Noise Based Approach for the Detection of Adversarial Examples

Matias Alejandro Kloster[1,3], Ariel Hernan Curiale[1,2,3] and German Mato[1,2,3,4]

[1] Instituto Balseiro, Universidad Nacional de Cuyo, Argentina
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.
[3] Departamento de Física Médica, Centro Atómico Bariloche e Instituto Balseiro, Av. Bustillo 9500, R8402AGP S. C. de Bariloche, Río Negro, Argentina.
[4] Comisión Nacional de Energía Atómica (CNEA), Argentina.

Abstract. We propose a new method for detecting adversarial examples based on a stochastic approach. An example is presented to the network several times and classified as adversarial if the fraction of times the output label is different from the label generated by the deterministic network is above some threshold value. We analyze the performance of the method for three attack methods (DeepFool, Fast Gradient Sign Method and norm 2 Carlini Wagner) and two datasets (MNIST and CIFAR-10). We find that our approach works best for stronger attacks such as DeepFool and CW2, and could be used as part of a scheme where several methods are applied simultaneously in order to estimate if a given input is adversarial or not.

1 Introduction

In recent years Deep Neural Networks (DNN) have been successfully applied to a wide range of problems, such as image processing, speech recognition and genomics [10]. Szegedy et al. [18] have made an intriguing discovery, in the particular case of image recognition, neural networks can misclassify an image that is slightly different from one extracted from the data distribution. In this sense, **adversarial examples** or **adversarial images** can be defined as those with this characteristic, and on the other hand, those extracted from the original dataset are called **natural images**. As it is mentioned in [6], a wide variety of networks with different architectures trained on different subsets of the training data misclassify the same adversarial example. This suggests that adversarial examples expose fundamental problems in the most used training algorithms.

Let us note that natural and adversarial images can be very similar and even almost indistinguishable to humans. As an example, in Fig. 1 we show the image of a dog, the perturbation found by a specific attack and finally the corresponding adversarial image which is categorized as a cat.

The analysis of this problem has been divided in several questions:

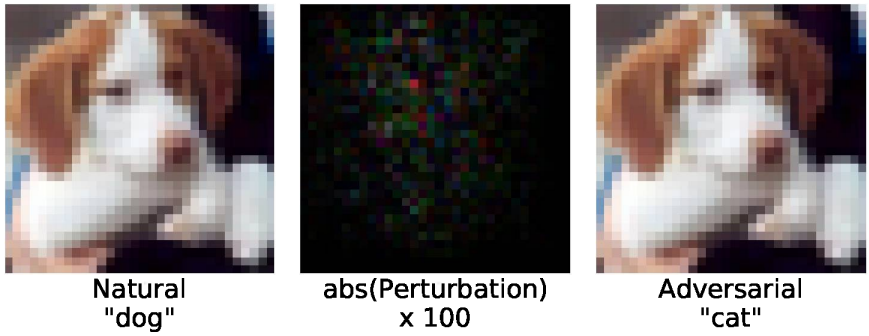


Fig. 1: Example of an adversarial image. Left panel: natural image classified as a dog. Central panel: perturbation found using the DeepFool attack (in absolute value and multiplied by a factor 100). Right panel: adversarial image classified by the network as a cat.

How to generate an adversarial example: The problem of finding an adversarial example was previously treated as an optimization problem. For example, Szegedy et al. [18] solve the problem using a box constrained L-BFGS algorithm. Another method that uses an optimization approach is Carlini-Wagner (CW) [2]. Here several possible regularizers are introduced in order to find the minimal perturbation with respect to different metrics.

Other approaches are based on a more geometrical interpretation. Neural networks for image categorization can be interpreted as a very high dimensional function, where the input is a vector representing the image and the output is a set of n numbers, being n the amount of possible categories. Each one of these numbers represents the probability that the image belongs to that class. The most probable category is the one that obtains the highest number. With this interpretation it is easy to see that there is a set of decision borders, that determine the points where two output units have the same value. Therefore, given a natural image, one can try modify it to reach and cross some of the decision borders of the neural network. The direction of the modification can be approximated using the gradient of the loss function, giving rise to the Fast Gradient Sign Method (FGSM) [6] or iteratively evaluating the gradient of the outputs themselves as in the DeepFool method [15].

Another approach is based in locating the part of the image that should be perturbed in order to change the classification given by the network. This can be achieved by evaluating the *saliency map* [16]. Given the saliency map, the method picks the most important pixel and modifies it to increase the likelihood of a given class . This is repeated until either more than a set threshold of pixels are modified which makes the attack detectable, or it succeeds in changing the classification.

How to defend against adversarial attacks: A defense is a process by which a neural network learns to correctly classify an example that was previously constructed to become an adversarial example. This topic has been extensively studied (see for instance [20]). Here we mention only a few of the several approaches that have been tried in this direction. For instance *defensive distillation* [17] is used to smooth the model learned by a DNN architecture during training by helping the model to generalize better to samples outside of its training dataset and making it more robust to adversarial examples. Similarly, in [4] stochastic pruning is proposed to improve robustness. Transformations in the input have been also proposed as a defense mechanism [8].

How to detect adversarial attacks: This problem is to detect whether a given input has been manipulated to generate a spurious output. Carlini and Wagner, show in [1] the difficulty of succeeding in this problem. In their paper, they analyze ten different strategies to identify which one obtains better results in terms of detection. The strategy that obtained the best results is *Dropout randomization* [5], in which Feinman et al. propose to measure the uncertainty of a network against a certain input. To do this, randomness is added to the network architecture, hoping that natural images will be categorized in the same way despite this random factor. On the other hand, it is expected that the prediction of adversarial images will not always be the same in all iterations. Other approaches are *feature squeezing* [19], where the search space available to an adversary is reduced by coalescing samples that correspond to many different feature vectors in the original space into a single sample. Comparing the prediction of the DNN model corresponding to the original input with the prediction of squeezed inputs, feature squeezing detects adversarial examples with good accuracy. Meng and Chen [13] independently proposed a similar adversary detection method as Xu et al. [19] that also uses the prediction vectors of the original and the filtered images. Metzen et al. [14] propose to attach a convolutional neural network-based detector as a branch off a middle layer of the original DNN model. Other authors base the detection on the study of the statistical properties of the inputs. For instance [7] adds a new adversarial class in the last layer of the DNN model . The revised model is trained with both legitimate and adversarial examples, in [12] the statistics is analyzed at the level of the convolutional filters. Dathathri et al. [3] propose a different approach called *neural fingerprinting*. It consists in detecting adversarial examples by verifying whether model behavior is consistent with a set of secret fingerprints, inspired by the use of biometric and cryptographic signatures.

Even when this problem has received a lot of attention it can be considered as unsolved. As it is mentioned in [1] these detection methods are not really robust meaning that the attacker can fool them by performing another attack on a modified network. This network has an additional output, that informs if the example is adversarial or not. Generating adversarial examples on this network will, with some probability, defeat the detection method. This approach is less efficient for systems that use random perturbations, as in the case of *Dropout randomization* [5]. Inspired by this result we propose a new approach,

that consists in analyzing the output of the network when the activation functions of the neurons themselves have a random component. The structure of the paper is as follows: in the next section we present the network architectures, the datasets and the adversarial attacks we are using. In section 3 we introduce the proposed detection method and show the results. In the last section we discuss the results and possible implications.

2 Network Architectures and Adversarial Attacks

We follow the notation proposed in [1]. A neural network is denoted by a function $F(x)$, where $x \in \mathbb{R}^n$. For a network that performs classification, the output of F is a softmax function, so the results can be interpreted as probabilities. We consider only feed-forward networks, where the output of layer i can be evaluated in terms of the output of the previous layer according to:

$$F^i(x) = g\left(A^i F^{i-1}(x) + b^i\right) \tag{1}$$

where A^i is a weight matrix and b^i a bias vector. All the network parameters can be grouped in vector θ. The non-linear input-output transfer function is denoted by g. The output of the network before taking the softmax function is denoted by $Z(x)$. The learning process involves the minimization of a cost function $J(\theta, \{x, y\})$ with respect to the different components of the vector θ, where y are the outputs associated with each input x. The prediction of the network for each input x is given by

$$C(x) = \arg\max_j(F(x)_j). \tag{2}$$

The datasets used for the training of the models, and later for the generation of adversarial images will be MNIST [11] and CIFAR-10 [9].

In Fig. 2 we show the architectures to learn the important characteristics of each of the datasets for their subsequent classification. In all of the cases the global architecture is the same: a set of 2D convolutional layers, followed by a set of dense layers. At the last dense layer a softmax operation is performed in order to interpret the outputs of the network as probabilities. The results for the training process are shown in Table 1. Let us note that the two problems involve different levels of complexity. The characteristics of MNIST dataset can be learnt almost perfectly while CIFAR-10 displays a greater level of difficulty.

Database	Val Acc
MNIST	0.992
CIFAR-10	0.762

Table 1: Validation accuracy for the two datasets with the networks of Fig. 2.

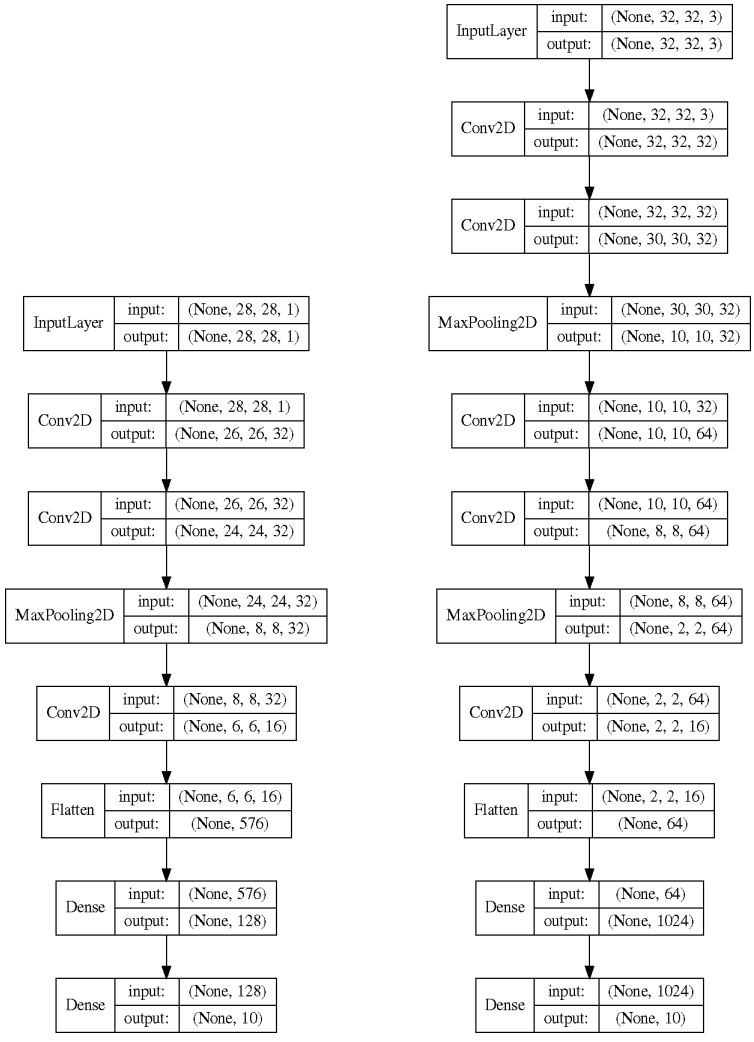


Fig. 2: Network architectures used for the MNIST dataset (left) and CIFAR-10 dataset (right). The indices represent the dimensions. For instance 32,32,2 means a structure with a spatial dimension of 32x32 and 3 channels.

The adversarial examples are then generated using three different techniques: DeepFool [15], FGSM [6] and CW2 (norm 2 Carlini-Wagner) [2]. Let us observe

that we are working in the situation most favorable to the attacker, given that it has perfect knowledge of the network and weights used, in other words they all are *white box* attacks. Moreover, in all of the cases we analyze the attack methods that generate by themselves the label of the adversarial example, instead of having to use a label provided by some external source. In other words we are using *untargeted* attacks.

Next, we briefly explain each one of the attack methods.

2.1 DeepFool Method

For an image x, whose correct classification is k_0 we first evaluate

$$k^* = \arg\min_{k \neq k_0} \frac{Z_{k_0}(x) - Z_k(x)}{\|\nabla_x (Z_{k_0}(x) - Z_k(x))\|_2} \tag{3}$$

In this way, we identify the target index that will require the minimal perturbation to cross the decision boundary. Now the input image is modified according to the following rule:

$$x \to x - (1+\eta) r, \tag{4}$$

where

$$r = \frac{Z_{k_0}(x) - Z_{k^*}(x)}{\|\nabla_x (Z_{k_0}(x) - Z_{k^*}(x))\|_2^2} \nabla_x (Z_{k_0}(x) - Z_{k^*}(x)), \tag{5}$$

and η is some small number included to make more likely to go beyond the decision boundary. If this is not achieved, eqs. 3 and 4 are applied iteratively until the output class has changed.

This attack intends to achieve the perturbation that minimizes the l_2- norm, but generalizations to another norms are also possible (see [15]).

2.2 Fast Gradient Sign Method

As it was proposed in [6] each input image is modified according to the following rule:

$$x \to x - \epsilon \operatorname{sign}\left(\nabla_x J(\theta, \{x\})\right) \tag{6}$$

Parameter ϵ can be tuned to have an acceptable success rate for the generation of adversarial examples with a small perturbation. Let us observe that this method affects all the pixels in the image with the same absolute value.

2.3 Carlini-Wagner Attack (l_2- norm)

This attack uses an optimization approach. For a given image x with initial classification k_0, we search the image x' that minimizes the following function

$$\|x' - x\|_2^2 + cL(x') \tag{7}$$

where the loss function L is defined by

$$L(x') = \max(\max\{Z(x')_i : i \neq k_0\} - Z(x')_{k_0}, -\kappa) \tag{8}$$

Parameter κ controls the confidence of the adversarial examples and parameter c regulates the relative weight of the constraint on the norm. Larger values of c will generate adversarial examples at a higher success rate but the size of the perturbation will be larger.

2.4 Performance of the Attack Methods

The performance of the attack methods can be quantified by the success rate P_{gen}, which is the fraction of times that they successfully converge on a desired label and by the mean size of the adversarial perturbations, which can be defined by ρ_{adv} or L_2. Due to the fact that by reading other papers one or the other parameter can be seen. In this work we present both.

$$\rho_{adv} = \frac{1}{N_{test}} \sum_{x \in test} \frac{\|r(x)\|_2}{\|x\|_2}, \tag{9}$$

$$L_2 = \frac{1}{N_{test}} \sum_{x \in test} \|r(x)\|_2, \tag{10}$$

where x are the images in the test set and the vector $r(x)$ represents the perturbation generated for each image.

In the Attack results section of Table 2 we show P_{gen} and L_2 for each dataset and attack. Note that the Table 2 also shows the parameters used to perform the attacks. These parameters were chosen in such a way that L_2 is similar when using different attacks over the same dataset.

Dataset	Attack	Attack params	Attack results			Optimal detection results					
			ρ_{adv}	L_2	P_{gen}	σ_{conv}	σ_{dense}	ROC-AUC set	ROC-AUC val	P_{det} set	P_{det} val
MNIST	DeepFool	$\eta = 0.01$	0.18	1.52	0.99	0.60	1.00	1.00	1.00	0.99	0.99
	FGSM	$\epsilon = 0.062$	0.19	1.65	0.08	1.00	3.00	0.80	0.86	0.73	0.80
	CW2	$cte = 0.011$	0.18	1.47	0.73	1.00	1.80	0.99	0.99	0.98	0.98
CIFAR-10	DeepFool	$\eta = 0.45$	0.01	0.26	0.99	0.06	0.02	0.93	0.94	0.89	0.89
	FGSM	$\epsilon = 0.005$	0.01	0.26	0.39	0.14	0.14	0.62	0.63	0.59	0.59
	CW2	$cte = 0.0032$	0.01	0.29	0.83	0.08	0.04	0.91	0.93	0.86	0.87

Table 2: Results obtained when performing attack and detection in the two datasets. The parameters of the attacks were chosen in such a way that the size of the adversarial perturbations are similar between attacks for the same dataset.

We can see that the three methods generate different results. DeepFool has the highest probability of success when trying to generate an adversarial example, so it could be considered the most successful method. In the other extreme FGSM

has the lowest probability of success. This probability could be increased by taking a larger value of parameter ϵ, but that would lead to a larger perturbation, that can be detected by visual inspection.

Fot the MNIST dataset, the adversarial image of a FGSM attack can be easily detected in the background of the image. The CW2 attack has an intermediate behavior between DeepFool and FGSM. On the other hand, for the CIFAR10 dataset, the three adversarial images are indistinguishable from the natural one.

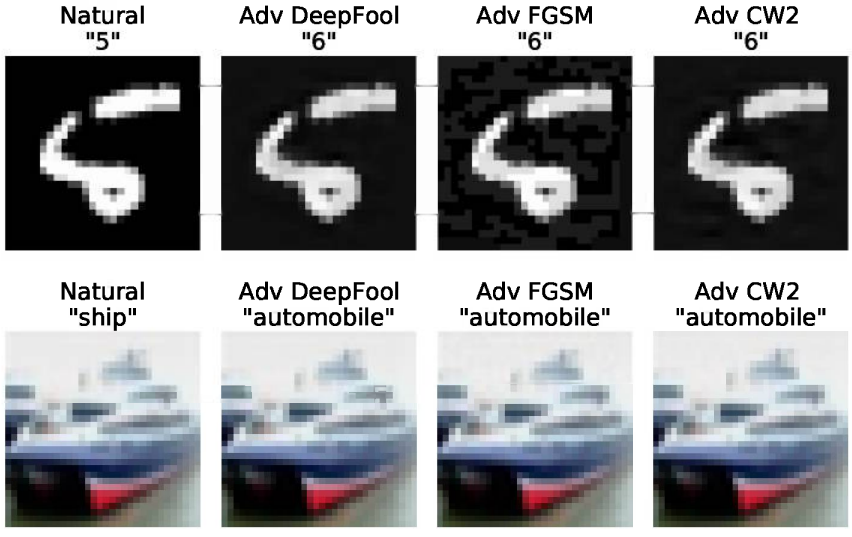


Fig. 3: Sample of results from the MNIST dataset in the top row and the CIFAR-10 in the bottom row. Left panel: natural images. Other panels, from left to right: result of the attacks for DeepFool, FGSM and CW2.

3 Noise Based Detection Method

As mentioned before, the main goal of this work is to detect whether a given image is natural or adversarial. In order to do this, stochastic neural activations are added to the model, which means adding Gaussian perturbations of mean zero and a variance to be determined, in the argument of the activation functions of some set of neurons. In the deterministic case the activation of neuron i is given by

$$F_i = g\left(\sum_j A_{ij} F_j + b_i\right) \tag{11}$$

where g is the activation function, F_j is the output of neuron j, A_{ij} is the connection strength between neuron j and neuron i and b_i is the bias. Note that in the case of convolutional networks, neurons only see a local surrounding, so matrix A would have non-zero values in local connections and zeros in the rest.

Our proposal is to replace Eq. 11 by

$$F_i = g\left(\sum_j A_{ij}F_j + b_i + \xi_i\right) \tag{12}$$

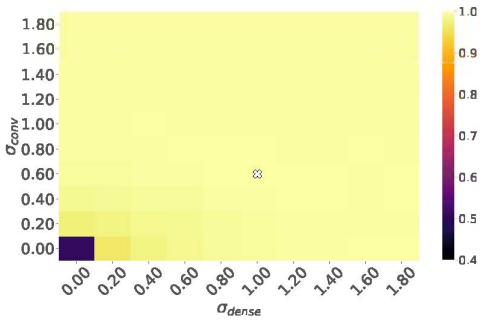
where ξ_i are independent Gaussian random variables with 0 mean and standard deviation σ_i. If σ_i is not zero the output of the network will be stochastic. We can present the same input several times obtaining different results. Suppose we present a given input N_r times. Let us denote with f_r the fraction of times the prediction of the network is the same as in the deterministic network. Our methods predicts that the input is an adversarial example if f_r is lower than some threshold value f_T. Otherwise it will be considered as a natural example. The quality for the prediction can be evaluated via the ROC-AUC, that gives the fraction of true positives as a function of the false positives for different values of the threshold. A value of ROC-AUC of 1 means that there is a threshold value that discriminates perfectly between natural and adversarial examples. If ROC-AUC is 0.5, the method gives a random answer.

The expected behavior of this detection method would be that when the size of the perturbation approaches 0, the ROC-AUC should tend to 0.5, where adversarial examples, by definition, fool the deterministic network. By contrast, with very large perturbations, natural and adversarial images are so disturbed that they become indistinguishable from each other and ROC-AUC is also 0.5. In between, there may be some perturbation size with a ROC-AUC greater than 0.5 and in the best case close to 1.

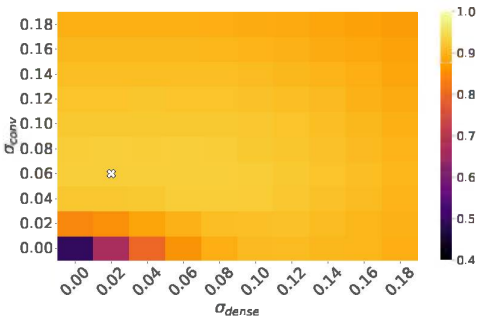
This prescription defines in fact a family of detection methods, based on which of the neurons have a stochastic activation. In this study, we have decided to place this stochastic activation in the last convolutional layer and in the penultimate dense layer. In this way we can analyze whether it is more effective to perturb the network at the feature level or at the level where the final decision is taken. A detailed analysis of which layers should be perturbed in order to optimize detection performance will be done in a future work.

Note that for each dataset the architecture to be used and the position of the stochastic activation has already been defined. Therefore, the only free parameters are the variances of the added Gaussian perturbations, that we denote by σ_{conv} and σ_{dense}. Fig. 4 shows the results for the three attacks and the two datasets. It is important to mention that to obtain all the graphs in Figure 4 the datasets were filtered so that the only images taken into account were the correctly classified by the neural network and with successful adversarial examples generated

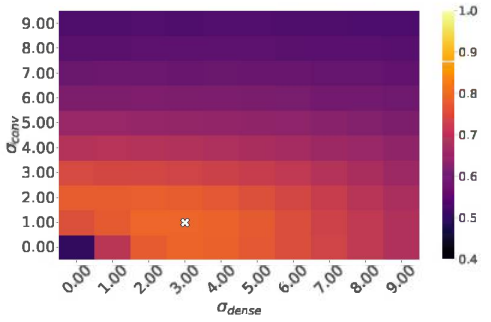
From Fig. 4 we can conclude that the optimal values of the variances of the stochastic neurons depends on the problem. For CIFAR-10 the optimal configuration is achieved for smaller values of σ_{conv} and σ_{dense}.



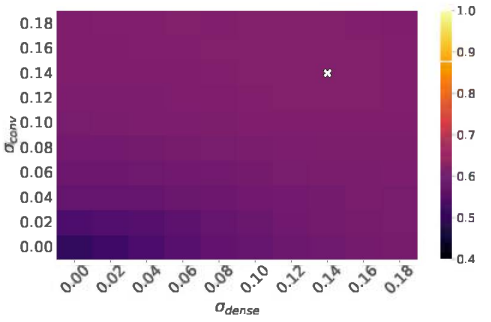
(a) MNIST - DeepFool



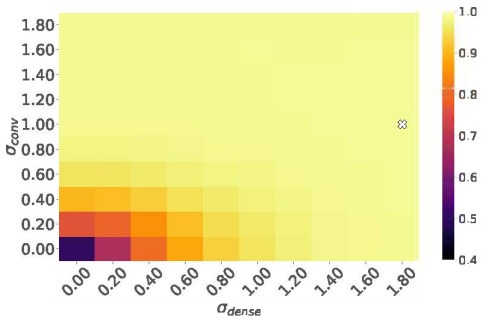
(b) CIFAR-10 - DeepFool



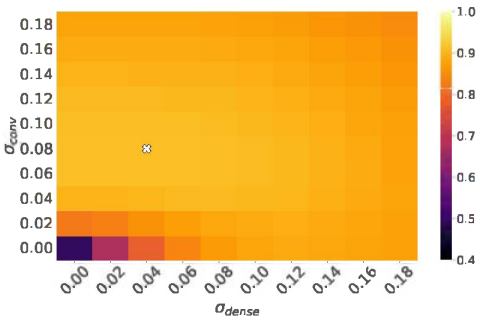
(c) MNIST - FGSM



(d) CIFAR-10 - FGSM



(e) MNIST - CW2



(f) CIFAR-10 - CW2

Fig. 4: Values of ROC-AUC for the different datasets and attack methods as a function of σ_{conv} and σ_{dense}. The optimal ROC-AUC value, i.e. the highest, is shown with a white cross.

From these results we can choose the combination of variances σ_{conv} and σ_{dense} that maximizes the ROC-AUC indicator. Then, using these optimal variances values, the optimal threshold f_T value is calculated. This is the value that minimizes the sum of false positives and false negatives. Or what is equivalent, it maximizes the probability of success in predicting the nature of an image (P_{det}) using this detection technique. Remember that the ROC-AUC and P_{det} obtained until now were determined in the process of setting f_T, so in Table 2 these values are shown as ROC-AUC set and P_{det} set.

To validate the results, we use a new collection of natural and adversarial images that were not used to set the optimal values of ROC-AUC and P_{det}. Using f_T and the optimal values variances σ_{conv} and σ_{dense} already obtained, we

calculate the validation values of ROC-AUC and P_{det} over the new set of images. These results are shown in the Table 2.

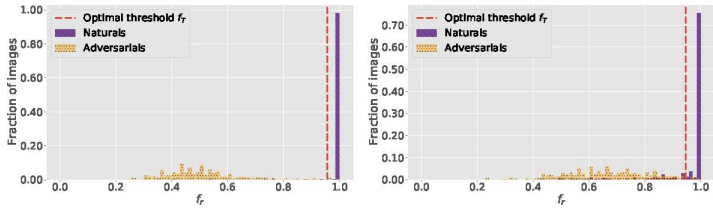
In order to emphasize the difference in behavior between the different cases we show in Fig. 5 the distribution of values of f_r for the different cases for the optimal choice of σ_{conv} and σ_{dense}. Remember that the f_T shown in each of the histograms graphs was obtained with a certain group of images that is not shown. Then, with another group of validation images, the f_T value previously obtained is put to the test. The histograms correspond to the validation images.

When performing the FGSM attack on the CIFAR-10 dataset, the lowest values of ROC-AUC and P_{det} were obtained. These values are congruent with the histograms of the Figure 5d which are difficult to discriminate. In the rest of the cases, there is an f_T with convincing results that vary from 80 to 99% in percentage of success in detecting the nature of the validation images.

In order to make a comparison with other papers, the results obtained from [5] and [19] are presented in the Table 3. The generation of the adversarial examples in this work was done with the smallest possible perturbations to keep the adversarial examples as similar as possible to the natural examples. As a result, in the Table 3 it can be seen that the magnitudes of the perturbations chosen in this work are smaller than the rest. If we compare two detection methods for a particular attack, it would be preferable if the size of the perturbations are similar.

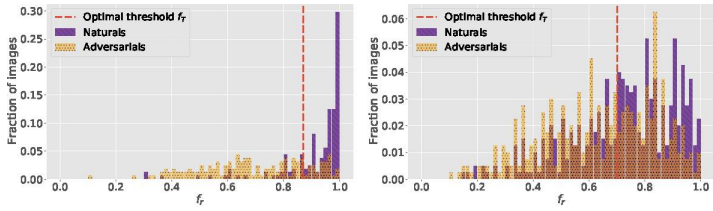
Data extracted from	**Dataset**	**Attack**	L_2	P_{det}(val)	ROC-AUC(val)
Noise Based Approach	MNIST	DeepFool	1.52	0.99	1.00
		FGSM	1.65	0.80	0.86
		CW2	1.47	0.98	0.99
	CIFAR-10	DeepFool	0.26	0.89	0.94
		FGSM	0.26	0.59	0.63
		CW2	1.29	0.87	0.93
Feature squeezing [19]	MNIST	DeepFool	-	-	-
		FGSM	5.91	1.00	-
		CW2	2.87	1.00	-
	CIFAR-10	DeepFool	0.23	0.77	-
		FGSM	0.86	0.20	-
		CW2	0.29	1.00	-
From artifacts [5]	MNIST	DeepFool	-	-	-
		FGSM	6.22	-	0.91
		CW2	4.71	-	0.98
	CIFAR-10	DeepFool	–	-	-
		FGSM	2.74	-	0.72
		CW2	2.70	-	0.92

Table 3: Comparison of results between this paper, [5] and [19]



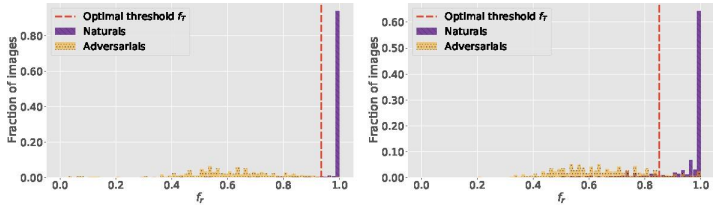
(a) MNIST - DeepFool
$\sigma_{conv} = 0.60$; $\sigma_{dense} = 1.00$

(b) CIFAR-10 - DeepFool
$\sigma_{conv} = 0.06$; $\sigma_{dense} = 0.02$



(c) MNIST - FGSM
$\sigma_{conv} = 1.00$; $\sigma_{dense} = 3.00$

(d) CIFAR-10 - FGSM
$\sigma_{conv} = 0.14$; $\sigma_{dense} = 0.14$



(e) MNIST - CW2
$\sigma_{conv} = 1.00$; $\sigma_{dense} = 1.80$

(f) CIFAR-10 - CW2
$\sigma_{conv} = 0.08$; $\sigma_{dense} = 0.04$

Fig. 5: Fraction of images that have a given value of f_r. Violet: natural images. Yellow: adversarial images. Remember that the histograms used for setting the value of f_T are not shown. Instead, the histograms that are shown are used to validate f_T.

4 Conclusion

We introduced here a new method for detecting adversarial examples. It is based on DNN's where some neurons can incorporate stochastic activation functions. We analyzed three different attack methods (DeepFool, FGSM and CW2) applied to two datasets (MNIST and CIFAR-10). We analyzed two possibilities for the location of the stochastic activation: the last convolutional layer and the penultimate dense layer. We found that depending on the characteristics of the problem, the optimal values of the variances of the stochastic variables are different. A more detailed optimization of the parameters of the stochastic activations (location

in the model, probability distribution, etc.) could be performed. This would not only improve detection performance, but would also give us an insight to develop better detections methods.

As it is suggested in [1] detection methods should be tested for several datasets and attack methods. Several defenses that are useful against attacks such as FGSM or JSMA [16] fail againts stronger attacks such as DeepFool or CW. Here we show a method that, in contrast, works best for these stronger attacks and could be used as part of a scheme where several methods are applied simultaneously in order to estimate if a given input is adversarial or not.

Acknowledgments

This work was partially supported by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) and Comisión Nacional de Energa Atómica (CNEA). German Mato acknowledges CONICET for the grant PIP 112 201301 00256.

References

1. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 3–14 (2017)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
3. Dathathri, S., Zheng, S., Murray, R.M., Yue, Y.: Detecting adversarial examples via neural fingerprinting. arXiv preprint arXiv:1803.03870 (2018)
4. Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A.: Stochastic activation pruning for robust adversarial defense. arXiv preprint arXiv:1803.01442 (2018)
5. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410 (2017)
6. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
7. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 (2017)
8. Guo, C., Rana, M., Cisse, M., Van Der Maaten, L.: Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117 (2017)
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436–444 (2015)
11. LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits, 1998. URL http://yann. lecun. com/exdb/mnist 10, 34 (1998)
12. Li, X., Li, F.: Adversarial examples detection in deep networks with convolutional filter statistics. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5764–5772 (2017)
13. Meng, D., Chen, H.: Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 135–147 (2017)

14. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267 (2017)
15. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
16. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy (EuroS&P). pp. 372–387. IEEE (2016)
17. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 582–597. IEEE (2016)
18. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
19. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155 (2017)
20. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems 30(9), 2805–2824 (2019)