We write the final output of the layer as $F(x) = \text{softmax}(Z(x))$, where $Z(x)$ is the network output vector containing *logits*, *i.e.* raw predictions. Finally, the classifier function that returns the most likely class label can be written as $C(x) = \text{argmax}_i(F(x)_i)$.

In the context of our work, we assume to be given a pre-trained NN classifier and a dataset containing normal, *i.e. unmodified*, images as well as the corresponding labels.

2.2 Adversarial examples

With the ever-growing research on the vulnerability of NNs, there are now many different methods to generate adversarial examples. The common objective of such methods is to, from a normal image x, create a perturbation δ and add it to the original image so that the new sample $x' = x + \delta$ is misclassified by a model. We refer to a sample that fulfills this objective as an *untargeted* adversarial example.

On the contrary, a *targeted* adversarial example x' is designed to be classified by a model as a specified target class t. Targeted adversarial examples are typically more challenging to produce and may require a more significant perturbation than untargeted ones. We can thus formulate the optimisation problem to craft adversarial examples as $\min_x D(x+\delta)$, such that $C(x') \neq y$ for an untargeted attack or $C(x') = t$ for a targeted attack, where D represents a distance metric, usually, p-norm defined as $\|D\|_p = (\sum_{i=1}^{n} |d|^p)^{\frac{1}{p}}$.

In the remainder of this section, we briefly discussed some existing attacking methods that we use in our experiments.

Basic Iterative Method (BIM). Kurakin *et al.* [12] proposed an extension of the Fast Gradient Sign Method (FGSM) [8]. Rather than generating the adversarial sample in one step, BIM applies adversarial noise multiple times with a step size α. The benefit of iteratively crafting the adversarial sample x' is that intermediate results can be clipped at each step to ensure that the generated samples are within an ε distance to the original input x:

$$x'_{n+1} = clip_{x,\varepsilon}\left\{x'_n + \alpha.sign\left(\nabla_x J\left(x'_n, y\right)\right)\right\}, \tag{1}$$

where $x'_0 = x$.

Carlini & Wagner (CW). Carlini & Wagner [4] proposed a powerful method to generate adversarial examples that can defeat the defensive distillation approach published by Papernot *et al.* [14]. In their paper, the authors construct an attack for: L_0, L_2 and L_∞ distance metrics.

In our experiments, we use the attack that seeks low distortion in the L_2 distance metric. The final optimisation problem for the CW L_2 attack can be defined as

$$\min \|x' - x\|_2^2 + c \cdot \ell(x'), \tag{2}$$

where c is a constant chosen via binary search that determine the success probability of the attack. The loss function ℓ is the best among seven evaluated by the authors, written as

$$\ell(x') = \max\left(\max\left\{Z(x')_i : i \neq t\right\} - Z(x')_t, -k\right), \tag{3}$$

where $-k$ is a parameter to specify how confident we want the adversarial example to be classified as t by the model.

The CW method is a state-of-the-art attack and effectively finds adversarial examples with a small perturbation size. However, this method is computationally expensive to run.

Decoupled Direction and Norm (DDN). Jérôme *et al.* [17] proposed an improvement over the CW method. The DDN method can obtain comparable results in terms of perturbation size but with considerably fewer iterations. At each iteration k, refine noise β_k by considering a larger norm $\varepsilon_{k+1} = (1+\gamma)\varepsilon_k$ or smaller norm $\varepsilon_{k+1} = (1-\gamma)\varepsilon_k$ depending on if $x+\beta_k$ is adversarial or not. Finally, the method returns the clipped adversarial sample that has the lowest L_2-norm.

2.3 Detection evaluation metrics

To attest to the effectiveness of the detection performances of our method, we adopt the recall and precision rates defined as $recall = \frac{tp}{tp+fn}$ and $precision = \frac{tp}{tp+fp}$, where tp (*true positive*) is the number of correctly detected adversarial examples, fn (*false negative*) is the number of adversarial examples that are not detected, and fp (*false positive*) is the number of normal images that are incorrectly identified as adversarial examples. Finally, we use the F_β score defined as $F_\beta = (1+\beta^2)\frac{recall \cdot precision}{recall+(\beta^2 precision)}$, where $\beta = 2$ to emphasize on the recall rate.

3 Related work

Since Szegedy *et al.* [23] discovered the existence of adversarial examples, much research has been conducted from the perspectives of both the attacker, *i.e.* the party trying to exploit vulnerabilities in a model and the defending party trying to mitigate these attacks. The authors also hypothesized that adversarial examples exist due to the high non-linearity nature of NNs. However, this hypothesis was later rebutted by Goodfellow *et al.* [8] when they argued that adversarial examples exist as a result of NNs being too linear rather than the contrary.

The research first focused on defending against adversarial examples by improving the robustness of neural networks via, among others, adversarial training [8, 13], where adversarial examples are included in a dataset alongside natural images in order to train a CNN on both natural images and adversarial samples. The computational cost required to generate adversarial examples makes this defence approach computationally expensive.

Defensive distillation [14] is another technique to improve the robustness of NN and involves the use of two networks. A first network F is conventionally trained on inputs X and labels Y and outputs a probability vector predictions $F(X)$. The second network F_d is then trained on the same inputs X, but the labels are replaced by the output of the first network $F(X)$ as a way of restraining the model from over-fitting on the data. Defensive distillation is a very effective defence but was defeated by the more recent CW attack [4].

Due to the difficulty of training robust NN against adversarial examples, recent research has focused on detecting them instead. However, a recent survey by Carlini & Wagner [3] examined ten detection defences that they all bypassed using their attack method and concluded that adversarial examples are significantly more complex to detect than previously recognized. Furthermore, among the ten detection methods surveyed, Carlini & Wagner concluded that only *bayesian neural network uncertainty*, introduced by Feinman *et al.* [7], was effective and made generating adversarial examples nearly five times more difficult on CIFAR-10 [11]. Feinman *et al.* [7] use dropout [21] to induce randomness during inference and predict an input multiple times in order to measure the prediction uncertainty. They show that the prediction uncertainty is typically higher on adversarial examples compared to normal and noisy images.

4 Experiments and results

4.1 Overview

Adversarial examples can be seen as a worst-case noise that fools the model into misclassification when applied to an image. On the other hand, neural networks are known to be relatively robust to random corruptions such as Gaussian noise [9]. Our intuition is that applying random noise to an adversarial example could hide parts of the adversarial perturbation, thus diminishing its effect.

In order to apply a random transformation to an image x, we generate a Gaussian noise $\tilde{y} = \mathcal{N}(0,1)$ that we add to the image in order to create a noisy version $\tilde{x}$:

$$\tilde{x} = x + \tilde{y}\kappa, \tag{4}$$

where κ is used to scale the noise intensity.

To further motivate and formulate our method discussed in Section 5, we first perform a series of experiments detailed in the remainder of this section to verify that our intuition holds correct.

4.2 Experimental setup

Below we describe the datasets and models used, as well as the settings of the attack methods.

4.2.1 Datasets and target models

For our experiments, we collect data from the following three datasets: **(1)** CIFAR-10 [11], consisting of 60000 32×32 RGB images divided into ten different classes; **(2)** ImageNet [18], containing over 1300000 high-resolution RGB images resized to 224×224 (we use the ImageNette subset [10] which contains 10 out of the 1000 ImageNet classes); **(3)** Dogs vs. Cats [6], the Asirra (Animal Species Image Recognition for Restricting Access) dataset comprised of 25000 224×224 RGB images divided into two classes.

We use the Pytorch [15] implementation of the pre-trained VGG-11 [20] model for ImageNette and Dogs vs Cats. For CIFAR-10 we use a GoogLeNet architecture [22].

4.2.2 Attacks

We implement the attacking methods introduced in Section 2.2 using the FoolBox library [16]. For targeted attacks defined in Section 2.2, the target t class is selected randomly such that $\{t : t \in C \setminus \{y\}\}$, where C is a set of the available classes in the dataset and y is the ground-truth label of the sample.

We generate adversarial examples using accurately classified images from the test sets of each dataset.

4.3 Robustness

In this section, to verify our intuition regarding the robustness of adversarial perturbations mentioned in Section 4.1, we perform two simple experiments where we compare the effect of applying additive random perturbations to the input on the output of the model, for normal and adversarial inputs.

4.3.1 Classification consistency

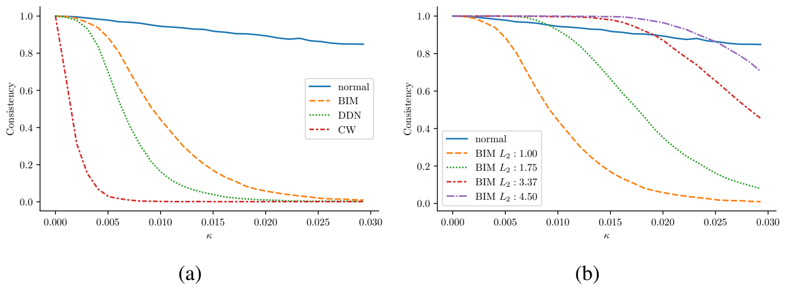


Figure 2: Consistency comparison between normal images and adversarial examples using different methods and perturbation budgets. As κ (see (4)) increases, the consistency decreases, more so for adversarial examples, unless we also increase the adversarial perturbation budget as seen in (b).

First, we conduct an experiment where we compare the classification output of a model given a normal image x and a transformed version $\tilde{x}$ (Eq. (4)). When the classifications $C(x)$ and $C(\tilde{x})$ are identical, we consider them to be *consistent*. Here, our objective is to compare the *consistent* between normal images and adversarial examples. To do so, we randomly select 2048 ImageNet images from the test set to record the classification consistency on each sample and at different κ (Eq. (4)) values. Finally, we follow the same procedure for adversarial examples and generate 2048 samples with the BIM, DDN, and CW methods.

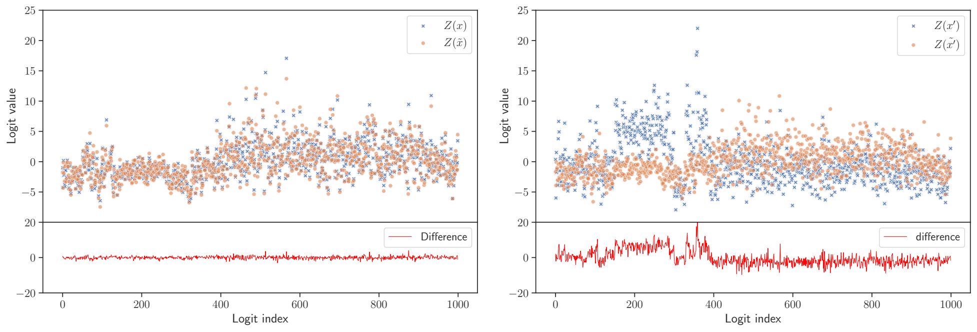
Figure 2a shows the average classification consistency of different input type and at different κ. For normal images, we observe the average consistency to decrease slowly as κ increases. This moderate decrease shows that, as expected, the model is not greatly affected by the random perturbation added to the image. On the contrary, we observe that the classification consistency of adversarial examples decreases rapidly when κ increases. These results seem to confirm our initial intuition that introducing random perturbations to an adversarial example could impede its effectiveness.

On the other hand, we also observe the classification consistency being different between attacking methods, *e.g.* to reach a consistency of ≈ 0.0, adversarial examples generated with BIM need a larger κ than samples generated with DDN and CW. This difference makes sense; adversarial examples generated with the BIM approach have a larger average L_2 adversarial perturbation of ≈ 1.0 compared with DDN at $L_2 \approx 0.85$ and $L_2 \approx 0.70$.

Figure 2b shows the same experiment but using adversarial examples generated with BIM at different perturbation budgets. These results undeniably show that the more substantial the adversarial perturbation is (*i.e.* higher perturbation budget), the more *robust* it is to random perturbations. Therefore, a larger κ may be needed to counter the effectiveness of the attack.

4.3.2 Prediction differences

We perform another simple experiment to investigate further the difference between the model output for an untransformed and a randomly perturbed image. Instead of recording the classification output of the model as done in Section 4.3.1, which is only a portion of



(a) Model output for a normal image (b) Model output for an adversarial example (BIM)

Figure 3: Comparison between the logits of an ImageNet image before, $Z(x)$, and after, $Z(\tilde{x})$, adding noise to the input. When the input is normal (left), the difference between predictions is small, but becomes larger when the input is adversarial (right).

the output, we decide to observe the output logits, which shows the entire, *raw* output.

We select a single ImageNet image x and generate a noisy version $\tilde{x}$ with $\kappa = 2\times10^{-2}$. Figure 3a shows the output $Z(x)$ and $Z(\tilde{x})$. Each logits are plotted (1000 logits for the 1000 ImageNet classes), as well as the difference between $Z(x)_i$ and $Z(\tilde{x})_i$. We observe both outputs to be very similar value-wise. On the contrary, when the original input is adversarial (BIM, ≈ 1.0) as in Figure 3b, the difference between outputs is striking.

Observing and comparing the model outputs showed us that adding a random perturbation to an image can help us detect the legitimate or adversarial nature of the input. Following these observations, we present in the following section a novel method to detect adversarial examples.

5 Method

The overview of the proposed method is introduced in Figure 1. We use a scoring strategy described in Section 5.1 that essentially evaluates the difference between predictions induced by the addition of random noise to the input images.

In the remainder of this section, we present the two scores used and the strategy to detect adversarial examples. The detection results are shown in Section 5.3.

5.1 Scores

As observed in Section 4.3.2, the model predictions differ to a more significant degree when adversarial examples are perturbed by additive random noise compared with normal images.

To capture these differences between the prediction before and after adding random noise, we use two scores defined as follows:

$$score_1(x,\tilde{x}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i-\mu)^2}, \tag{5}$$

where $d = (Z(x) - Z(\tilde{x}))$ is the element-wise difference between two sets of logits and $\mu = \frac{1}{N}\sum_{i=1}^{N} d_i$ is the mean.

The second score similarly compares two sets of predictions by computing the L_1-norm as follows:

$$score_2(x,\tilde{x}) = \|F(x) - F(\tilde{x})\|_1. \tag{6}$$

Notably, $score_1$ uses the raw output of the model, *i.e.* logits, whereas $score_2$ is computed using the output of the softmax layer. We found each score to perform differently for various attacking methods, and, therefore, we used both for better generalisation over a wide variety of attacks.

5.2 Selection of κ

Because of the relation between adversarial and random perturbation discussed in Section 4.3.1, to be able to identify adversarial samples with diverse adversarial budgets, we generate a linearly spaced vector containing ten κ from 1×10^{-2} to 1×10^{-1} and accordingly generate ten noisy images per given input. For each input pair x and $\{\tilde{x}\}_{i=1}^{10}$, we compute both $score_1$ (Eq. 5) and $score_2$ (Eq. 6). The objective is to detect if a sample has an abnormally large value with either of the scores.

Because the thresholds for each score are determined solely using normal training images, our method does not require prior knowledge of the attack to perform well.

5.3 Detection results

Following the same procedure as in Section 5.2, we evaluate our method with adversarial examples generated using different approaches and various perturbation budgets. Table 1 shows the results of this evaluation. We observe a high overall recall rate of 98.5% on ImageNet and 100% on Dogs vs Cats using the two scores combined. The efficacy of each score depends on the type of the attack used. For example, $score_2$ does not perform as well on untargeted attacks but always performs better on CW adversarial examples, compared to $score_1$.

From Table 1, we note that combining the scores shows to be a good strategy: for the cost of a slight decrease in precision, the recall rate is improved as well as the overall F_β score. With BIM, increasing the perturbation budget from an average L_2 perturbation of 1.00 to 3.00, does not affect the detection precision performances (Table 1, ImageNet/Dogs vs Cats, No. 1-3.).

It is common that detection precision is higher when facing adversarial examples with smaller perturbation sizes, such as samples generated with the CW attack method. For instance, in Weilin *et al.* [27], the authors measure the L_1 distance between the prediction vectors of the original image and its squeezed version, using bit depth reduction as well as local and non-local spatial smoothing. They show high detection rates on adversarial examples generated with the CW method, whereas the detection rate falls to 55.56% on samples crafted with BIM. With the same attack settings (BIM$_\infty$, average L_2 of 1.40), our approach demonstrates a 100.0% detection rate using the combined scores (Table 1, ImageNet, No. 5). Therefore, a strength of our method is that it can detect both low-perturbation and large-perturbation adversarial examples. This adaptability comes from the fact that we transform an input with varying random perturbation intensities, which allows us to detect samples generated with CW (at low κ values) and detect BIM or similar methods (at higher κ values). As for the samples generated with the CW method, increasing the number of iterations

Table 1: Detection results for each dataset. Results are shown for both scores individually ($score_1$ and $score_2$) and combined (*combined*). For each attack, we specify the metric with which it was created (∞ or 2), whether the attack is targeted (t), and the corresponding average L_2 distance.

	No.	Attack	L_2	TP			Precision			Recall			F_β		
				$score_1$	$score_2$	*combined*	$score_1$	$score_2$	*combined*	$score_1$	$score_2$	*combined*	$score_1$	$score_2$	*combined*
ImageNet	1		1.00	98	**100**	**100**	0.951	**0.952**	0.917	0.980	**1.000**	**1.000**	0.974	**0.990**	0.982
	2	BIM_2^t	2.00	99	**100**	**100**	**0.952**	**0.952**	0.917	0.990	**1.000**	**1.000**	0.982	**0.990**	0.982
	3		3.00	95	**100**	**100**	0.950	**0.952**	0.917	0.950	**1.000**	**1.000**	0.950	**0.990**	0.982
	4	BIM_2	1.00	96	63	**98**	**0.950**	0.926	0.916	0.960	0.630	**0.956**	0.958	0.673	**0.966**
	5	BIM_∞	1.40	99	70	**100**	**0.952**	0.937	0.917	0.990	0.700	**1.000**	**0.982**	0.737	**0.982**
	6	DDN_2^t	0.85	98	**100**	**100**	0.951	**0.952**	0.917	0.980	**1.000**	**1.000**	0.974	**0.990**	0.982
	7	CW_2^t	0.46	81	**94**	**94**	0.942	**0.949**	0.913	0.810	**0.940**	**0.940**	0.833	**0.942**	0.934
	8	CW_2^t*	0.40	88	**98**	**98**	**0.957**	0.951	0.925	0.880	**0.980**	**0.980**	0.894	**0.974**	0.968
Total / Average				754	725	**790**	**0.951**	0.946	0.918	0.943	0.906	**0.985**	0.944	0.911	**0.973**
Dogs vs Cats	1		1.00	**100**	72	**100**	**0.962**	0.935	0.926	**1.000**	0.720	**1.000**	**0.992**	0.755	0.984
	2	BIM_2	2.00	**100**	46	**100**	**0.962**	0.902	0.926	**1.000**	0.460	**1.000**	**0.992**	0.510	0.984
	3		3.00	**100**	41	**100**	**0.962**	0.891	0.926	**1.000**	0.410	**1.000**	**0.992**	0.460	0.984
	4	DDN_2	0.71	**100**	88	**100**	**0.962**	0.946	0.926	**1.000**	0.880	**1.000**	**0.992**	0.892	0.984
	5	CW_2	0.53	91	99	**100**	**0.958**	0.952	0.926	0.910	0.990	**1.000**	0.919	**0.982**	0.984
Total / Average				491	346	**500**	**0.961**	0.925	0.926	0.982	0.692	**1.000**	0.977	0.720	**0.984**
CIFAR-10	1	BIM_2^t	0.75	**67**	46	**67**	**0.882**	0.754	0.798	0.857	0.460	**0.670**	**0.704**	0.499	0.692
	2	BIM_2	0.75	**78**	34	**78**	**0.897**	0.694	0.821	0.857	0.340	**0.780**	**0.801**	0.379	0.788
	3	BIM_∞	1.00	**85**	40	**85**	**0.904**	0.727	0.833	0.857	0.400	**0.850**	**0.860**	0.440	0.847
	4	DDN_2	0.60	78	47	**79**	**0.897**	0.758	0.823	0.780	0.470	**0.790**	**0.801**	0.509	0.796
	5	CW_2	0.10	28	**89**	**89**	0.757	**0.856**	0.840	0.280	**0.890**	**0.890**	0.320	**0.883**	0.879
	6	CW_2^t	0.22	37	**82**	**82**	0.804	**0.845**	0.828	0.370	**0.820**	**0.820**	0.415	**0.825**	0.822
Total / Average				373	338	**480**	**0.857**	0.772	0.824	0.622	0.563	**0.800**	0.650	0.589	**0.804**

from 100 (Table 1, ImageNet, No. 7) to 10000 (Table 1, ImageNet, No. 8) does not reduce the detection performances.

We select a wider range of κ for the evaluation on CIFAR-10 images. The range goes from 1×10^{-4} to 1×10^{-1} and contains 30 κ. We explain the reasons for this choice in Section 6.

6 Discussion

As shown in Table 1, the detection performances on CIFAR-10 are inferior compared to ImageNet or Dogs vs Cats. We suppose that this is caused by the fact that the perturbation needed to produce adversarial examples on lower-resolution images (*e.g*. CIFAR-10) is proportionally larger than on images with higher resolution. As discussed in Section 4.3, more significant adversarial perturbations (measured by L_2-norm) are more robust to the random noise we add to the images. Consequently, to detect adversarial examples that contains a more significant relative adversarial perturbation, we need to use a larger κ to improve the detection. However, while using a larger κ proved to be helpful to detect larger-perturbation adversarial examples, a κ too high can deteriorate normal images, leading the model not to recognize them, thus, increasing the number of false positives.

In all our experiments, we considered adversaries who have full access to the model parameters but do not adapt to bypass our detection method. To do so, an adversary would need to produce examples such that $score_1$ and $score_2$ threshold levels are not exceeded, which is a considerably more challenging problem than simply crafting adversarial examples.

We hypothesise that the detection performances of our method could be improved by performing by adding Gaussian noise as a form of data augmentation during the training phase.

References

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *arXiv:1512.02595 [cs]*, December 2015. URL http://arxiv.org/abs/1512.02595. arXiv: 1512.02595.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *arXiv:1604.07316 [cs]*, April 2016. URL http://arxiv.org/abs/1604.07316. arXiv: 1604.07316.

[3] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *arXiv:1705.07263 [cs]*, November 2017. URL http://arxiv.org/abs/1705.07263. arXiv: 1705.07263.

[4] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv:1608.04644 [cs]*, March 2017. URL http://arxiv.org/abs/1608.04644. arXiv: 1608.04644.

[5] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. *arXiv:1801.01944 [cs]*, March 2018. URL http://arxiv.org/abs/1801.01944. arXiv: 1801.01944.

[6] Jeremy Elson, John (JD) Douceur, Jon Howell, and Jared Saul. Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization. October 2007. URL https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-

[7] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting Adversarial Samples from Artifacts. *arXiv:1703.00410 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1703.00410. arXiv: 1703.00410.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, March 2015. URL http://arxiv.org/abs/1412.6572. arXiv: 1412.6572.

[9] Dan Hendrycks and Thomas G. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations. *arXiv:1807.01697 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1807.01697. arXiv: 1807.01697.

[10] Jeremy Howard. ImageNette. URL https://github.com/fastai/imagenette/.

[11] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. April 2009.

[12] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*, February 2017. URL http://arxiv.org/abs/1607.02533. arXiv: 1607.02533.

[13] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv:1511.07528 [cs, stat]*, November 2015. URL http://arxiv.org/abs/1511.07528. arXiv: 1511.07528.

[14] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, May 2016. doi: 10.1109/SP.2016.41. ISSN: 2375-1207.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. page 12.

[16] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX. *Journal of Open Source Software*, 5(53):2607, September 2020. ISSN 2475-9066. doi: 10.21105/joss.02607. URL https://joss.theoj.org/papers/10.21105/joss.02607.

[17] Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. *arXiv:1811.09600 [cs]*, April 2019. URL http://arxiv.org/abs/1811.09600. arXiv: 1811.09600.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575 [cs]*, January 2015. URL http://arxiv.org/abs/1409.0575. arXiv: 1409.0575.

[19] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961. Publisher: Nature Publishing Group.

[20] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015. URL http://arxiv.org/abs/1409.1556. arXiv: 1409.1556.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014. URL http://arxiv.org/abs/1409.4842. arXiv: 1409.4842.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, February 2014. URL http://arxiv.org/abs/1312.6199. arXiv: 1312.6199.