






Adversarial Examples Detection in Features Distance Spaces

Fabio Carrara¹(✉) , Rudy Becarelli², Roberto Caldelli^{2,3} ,
Fabrizio Falchi¹ , and Giuseppe Amato¹

¹ ISTI-CNR, Via Giuseppe Moruzzi, 1, 56127 Pisa, Italy

{fabio.carrara,fabrizio.falchi,giuseppe.amato}@isti.cnr.it

² MICC, University of Florence, Viale Morgagni 65, 50134 Firenze, Italy

{roberto.caldelli,rudy.becarelli}@unifi.it

³ CNIT, Viale G.P. Usberti, 181/A, 43124 Parma, Italy

Abstract. Maliciously manipulated inputs for attacking machine learning methods – in particular deep neural networks – are emerging as a relevant issue for the security of recent artificial intelligence technologies, especially in computer vision. In this paper, we focus on attacks targeting image classifiers implemented with deep neural networks, and we propose a method for detecting adversarial images which focuses on the trajectory of internal representations (i.e. hidden layers neurons activation, also known as *deep features*) from the very first, up to the last. We argue that the representations of adversarial inputs follow a different evolution with respect to genuine inputs, and we define a distance-based embedding of features to efficiently encode this information. We train an LSTM network that analyzes the sequence of deep features embedded in a distance space to detect adversarial examples. The results of our preliminary experiments are encouraging: our detection scheme is able to detect adversarial inputs targeted to the ResNet-50 classifier pre-trained on the ILSVRC'12 dataset and generated by a variety of crafting algorithms.

Keywords: Adversarial examples · Distance spaces · Deep features · Machine learning security

1 Introduction

In recent years, Deep Learning, and in general Machine Learning, undergone a considerable development, and an increasing number of fields largely benefit from its adoption. In particular, deep neural networks play a central role in many fields spanning from computer vision – with applications such as image [17] and audio-visual understanding [29], multi-media sentiment analysis [38], automatic video captioning [11], relational reasoning [35], cross-modal information retrieval [7] – to cybersecurity – enabling malware detection [32], automatic content filtering [39], and forensic applications [3], just to name a few. However, it

is known to the research community that machine learning and specifically deep neural networks, are vulnerable to *adversarial examples*.

Adversarial examples are maliciously manipulated inputs – often indiscernible from authentic inputs by humans – specifically crafted to make the model misbehave. In the context of image classification, an adversarial input is often obtained adding a small, usually imperceptible, perturbation to a natural image that leads the model to misclassify that image. The ease of generating adversarial examples for machine learning based classifiers poses a potential threat to systems relying on neural-network classifiers in sensitive applications, such as filtering of violent and pornographic imagery, and in the worst case even in safety-critical ones (e.g. road sign recognition for self-driving cars).

Most of the scientific work on the subject focus on two main antithetic aspects of adversarial examples, which are their generation and the defense against them. About the latter, many works propose techniques to change the attacked model in order to be more robust to such attacks (unfortunately without fully solving the problem).

Recently, an alternative defensive approach has been explored, which is the detection of adversarial examples. In this setting, we relax the defensive problem: we dedicate a separate detector to check whether an input is malicious, and we relieve the model from correctly classifying adversarial examples.

In this work, we propose a detection scheme for adversarial examples in deep convolutional neural network classifiers, and we conduct a preliminary investigation of its performance. The main idea on which our approach is based is to observe the trajectory of internal representations of the network during the forward pass. We hypothesized that intermediate representations of adversarial inputs follow a different evolution with respect to natural inputs. Specifically, we focus on the relative positions of internal activations with respect to specific points that represent the dense parts of the feature space. Constructing a detector based on such information allows us to protect our model from malicious attacks by effectively filtering them. Our preliminary experiments give an optimistic insight into the effectiveness of the proposed detection scheme. The code to reproduce our results is publicly available¹.

2 Related Work

Adversarial Examples. One of the first works exploring adversarial examples for image classifiers implemented with convolutional neural network is the one of Szegedy et al. [37]. The authors used a quasi-newtonian optimization method, namely L-BFGS, to find an image \mathbf{x}_{adv} close to an original one \mathbf{x} in terms of L2 distance, yet differently classified. They also have shown that the obtained adversarial images were also affecting different models trained on the same training set (*cross-model generalization*) and also models trained with other yet similar training sets (*cross-training set generalization*).

¹ <https://github.com/fabio carrara/features-adversarial-det>.

Crafting Algorithms. Goodfellow et al. [15] proposed the Fast Gradient Sign Method (FGSM) to efficiently find adversarial perturbations following the gradient of the loss function with respect to the image, which can be efficiently computed by back-propagation. Many other methods derived from FGSM have been proposed to efficiently craft adversarial images. Kurakin et al. [22] proposed a basic iterative version of FGSM in which multiple finer steps are performed to better explore the adversarial space. Dong et al. [12] proposed a version of iterative FGSM equipped with momentum which won the NIPS Adversarial Attacks and Defences Challenge [23] as best attack; this resulted in adversarial images with an improved transferability among models. Other attack strategies aim to find smaller perturbations using a higher computational cost. In [30], a Jacobian-based saliency map is computed and used to greedily identify the best pixel to modify in order to steer the classification to the desired output. In [28], the classifier is locally linearized and a step toward the simplified classification boundary is taken, repeating the process until a true adversarial is reached. Carlini and Wagner [5] relied on a modified formulation of the adversarial optimization problem initially formalized by Szegedy et al. [37]. They move the misclassification constraint in the objective function by adding a specifically designed loss term, and they change the variable of the optimization to ensure the obtained image is valid without enforcing the box constraint; Adam is then employed to optimize and find better adversarial perturbations. Sabour et al. [34] explored the manipulation of a particular internal representation of the network by means of adversarial inputs, showing that it is possible to move the representation closer to the one of a provided guide image.

Defensive Methodologies. In the recent literature, a considerable amount of work has been dedicated to increasing the robustness of the attacked models to adversarial inputs. Fast crafting algorithms, such as FGSM, enabled a defensive strategy called *adversarial training*, in which adversarial examples are generated on the fly and added to the training set while training [15, 20]. While models that undergo adversarial training still suffer from the presence of adversarial inputs, the perturbation needed to reach them is usually higher, resulting in a higher detectability. In [31], the authors proposed to use a training procedure called *distillation* to obtain a more robust version of a vulnerable model by smoothing the gradient directions around training points an attacker would exploit. Other defenses aim at removing the adversarial perturbation via image processing techniques, such as color depth reduction or median filtering of the image [40]. Despite performing well on specific attacks with very stringent threat models, they usually fail on white-box attacks [19].

Adversarial inputs detection is also extensively studied. Despite many detection strategies have been proposed based on detector sub-networks [14, 27], statistical tests [13, 16], or perturbation removal [25], yet results are far from satisfactory for all threat models [6], and adversarial detection still pose a challenge.

In our work, we focus on feature-based detection scheme. Being Deep Learning a family of representation-learning methods capable of building a hierarchy of features of increasing level of abstraction [24], the relevance of the internal

representation learned by deep models has been proved by many works starting from [2, 33, 36]. Typically used for transfer learning in scenarios, they have been proved to be useful for adversarial detection in [1, 8, 9, 27]. The works most related to our are [8] and [1]; the former looks at the neighborhood of the input in the space of CNN internal activations to discriminate adversarial examples, while the latter proposes to measure the average local spatial entropy on back-propagated activations, called feature responses, to trace and identify effects of adversarial perturbations. Our work is still based on internal CNN activations, but focus on their evolution throughout the forward pass; in particular, we search for discrepancies between trajectories traced by natural inputs and adversarial ones.

3 Background

3.1 Attack Model

Biggio et al. [4] categorized the kind of attack based on the knowledge of the attacker. A *zero-knowledge* adversary is the one producing adversarial examples for the classifier while being unaware of the defensive strategy deployed; this scenario is usually over-optimistic since it considers a very limited adversary, but is the basic benchmark to test new detection algorithms. Instead, a *perfect-knowledge* adversary is aware of both the classifier and the detector and can access the parameters of both models; this is the worst-case scenario in which the adversary has full control and on which many of the detection schemes are bypassed [6]. A half-way scenario is the one with a *limited-knowledge* adversary, that is aware of the particular defensive strategy being deployed, but does not have access to its parameters or training data.

In this preliminary work, we focus on the *zero-knowledge* scenario and plan to test our approach in the other scenarios in future work.

3.2 Adversarial Crafting Algorithms

In this section, we review the algorithms used to craft adversarial examples used in our experiments. We focus on untargeted attacks, i.e. attacks that cause a misclassification without caring of the precise class we are promoting instead of the real one; thus, whenever possible, we employ the untargeted version of the classification algorithms, otherwise, we resort to the targeted version choosing a random target class. As distance metric to quantify the adversarial perturbation, we choose the L_∞ distance. Thus, we generated adversarial examples such that

$$\|\mathbf{x}_{adv} - \mathbf{x}\|_\infty = \max(\mathbf{x}_{adv} - \mathbf{x}) < \varepsilon,$$

where \mathbf{x} is the natural input, \mathbf{x}_{adv} its adversarial version, and ε the chosen maximum perturbation.

L-BFGS. The first adversarial attack for convolutional neural networks has been formulated as an optimization problem on the adversarial perturbation [37]:

$$\begin{aligned} & \underset{\eta}{\text{minimize}} && \|\eta\|_2 + C \cdot \mathcal{L}_t(\mathbf{x} + \eta) \\ & \text{subject to} && L \leq \mathbf{x} + \eta \leq U \end{aligned} \quad (1)$$

where η is the adversarial perturbation, $\mathcal{L}_t(\mathbf{x} + \eta)$ is the loss relative to the target class t , and $[L, U]$ is the validity range for pixels. The box-constrained L-BFGS algorithm is employed to find a solution. The loss weight C is tuned via grid search in order to obtain a minimally perturbed image $\mathbf{x}_{adv} = \mathbf{x} + \eta$ which is actually labelled as the target class t .

Fast Gradient Sign Method. The Fast Gradient Sign Method (FGSM [15]) algorithm searches for adversarial examples following the direction given by the gradient $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})$ of the loss function $\mathcal{L}(\mathbf{x})$ with respect to the input image \mathbf{x} . In particular, the untargeted version of FGSM sets

$$\mathbf{x}_{adv} = \mathbf{x} + \varepsilon \cdot \text{sign}(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})).$$

Following this direction, the algorithm aims to increase the loss, thus decreasing the confidence of the actual class assigned to \mathbf{x} .

Iterative Methods. The Basic Iterative Method (BIM) was initially proposed in [22]. Starting from the natural image \mathbf{x} , iterative methods apply multiple steps of FGSM with a distortion $\varepsilon_i \leq \varepsilon$. The untargeted attack performs

$$\mathbf{x}_{adv}^0 = \mathbf{x}, \quad \mathbf{x}_{adv}^{i+1} = \text{clip}(\mathbf{x}_{adv}^i + \varepsilon_i \nabla \mathcal{L}(\mathbf{x}_{adv}^i)),$$

where $\text{clip}(\cdot)$ ensures the image obtained at each step is in the valid range. Madry et al. [26] proposed an improved version of BIM – referred to as Projected Gradient Descent (PGD) – which starts from an initial acceptable random perturbation.

Iterative FGSM with Momentum. The Iterative FGSM with Momentum (MI-FGSM [12]) won the first place as the most effective attack in the NIPS 2017 Adversarial Attack and Defences Challenge [23]. The main idea is to equip the iterative process with the same momentum term used in SGD to accelerate the optimization. The untargeted attack performs

$$\mathbf{g}^{i+1} = \mu \mathbf{g}^i + \frac{\nabla \mathcal{L}(\mathbf{x}_{adv}^i)}{\|\nabla \mathcal{L}(\mathbf{x}_{adv}^i)\|_1}, \quad (2)$$

$$\mathbf{x}_{adv}^{i+1} = \text{clip}(\mathbf{x}_{adv}^i + \varepsilon_i \nabla \mathcal{L}(\mathbf{x}_{adv}^i)), \quad (3)$$

where $\mathbf{x}_{adv}^0 = \mathbf{x}$, $\mathbf{g}^0 = 0$, and μ is the decay factor for the running average.

4 Feature Distance Spaces

In this section, we introduce the intuition on which our detection scheme is based, and we formalize the concept of *feature distance spaces*.

Our hypothesis states that the positions of the internal activations of the network in the feature space differ in their evolution from input to output between adversarial examples and natural inputs. Inspired by works on Euclidean embeddings of spaces for indexing purposes [10, 41], we encode the position of the internal activations of the network for a particular image in the feature space, and we rely on this information to recognize it as adversarial or genuine. Rather than keeping the absolute position of the activations in the space, we claim that their relative position with respect to the usual locations occupied by genuine activations can give us insight about the authenticity of the input. We define different *feature distance spaces* – one per layer – where dimensions in those new spaces represent the relative position of a sample with respect to a given reference point (or pivot) in the feature space. Embedding all the internal representations of an input into these spaces enables us to compactly encode the evolution of the activations through the forward pass of the network and search for differences between trajectories traced by genuine and adversarial inputs. A toy example of this concept is depicted in Fig. 1, where the dashed red lines represent the information we rely on to perform the detection.

Pivoted Embedding. Let I the image space, $f : I \rightarrow \{1, \dots, C\}$ a C -way single-label DNN image classifier comprised by L layers, and $\mathbf{o}^{(l)}$ the output of the l -th layer, $l = 1, \dots, L$. For each layer l , we encode the position in the feature space of its output $\mathbf{o}^{(l)}$ by performing a pivoted embedding, i.e. an embedding in a feature distance space where each dimension represent the distance (or similarity) to a particular pivot point in the feature space. As pivots, we choose C points that are representative of the location each of the C classes occupy in the feature space. Let $\mathbf{P}^{(l)} = \{\mathbf{p}_1^{(l)}, \dots, \mathbf{p}_C^{(l)}\}$ the set of C points chosen as pivots in the activation space of layer l , and $d(x, y)$ a distance function defined over real vectors; the embedded version $\mathbf{e}^{(l)} \in \mathbb{R}^C$ of $\mathbf{o}^{(l)}$ is defined as

$$\mathbf{e}^{(l)} = \left(d\left(\mathbf{o}^{(l)}, \mathbf{p}_1^{(l)}\right), d\left(\mathbf{o}^{(l)}, \mathbf{p}_2^{(l)}\right), \dots, d\left(\mathbf{o}^{(l)}, \mathbf{p}_C^{(l)}\right) \right).$$

Given an input image, we perform a forward pass of the classifier, collect the internal activations $(\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(L)})$, and embed them using the L sets of pivots $\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(L)}$, obtaining a L -sized sequence of C -dimensional vectors $\mathbf{E} = (\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(L)})$. Our detector is then a binary classifier trained to discern between adversarial and natural inputs solely based on \mathbf{E} , and it is employed at test time to check whether the input has been classified reliably by the DNN.

The rationale behind this approach based on class dissimilarity space is to highlight possible different behaviors of adversarial and original images when passing throughout the DNN layers. In fact, it is expected that original images, correctly classified by the network, would follow a path much more similar to that of the pivots representing their output class rather than adversarial ones which

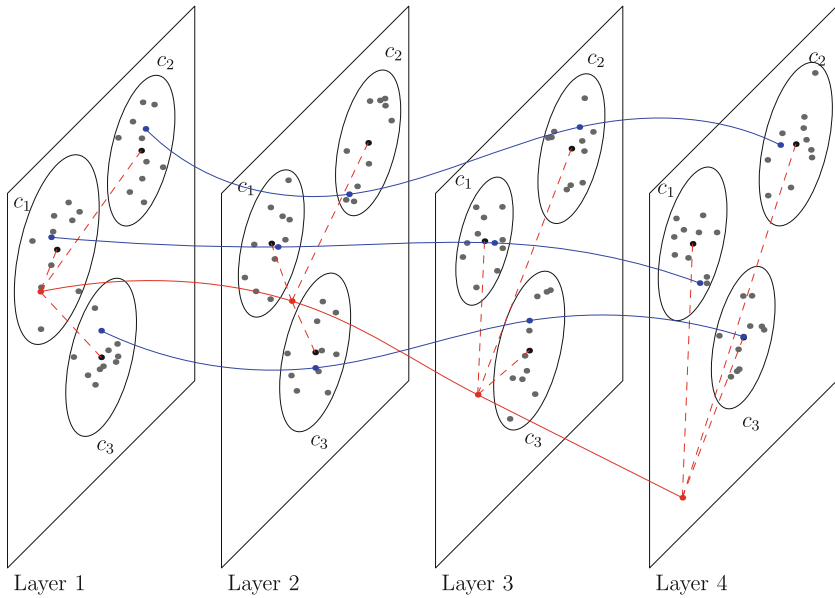


Fig. 1. Example of the evolution of features while traversing the network that illustrates our hypothesis. Each plane represents a feature space defined by the activations of a particular layer of the deep neural network. Circles on the features space represent clusters of features belonging to a specific class. Blue trajectories represent authentic inputs belonging to three different classes, and the red trajectory represent an adversarial input. We rely on the distances in the feature space (red dashed lines) between the input and some reference points representatives of the classes to encode the evolution of the activations. (Color figure online)

artificially fall in that class. Consequently, all the other relative distances with respect to the $(C - 1)$ pivots should evidence some dissimilarities. An overview of the complete detection scheme is depicted in Fig. 2.

Pivot Selection. In our approach, the pivots constitute a sort of “inter-layer reference map” that can be used to make a comparison with the position of a test image at each layer in the feature space. Activations of the images belonging to the training set of the classifier are used to compute some representative points eligible to be employed as pivots. We propose two strategies for selecting the pivot points $\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(L)}$ used for the pivoted embedding.

In the first one, we select as pivot $\mathbf{p}_c^{(l)}$ the *centroid* of the activations of layer l of the images belonging to class c

$$\mathbf{p}_c^{(l)} = \frac{1}{K_c} \sum_{j=1}^{K_c} \mathbf{o}_{c,j}^{(l)}, \tag{4}$$

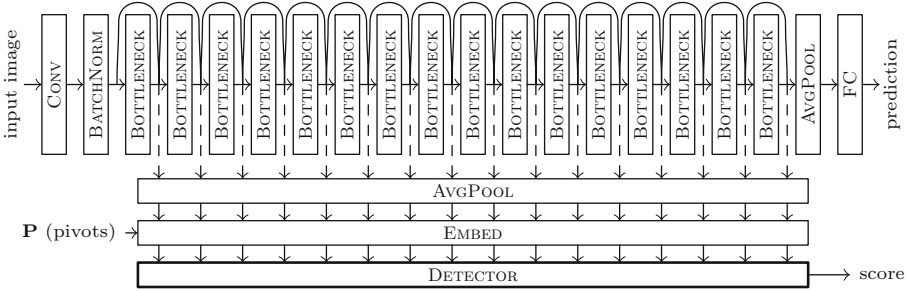


Fig. 2. Scheme of the proposed detection method. The network represented on the top is the ResNet-50. Given an input image and a set of pivots, our detector outputs a score representing the probability of the input image being an adversarial input.

where K_c indicates the cardinality of class c , and $\mathbf{o}_{c,j}^{(l)}$ is the activation of the l -th layer produced by the j -th training sample belonging to class c .

In the second strategy, the pivot $\mathbf{p}_c^{(l)}$ is selected as the *medoid* of the activations of layer l of the images belonging to class c , i.e. the training sample that minimize the sum of the distances between itself and all the others sample of the same class. Formally, assuming $\mathcal{O}_c^{(l)} = \{\mathbf{o}_{c,1}^{(l)}, \dots, \mathbf{o}_{c,K_c}^{(l)}\}$ the set of activations of the l -th layer of the training samples belonging to class c ,

$$\mathbf{p}_c^{(l)} = \operatorname{argmin}_{\mathbf{x} \in \mathcal{O}_c^{(l)}} \sum_{j=1}^{K_c} \|\mathbf{x} - \mathbf{o}_{c,j}^{(l)}\|_2. \quad (5)$$

The pivots are compute off-line once and stored for the embedding operation.

5 Evaluation Setup

In this section, we present the evaluation of the proposed feature distance space embeddings for adversarial detection in DNN classifiers.

We formulate the adversarial example detection task as a binary image classification problem, where given a DNN classifier f and an image \mathbf{x} , we assign the positive label to \mathbf{x} if it is an adversarial example for f . The detector D is implemented as a neural network that takes as input the embedded sequence \mathbf{E} of internal activations of the DNN and outputs the probability p that \mathbf{x} is an adversarial input. We tested both the pivot selection strategies, namely *centroid* and *medoid*; for the choice of the distance function $d(\cdot, \cdot)$ used in the pivoted embeddings, we tested the Euclidean distance function and the cosine similarity function.

5.1 Dataset

Following the research community in adversarial attacks and defenses, we chose to run our experiments matching the configuration proposed in the NIPS

2017 Adversarial Attacks and Defenses Kaggle Competitions [23]. Specifically, we selected the famous ImageNet Large Scale Visual Recognition Challenge (ILSVRC) as the classification task, and we chose the ResNet-50 model pre-trained on ILSVRC training set as the attacked DNN classifier. As images to be perturbed by adversarial crafting algorithms, we selected the DEV image set proposed in the NIPS challenge, which is composed by 1,000 images that are not in the ILSVRC sets but share the same label space. We split the images in a train, validation and test sets respectively counting 700, 100, and 200 images.

For every image, we obtained adversarial examples by applying the crafting algorithms reported in Sect. 3.2. We performed the untargeted version of the attacks and used maximum perturbations $\varepsilon \in \{\frac{20}{255}, \frac{40}{255}, \frac{60}{255}, \frac{80}{255}\}$. For iterative attacks, we set $\varepsilon_i = \frac{20}{255}$ and performed 10 iterations. Depending on the type and the parameters of the attack, the attack success rates vary. The detailed composition of the dataset can be found in Table 1, and an example of adversarial inputs generated is available in Fig. 3.

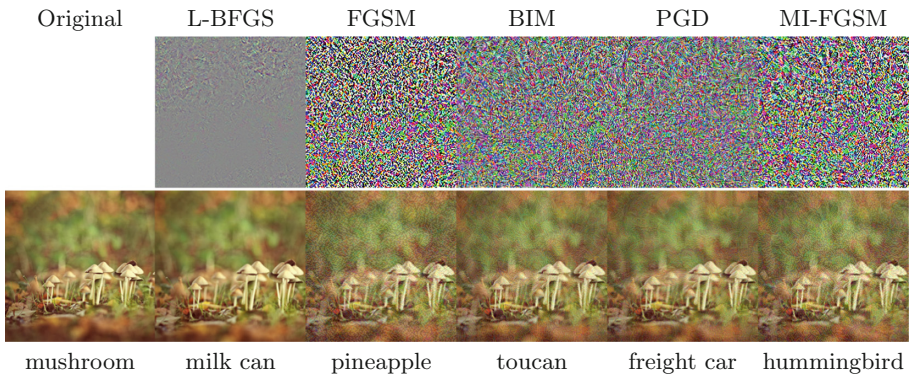


Fig. 3. Examples of adversarial perturbation (on top) and inputs (on bottom) generated by the adopted crafting algorithms. Perturbations are magnified for visualization purposes.

For each image, we extracted 16 intermediate representations computed by the ResNet-50 network; we considered only the output of the 16 Bottleneck modules ignoring internal layers; for more details about the ResNet-50 architecture, we refer the reader to [18]. Internal features coming from convolutional layers have big dimensionality due to large spatial information; we reduced their dimensionality by applying a global average pooling to each extracted feature. We then embedded the feature in each layer in the feature distance space as explained in Sect. 4 using the *cleverhans*² library. Thus, we obtained a sequence of 16 1,000-dimensional features where the i -th feature vector represents the distances between the i -th internal representation and the 1,000 class pivots of that particular layer.

² <https://github.com/tensorflow/cleverhans>.

Table 1. Details of the adversarial generated for the experiments. The maximum perturbations ε and ε_i are expressed in fractions of 255. Statistics for BIM, PGD, and MI-FGSM are the same for every ε used; thus the aggregated number of images for those configurations are reported.

Attack	ε	ε_i	Iterations	TRAIN	VAL	TEST	Success rate
Authentic	–	–	–	700	100	200	–
L-BFGS	–	–	–	698	100	200	99.8 %
FGSM	20	–	–	682	96	166	94.4 %
	40	–	–	666	91	163	92.0 %
	60	–	–	658	93	160	91.1 %
	80	–	–	654	95	165	91.4 %
BIM	20–80	20	10	2,800	400	800	100.0 %
PGD	20–80	20	10	2,800	400	800	100.0 %
MI-FGSM	20–80	20	10	2,800	400	800	100.0 %
Total	–	–	–	12,458	1,775	3,454	–

Applying this procedure to the whole set of images yielded a dataset of sequences we used to train our adversarial detector.

5.2 Detector Architecture and Training Details

To capture the evolution of the activations through layers, the architecture of the detector is composed by an LSTM cell with hidden state size of 100 that efficiently process the embedded sequence \mathbf{E} . The last hidden state of the LSTM is then fed to a fully connected layer with one output followed by a sigmoid activation.

As a baseline, we also report the results obtained with a multi-layer perceptron network (MLP) which takes as input the concatenation of the 16 1,000-dimensional features, i.e. a unique 16,000-dimensional vector. The MLP is composed by two layers; the first with 100 neurons and ReLU activation, and the last one with a single output followed by a sigmoid activation which represents the probability of the input of being an adversarial example.

Both models are trained with the Adam optimizer [21] for 100 epochs with a batch size of 128 and a learning rate of $3 \cdot 10^{-4}$. We employ a weighted random sampler to ensure a balanced distribution of authentic images and adversarial attacks in mini-batches thus avoiding biases towards most present attacks.

5.3 Results

In Fig. 4, we report the ROC curves and AUC values for each configuration of architecture (LSTM or MLP), pivot-selection strategy (*medoid* or *centroids*), and embedding function (Euclidean distance or cosine similarity). The *medoid* pivot-selection strategy yields a detector with a very high performance, as we can

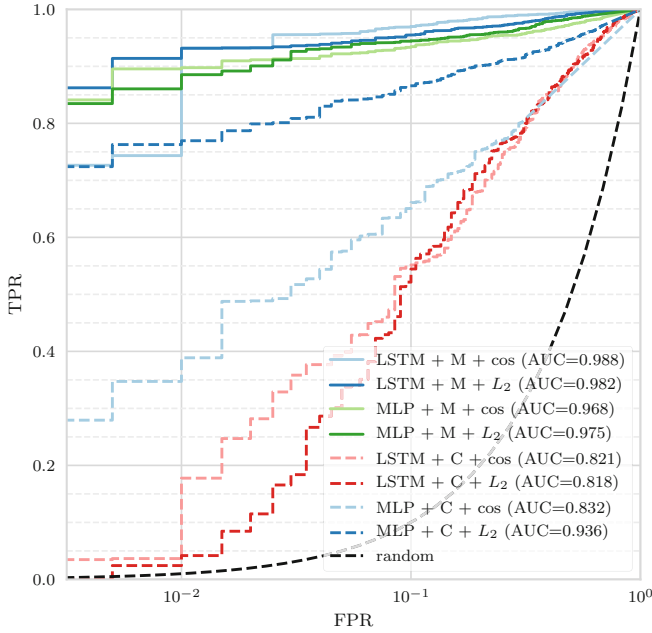


Fig. 4. ROC curves for all the configurations of the detection scheme tested. The label ‘M’ stands for the *medoid* pivot-selection strategy, while ‘C’ for *centroid*.

Table 2. Area Under the ROC Curves (AUC) broken down by attack. The last column reports the unweighted mean of the AUCs.

Method	L-BFGS	FGSM	BIM	PGD	MI-FGSM	Macro-AUC
LSTM + M + cos	.854	.996	.997	.997	.997	.968
LSTM + M + L_2	.743	.996	.998	.998	1.000	.947
MLP + M + cos	.551	.992	.996	.995	.998	.907
MLP + M + L_2	.681	.976	.998	.999	1.000	.931
LSTM + C + cos	.709	.811	.784	.784	.930	.804
LSTM + C + L_2	.482	.854	.819	.816	.872	.769
MLP + C + cos	.388	.694	.881	.878	.962	.761
MLP + C + L_2	.626	.820	.990	.989	1.000	.885

notice from the high AUC values obtained by both architectures; this strategy is also robust to the choice of the embedding function. On the other hand, we obtained mixed results when using the *centroid* strategy.

The superiority of the *medoid* strategy is even clearer looking at the AUC values broken down by attack types and their mean (macro-averaged AUC), reported in Table 2.

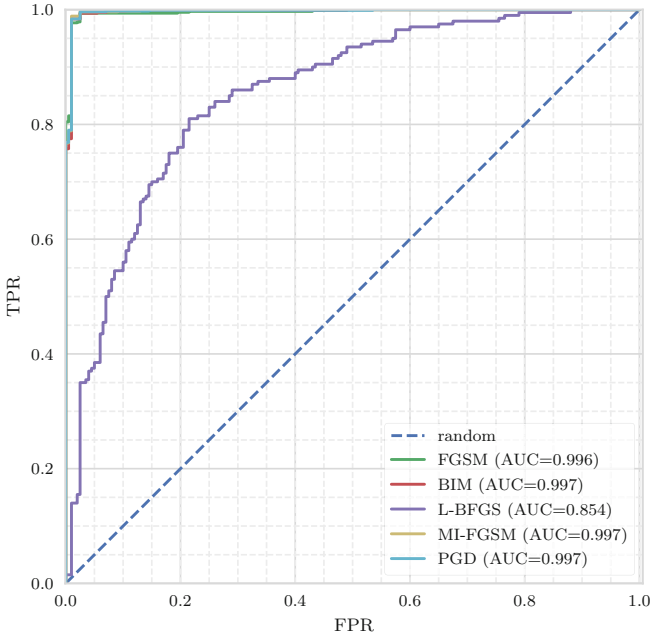


Fig. 5. ROC curves obtained by our best performing model (LSTM + M + cos) for each type of adversarial attack.

As expected, stronger attacks, i.e. L-BFGS, are more difficult to detect on average; however, the increased attack performance of L-BFGS is obtained at a higher computational cost, which is roughly two orders of magnitude higher with respect to the other attacks in our setup. The perturbations produced by L-BFGS are usually smaller than other methods (the mean perturbation has L_∞ norm of $\sim \frac{20}{255}$) and is visually more evasive, see Fig. 3). Still, we are able to reach a satisfactory level of performance on such attacks while correctly detecting FGSM-based attacks with high accuracy. Overall, the LSTM-based detector performs better than the MLP model: the recurrent model has considerably fewer parameters (0.4M vs 1.6M of the MLP) which are shared across the elements of the sequence; thus, it is less prone to overfitting and also less computationally expensive.

Figure 5 shows the ROC curves – one per crafting algorithm – obtained by our best model, i.e. LSTM + *medoid* + cosine similarity. On FGSM-based attacks, this detection scheme is able to correctly detect near all the manipulated input, reaching an equal error rate (EER) accuracy – i.e. the accuracy obtained when the true positive rate is equal to the false positive rate – of 99%. On images generated with L-BFGS, our model reaches an EER accuracy of roughly 80%.

6 Conclusions

The vulnerability of deep neural network to adversarial inputs still poses security issues that need to be addressed in real case scenarios. In this work, we propose a detection scheme for adversarial inputs that rely on the internal activations (called deep features) of the attacked network, in particular on their evolution throughout the network forward pass. We define a feature distance embedding which allowed us to encode the trajectory of deep features in a fixed length sequence, and we train an LSTM-based neural network detector on such sequences to discern adversarial inputs from genuine ones. Preliminary experiments have shown that our model is capable of detecting FGSM-based attacks with almost perfect accuracy, while the detection performance on stronger and computational intensive attacks, such as L-BFGS, reaches around the 80% of EER accuracy. Given the optimistic results obtained in the basic threat model considered, in future work, we plan to test our detection scheme on more stringent threat models – e.g. considering a limited-knowledge or perfect-knowledge adversary – and to incorporate more adversarial crafting algorithms – such as JSMA, DeepFool, and C&W attacks – into the analysis. Moreover, we plan to extend our insight on the trajectories of adversarial examples in feature spaces with an extended quantitative analysis.

Acknowledgments. This work was partially supported by Smart News, Social sensing for breaking news, co-founded by the Tuscany region under the FAR-FAS 2014 program, CUP CIPE D58C15000270008, and Automatic Data and documents Analysis to enhance human-based processes (ADA), CUP CIPE D55F17000290009. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 and Titan Xp GPUs used for this research.

References

1. Amirian, M., Schwenker, F., Stadelmann, T.: Trace and detect adversarial attacks on CNNs using feature response maps. In: Pancioni, L., Schwenker, F., Trentin, E. (eds.) ANNPR 2018. LNCS (LNAI), vol. 11081, pp. 346–358. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99978-4_27
2. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 584–599. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_38
3. Bayar, B., Stamm, M.C.: A deep learning approach to universal image manipulation detection using a new convolutional layer. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2016, pp. 5–10. ACM, New York (2016). <https://doi.org/10.1145/2909827.2930786>
4. Biggio, B., et al.: Evasion attacks against machine learning at test time. In: Bloekel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 387–402. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_25
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. arXiv preprint [arXiv:1608.04644](https://arxiv.org/abs/1608.04644) (2016)

6. 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, AISec 2017, pp. 3–14. ACM, New York (2017). <https://doi.org/10.1145/3128572.3140444>
7. Carrara, F., Esuli, A., Fagni, T., Falchi, F., Fernández, A.M.: Picture it in your mind: generating high level visual representations from textual descriptions. *Inf. Retrieval J.* **21**(2), 208–229 (2017)
8. Carrara, F., Falchi, F., Caldelli, R., Amato, G., Becarelli, R.: Adversarial image detection in deep neural networks. *Multimed. Tools Appl.* **2018**, 1–21 (2018)
9. Carrara, F., Falchi, F., Caldelli, R., Amato, G., Fumarola, R., Becarelli, R.: Detecting adversarial example attacks to deep neural networks. In: Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing, CBMI 2017, pp. 38:1–38:7. ACM, New York (2017). <https://doi.org/10.1145/3095713.3095753>
10. Connor, R., Vadicamo, L., Rabitti, F.: High-dimensional simplexes for supermetric search. In: Beecks, C., Borutta, F., Kröger, P., Seidl, T. (eds.) SISAP 2017. Lecture Notes in Computer Science, vol. 10609, pp. 96–109. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68474-1_7
11. Dong, J., Li, X., Lan, W., Huo, Y., Snoek, C.G.: Early embedding and late reranking for video captioning. In: Proceedings of the 2016 ACM on Multimedia Conference, pp. 1082–1086. ACM (2016)
12. Dong, Y., et al.: Boosting adversarial attacks with momentum. arXiv preprint (2018)
13. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts. arXiv preprint [arXiv:1703.00410](https://arxiv.org/abs/1703.00410) (2017)
14. Gong, Z., Wang, W., Ku, W.S.: Adversarial and clean data are not twins. arXiv preprint [arXiv:1704.04960](https://arxiv.org/abs/1704.04960) (2017)
15. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014). arXiv preprint [arXiv:1412.6572](https://arxiv.org/abs/1412.6572)
16. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint [arXiv:1702.06280](https://arxiv.org/abs/1702.06280) (2017)
17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988. IEEE (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. He, W., Wei, J., Chen, X., Carlini, N., Song, D.: Adversarial example defenses: ensembles of weak defenses are not strong. arXiv preprint [arXiv:1706.04701](https://arxiv.org/abs/1706.04701) (2017)
20. Huang, R., Xu, B., Schuurmans, D., Szepesvári, C.: Learning with a strong adversary. arXiv preprint [arXiv:1511.03034](https://arxiv.org/abs/1511.03034) (2015)
21. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
22. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint [arXiv:1607.02533](https://arxiv.org/abs/1607.02533) (2016)
23. Kurakin, A., et al.: Adversarial attacks and defences competition. arXiv preprint [arXiv:1804.00097](https://arxiv.org/abs/1804.00097) (2018)
24. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
25. Li, X., Li, F.: Adversarial examples detection in deep networks with convolutional filter statistics. In: ICCV, pp. 5775–5783 (2017)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint [arXiv:1706.06083](https://arxiv.org/abs/1706.06083) (2017)

27. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. arXiv preprint [arXiv:1702.04267](https://arxiv.org/abs/1702.04267) (2017)
28. 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)
29. Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E.H., Freeman, W.T.: Visually indicated sounds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2405–2413 (2016)
30. 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)
31. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. arXiv preprint [arXiv:1511.04508](https://arxiv.org/abs/1511.04508) (2015)
32. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.: Malware detection by eating a whole exe. arXiv preprint [arXiv:1710.09435](https://arxiv.org/abs/1710.09435) (2017)
33. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 512–519. IEEE (2014)
34. Sabour, S., Cao, Y., Faghri, F., Fleet, D.J.: Adversarial manipulation of deep representations. arXiv preprint [arXiv:1511.05122](https://arxiv.org/abs/1511.05122) (2015)
35. Santoro, A., et al.: A simple neural network module for relational reasoning. In: Advances in Neural Information Processing Systems, pp. 4967–4976 (2017)
36. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: OverFeat: integrated recognition, localization and detection using convolutional networks. arXiv preprint [arXiv:1312.6229](https://arxiv.org/abs/1312.6229) (2013)
37. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint [arXiv:1312.6199](https://arxiv.org/abs/1312.6199) (2013)
38. Vadicamo, L., et al.: Cross-media learning for image sentiment analysis in the wild. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 308–317 (2017)
39. Wehrmann, J., Simões, G.S., Barros, R.C., Cavalcante, V.F.: Adult content detection in videos with convolutional and recurrent neural networks. *Neurocomputing* **272**, 432–438 (2018)
40. Xu, W., Evans, D., Qi, Y.: Feature squeezing: detecting adversarial examples in deep neural networks. arXiv preprint [arXiv:1704.01155](https://arxiv.org/abs/1704.01155) (2017)
41. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, vol. 32. Springer, Heidelberg (2006). <https://doi.org/10.1007/0-387-29151-2>