Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Adversarial image detection in deep neural networks

Fabio Carrara · Fabrizio Falchi · Roberto Caldelli · Giuseppe Amato · Rudy Becarelli

Received: date / Accepted: date

Abstract Deep neural networks are more and more pervading many computer vision applications and in particular image classification. Notwithstanding that, recent works have demonstrated that it is quite easy to create adversarial examples, i.e., images malevolently modified to cause deep neural networks to fail. Such images contain changes unnoticeable to the human eye but sufficient to mislead the network. This represents a serious threat for machine learning methods. In this paper, we investigate the robustness of the representations learned by the fooled neural network, analyzing the activations of its hidden layers. Specifically, we tested scoring approaches used for kNN classification, in order to distinguish between correctly classified authentic images and adversarial examples. These scores are obtained searching only between the very same images used for training the network. The results show that hidden layers activations can be used to reveal incorrect classifications caused by adversarial attacks.

Keywords Adversarial images detection · Deep Convolutional Neural Network · Machine Learning Security

1 Introduction

Deep Neural Networks (DNNs) have recently led to significant improvement in many areas of machine learning. They are the state of the art in many vision and content-base multimedia indexing tasks such as classification [23, 39, 45], recognition [40], image tagging [26], video captioning [5], face verification [35, 38], content-based image retrieval [1, 21], super resolution [13], cross-media searching [9, 14], and image forensics [4, 6, 49, 50].

F. Carrara, F. Falchi, G. Amato
ISTI - CNR
Via G. Moruzzi, 1, Pisa, Italy
E-mail: {name.surname}@isti.cnr.it

R. Caldelli, R. Becarelli
CNIT / MICC - UniFi
Viale Morgagni, 65, Firenze, Italy
E-mail: {name.surname}@unifi.it

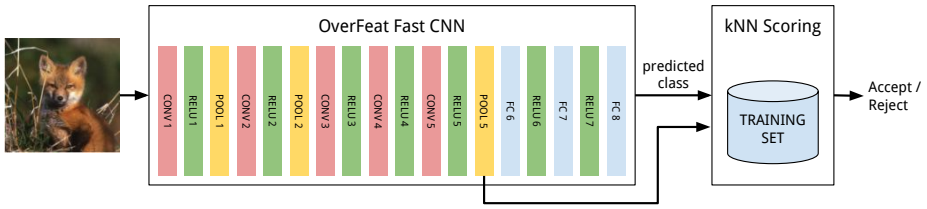


Fig. 1: Overview of our detection approach. The input image is classified by the CNN, but we consider the classification valid only if the kNN score of the predicted class based on deep features (*pool5*) is above a certain threshold.

Unfortunately, researchers have shown that machine learning models, including deep learning methods, are highly vulnerable to adversarial examples [15, 22, 31, 47]. An *adversarial example* is a malicious input sample typically created applying a small but intentional perturbation, such that the attacked model misclassifies it with high confidence [15]. In most of the cases, the difference between the original and perturbed image is imperceptible to a human observer. Moreover, adversarial examples created for a specific neural network have been shown to be able to fool different models with different architecture and/or trained on similar but different data [31, 47]. These properties are known as cross-model and cross-dataset generalization of adversarial examples and imply that adversarial examples pose a security risk even under a threat model where the attacker does not have access to the target's model definition, model parameters, or training set [24, 31].

It becomes crucial to understand how adversarial actions are perpetrated in order to consequently build up solutions to be robust against such attacks. Most of the effort of the research community in defending from adversarial attacks had gone into increasing the model robustness to adversarial examples via enhanced training strategies, such as adversarial training [15, 32] or defensive distillation [18, 34]. However, studies have shown [31] that those techniques only make the generation of adversarial examples more difficult without solving the problem. A different, less studied, approach is to defend from adversarial attacks by distinguishing adversarial inputs from authentic inputs.

In this work, we present an approach to detect adversarial examples in deep neural networks, based on the analysis of activations of the neurons in hidden layers (often called deep features) of the neural network that is attacked. Being deep learning a subset of representation learning methods, we expect the learned representation to be more robust than the final classification to adversarial examples. Moreover, adversarial images are generated in order to look similar to humans and deep features have shown impressive results in visual similarity related tasks such as content-based image retrieval [16, 41]. The results reported in this paper show that, given an input image, searching for similar deep features among the images used for training, allows to predict the correctness of the classification produced by a DNN.

In particular, we use traditional kNN classifiers scoring approaches as a measure of the confidence of the classification given by the DNN (see Figure 1). The score assigned by the kNN classifier to the class predicted by the DNN is used as a confidence measure of the reliability of the predicted class. The experiments

show that we are able to filter out many adversarial examples, while retaining most of the correctly classified authentic images. The choice of the discriminative threshold is a trade-off between accepted false positives (FP) and true positives (TP), where positive means non-adversarial.

A preliminary version of this method has been presented in [10]. We extended that work with a twofold contribution. First, we considered the behavior of different intermediate layers in determining adversarial examples. Second, we analyzed the cross-model applicability and resilience of the introduced approach when dealing with a diverse and more recent type of convolutional neural network.

The rest of the paper is structured as follows. Section 2 reviews the most relevant works in the field of adversarial attacks and their analysis. Section 3 provides background knowledge to the reader about DNNs, image representations (known as deep features), and adversarial generation. In section 4 our approach is presented, while in section 5 we describe the experimental settings we used to validate it. Finally, section 6 concludes the paper and presents some future research directions.

2 Related Work

2.1 Generation of Adversarial Examples

Szegedy et al. [47] firstly defined an *adversarial example* as the smallest perturbed image that induces a classifier to change prediction with respect to the original one. They successfully generated adversarial examples through the use of the box-constrained Limited-memory approximation of Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithm, and they proved that adversarial examples exhibit cross-model and cross-training set generalization properties. The same adversarial images also affect different models with different architectures also trained on different subsets of the training set. Cross-model and cross-training set generalization properties of adversarial examples are also confirmed in [48], where the authors explored the pixel space around adversarial images moving in random directions applying different kinds of noise. They show that adversarial images are not isolated, spurious points, but they occupy large regions of the pixel space that mislead models with similar classification boundaries. To overcome to the high computational cost of the L-BFGS approach, Goodfellow et al. [15] proposed the Fast Gradient Sign (FGS) method, which derives adversarial perturbations from the gradient of the loss function with respect to the input image, that can be efficiently computed by backpropagation. In [29], Nguyen et al. used evolutionary algorithms and gradient ascent optimizations to produce fooling images which are unrecognizable to human eyes but are classified with high confidence by DNNs. Papernot et al. [32] used forward derivatives to compute adversarial saliency maps that show which input feature have to be increased or decreased to produce the maximum perturbation of the last classification layer towards a chosen adversarial class. In [28], Moosavi et al. presented an algorithm to find image-agnostic (universal) adversarial perturbations for a given trained model, that are able fool the classifier with high probability when added to any input.

2.2 Defense Strategies for Adversarial Attacks

Different kinds of defenses against adversarial attacks have been proposed. Fast adversarial generation methods (such as FGS) enable adversarial training, that is the inclusion in the training set of adversarial examples generated on-the-fly in the training loop. Adversarial training allows the network to better generalize and to increase its robustness to this kind of attacks. However, easily optimizable models, such as models with non-saturating linear activations, can be easily fooled due to their overly confident linear responses to points that not occur in the training data distribution [15]. This concept is also discussed in [33], where the authors shows that there are tensions between model accuracy and resilience that has to be calibrated for each particular use case. They also provide a detailed description of the threat model for a machine learning system according to adversarial goals and capabilities, and a categorization of attacks and defenses in this framework. In [19], the authors found that denoising autoencoders can remove substantial amounts of the adversarial noise. However, when stacking the autoencoders with the original neural network, the resulting network can again be attacked by new adversarial examples with even smaller distortion. Thus, the authors proposed Deep Contractive Network, a model with an end-to-end training procedure that includes a smoothness penalty. Similarly, in [34] a two-phase training process known as *distillation* is used to increase the robustness of a model to small adversarial perturbations by smoothing the model surface around training points and vanishing the gradient in the directions an attacker would exploit. Still, attackers can find potential adversarial images using a non-distilled substitute model. Papernot et al. [31] showed that successfully attacks are possible even if the attacker does not have direct access to the model weights or architecture. In fact, the authors successfully performed adversarial attacks to remotely hosted models, and Kurakin et al. [24] also showed that attacks in physical scenarios, such as feeding a model with a printout adversarial example through a digital camera, are possible and effective. Detection of adversarial examples is still an open problem [32]. In this direction, Metzen et al. [27] proposed to add a parallel branch to the classifier and train it to detect whether the input is an adversarial example. However, the proposed branch is still vulnerable to adversarial attacks, and a more complicate adversarial training procedure is needed to increase the robustness of the whole system. The work most related to ours is from Grosse et al. [17], that proposed to detect adversarial inputs based on the statistical differences of their distribution from the one describing authentic inputs. However, the main drawback of this method is the inability to perform a per-sample detection.

3 Background

3.1 Deep Learning and Features

Deep learning methods are "representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level" [25].

Starting from 2012, deep learning has become state-of-the-art in image classification given the excellent results in ILSVRC challenges based on ImageNet [20, 23, 39, 42, 45]. In the context of Content-Based Image Retrieval, deep learning architectures are used to generate high level features. The relevance of the internal representation learned by the neural network during training have been proved by many recent works [2, 7, 12, 25]. In particular, the activation produced by an image within the intermediate layers of a deep convolutional neural network can be used as a high-level descriptor of the image visual content [2, 3, 11, 36, 39].

In this work, we employed the image representations extracted using OverFeat [39], a well-known and successful deep convolutional network architecture that have been studied for the analysis of adversarial attacks to convolutional neural networks [48], and for which implementations of adversarial generation algorithms are publicly available (see Section 5). Specifically, we used the Fast OverFeat network pre-trained on ImageNet (whose code and weights are publicly available at `https://github.com/sermanet/OverFeat`), and we selected the activations of the *pool5* layer as deep features for images.

3.2 Adversarial Generation

In this subsection we provide a brief description of the two approaches we used in our work to generate adversarial images.

Box Constrained L-BFGS [47, 48] Given an input image x and a DNN classifier $y = f(x)$, an adversarial example is generated finding the smallest distortion η such that $x' = x + \eta$ is misclassified by the target model, that is $f(x+\eta) \neq y$. The adversarial perturbation η is modeled as the solution of the following optimization problem:

$$\begin{aligned} \underset{\eta}{\text{minimize}} \quad & ||\eta|| + C \cdot H(y, y^A) \\ \text{subject to} \quad & L <= x + \eta <= U, \\ & y = f(x + \eta) \end{aligned} \tag{1}$$

where L and U are respectively lower and upper bound of pixel values, f is the attacked classifier, $H(y, y^A)$ is the cross-entropy loss computed between the output class probability distribution y and the target adversarial distribution y^A (which assigns probability 1 to the adversarial label and 0 to the remaining ones). The parameter C controls the trade-off between the magnitude of η and its fooling power. An adversarial perturbation is found by solving (1) using the box-contrained L-BFGS optimization algorithm. A first feasible value of C is found with a coarse grid search and then tuned with a binary search.

Fast Gradient Sign [15] In the Fast Gradient Sign method, the adversarial perturbation is proportional to the sign of the gradient back-propagated from the output to the input layer. Mathematically speaking, let θ be the parameters of a model, x the input to the model, y the targets (the desired output) associated with x, and $J(\theta, x, y)$ the cost function used to train the neural network. The cost function

can be linearized around the current value of θ, obtaining an optimal max-norm constrained perturbation:

$$\eta = \epsilon \cdot sign(\nabla_x J(\theta, x, y))$$

Note that the gradient can be computed easier using back-propagation. The adversarial input is given by $x' = x + \eta$.

4 Detecting adversarial examples

In this work, we propose to detect adversarial examples analyzing the representation learned in the hidden layers (deep features) of the fooled convolutional neural networks. Being deep learning a subset of representation learning methods, we expect the learned representation to be more robust than the final classification to adversarial examples. The recent renaissance of neural networks is due to the ability of learning powerful representations that can be used for classification but also for many other tasks such as recognition [40], face verification [38], content-based image retrieval [21], super resolution [13], cross-media searching [9, 14], etc. There are two reasons why deep features should be more robust: first, the adversarial generation algorithms are not meant to fool the representation itself but only the final classification; second, adversarial images are generated in order to look similar to authentic ones for humans, and deep features have shown impressive results in visual similarity related tasks such as content-based image retrieval [16, 41].

To detect whether an image is tampered, we first classify it with the DNN. Then, the activations of an hidden layer of the DNN (deep features) are used as query to perform a kNN search on the training set of the DNN. The deep features are used to judge image similarity. We then use the score assigned by a kNN classifier to the class predicted by the DNN as a measure of confidence of the classification. Please note, we do not rely on the classification produced by the kNN classifier, but we only use the score assigned to the class predicted by the DNN as a measure of confidence.

More formally, given a set of labeled images $X = \{(x_i, c_i)\}$ where x_i is an image and c_i is its class label, a kNN classifier assigns labels to an unknown image q considering the ordered results of its k nearest neighbors $NN(q, k) = \{(x_1, c_1) \dots (x_k, c_k)\}$, obtained performing a kNN search over X for a predefined distance function $d(x, y)$ between any two images. We define the distance function as $d(x, y) = ||\phi(x) - \phi(y)||_2$, where $\phi(x)$ is the deep feature extracted from the image x using the DNN, and $|| \cdot ||_2$ is the L2 norm. A score $s(q, c)$ is assigned to every class c found in the retrieved nearest neighbors of q as follows:

$$s(q, c) = \frac{\sum_{i=1}^{k} w_i \mathbb{1}\{c_i = c\}}{\sum_{i=1}^{k} w_i} \tag{2}$$

where q is a query image, c is the class for which we are computing the score, c_i is the groundtruth class of the i-th result in $NN(q, k)$, w_i the weight assigned to the same i-th result, and $\mathbb{1}\{c_i = c\}$ has value 1 if $c_i = c$, 0 otherwise. In Table 1, we report w_i assignments for famous variants of kNN classifiers.

Let x be the input image and c_x the class predicted by the DNN with a forward computation: $c_x = f(x)$. The kNN classifier is used to compute the score $s(x, c_x)$

kNN	Weighted kNN	d-Weighted kNN
$w_i = 1$	$w_i = \frac{1}{i}$	$w_i = \frac{1}{d(q, x_i)^2}$

Table 1: Weighting functions for the various kNN classifiers

of the class c_x predicted by the DNN. We decide that the classification is reliable, i.e. it is not an adversarial, if the score $s(x, c_x)$ is above a predefined threshold. The score $s(x, c_x)$ is, basically, a measure of the confidence of the classification given by the DNN. The intuition behind this choice is that while it is unlikely that a class correctly predicted by the DNN has the highest kNN score among the scores of all the classes, it is implausible that a correct classification has a very low score. As anticipated, the choice of the score threshold is a trade-off between false positives (FP) and true positives (TP), where FP are the adversarial examples (negatives) not detected using the specific threshold (false) and TP is the rate of correctly classified authentic images (positives) successfully identified (true). Please note that we do not rely on additional models or data other than the fooled DNN and its original training set for the extraction of deep features or for the detection task.

5 Experimental Settings

In this section, we describe the experimental setups used to evaluate the proposed adversarial detection approach. Our method has been evaluated as a binary classification of the correctness of the prediction given by a DNN, in which a positive outcome means that the prediction given by the DNN is trustful, while a negative outcome indicates that the prediction given by the DNN is spurious and have to be discarded.

The datasets used in this paper are the ILSVRC2012 subset of ImageNet [37] and the NIPS 2017 Adversarial Attacks and Defenses Kaggle Competitions [8] images. Both datasets share the same ILSVRC2012 label space. We selected these datasets for the experiments due to the big availability of classifiers pre-trained on ILSVRC2012. We search for similar images on the very same data that have been used for training the neural networks (i.e., the ILSVRC2012 training subset). Thus, the kNN methods rely on the very same information the neural network have seen during training.

We performed experiments with two DNN architectures. First, we applied our approach to *OverFeat-Fast* DNN [39] (Subsection 5.1), using a subset of the ILSVRC2012 validation images for generating adversarials. Second, we aligned with the configuration proposed in the context of the NIPS 2017 Adversarial Defenses Kaggle Competition [8] (Subsection 5.2). Hence, we tested our method on the network used as baseline in the competition (*InceptionV3*), and we used the DEV image set provided through Kaggle for generating adversarials. Generated adversarials and other resources have been made public available on the paper web page [1].

[1] `http://deepfeatures.org/adversarials/`

5.1 *OverFeat-Fast* network

In the experiments reported in this section, we selected the *OverFeat-Fast* DNN [39] given that this network (trained network on ILSVRC2012) was used in the papers in which both L-BFGS and Fast Gradient Sign (see Section 3.2) were presented.

As reported in Section 4, our approach performs a kNN similarity search over the images that were used for training the attacked DNN (the ILSVRC2012 training set). For generating adversarial examples, we selected images from the ILSVRC2012 validation set. In particular, we selected two subsets of images based on the classification results. The first set is composed by randomly selecting a correctly classified image for each of the $1,000$ ILSVRC classes, while the second set is composed by randomly selecting a wrongly classified image (for which the network has given a wrong prediction) for each of the same classes. We could not select a wrongly classified image in the class coded "n12057211" (yellow lady's slipper, yellow lady-slipper, Cypripedium calceolus, Cypripedium parviflorum) because all the instances of this class in the validation set are correctly classified by OverFeat. Thus, the two sets respectively count $1,000$ and 999 images. We named those subsets respectively *Authentic* and *Authentic Errors*, where 'authentic' stands for non-adversarial images.

For each image in the *Authentic* subset, we generated two adversarial images using both box constrained L-BFGS[2] and FGS[3] algorithms. For both methods, we used the default parameters in every generation and we randomly selected the target class, that is the class we fool the network to predict. We observed that L-BFGS algorithm failed to generate 8 adversarial images, in the sense that the class prediction of the generated adversarial image was the same of the original image. Those failures in the generation process could be avoided tuning the parameters of the algorithm for each input, but for sake of simplicity we discarded the failed adversarial examples, ending up with two sets of adversarial images respectively composed by 1000 images generated by FGS, and 992 images obtained with L-BFGS. The generated adversarial images are made publicly available[4] to make easier to reproduce the experiments.

We extracted the activations of the *pool5*, *fc6*, and *fc7* intermediate layer of the pre-trained OverFeat fast network [39] from the following sets of images: *Authentic*, *Authentic Errors*, *L-BFGS Adversarial*, *FGS Adversarial* and *ILSVRC2012 train set*. Both fully connected layers *fc6* and *fc7* are composed by 4096 floats, while activations of *pool5* are composed by 1024 6x6 feature maps. Following [43], we applied global average pooling (GAP) to the *pool5* feature maps, which acts as a structural regularizer, obtaining an image representation of 1024 floats. For the kNN classifier, we used the features extracted from ILSVRC2012 train set as labeled set X, and we defined the distance function $d(q, x)$ as the euclidean distance between the extracted features. We chose $k = 1,000$ to have a number of nearest neighbors of the same order of magnitude of the number of images per class in the labeled set. We tested also feature L2 normalization and dimensionality reduction using PCA+Whitening with 256 dimensionality.

[2] `https://github.com/tabacof/adversarial`

[3] `https://github.com/e-lab/torch-toolbox/tree/master/Adversarial`

[4] `http://deepfeatures.org/adversarials/`

Given an input image x, we compute the kNN score $s(x, c)$ for the class $c = f(x)$ predicted by OverFeat, and we discard this classification if the score is below a certain threshold. We computed the kNN score for each image in the *FGS Adversarial*, *L-BFGS Adversarial* and *Authentic Errors* image sets, and for each set we measure the ability to detect an adversarial input as the performance of a binary classification problem ('trustful' / 'spurious' classification).

5.1.1 Results

In Table 2, we report the detection accuracy of our proposed approach for different settings. Accuracy of the binary 'trustful' / 'spurious' classification is evaluated in the equal error rate (EER) setting, that is when we choose a threshold yielding equal false positive and false negative rates. The best results were obtained processing *pool5* deep features using PCA and Whitening, delivering an aggregated accuracy of roughly 85% irregardless of the generation method of the adversarial perturbation. The three scoring approaches considered revealed similar performance with DW-kNN and W-kNN more effective in detecting L-BFGS and FGS, respectively.

We found that adversarial examples generated with FGS are in general easier to detect using higher-level activations (*fc6*, *fc7*), while L-BFGS examples are significantly harder to filter out. This is reasonable because the optimization problem solved by L-BFGS produces small adversarial perturbation that are usually more difficult to detect even at higher layers at the cost of a more time-consuming generation process. While *pool5* activations perform best when applying a considerable amount of preprocessing on them (PCA+Whitening), fully connected activations perform well when used as is. We think a meaningful explanation for this behavior is given by the presence of dropout in the fully connected layers during the network training. Dropout is known to reduce the co-adaptation of hidden units and help learning robust, independent representations [44]. This may limit the benefits brought by dimensionality reduction schemes, such as PCA.

In Figure 2, we report the Receiver operating characteristic (ROC) curves of the DW-kNN scoring approach on adversarial images and errors. The curves illustrate the performance of the proposed binary classifier when varying the threshold on the score $s(x, f(x))$. As mentioned before, despite that *fc6* perform better detecting FGS adversarials, using *pool5* activations result in a detector more robust in general. However, we observed that higher kNN scores (which correspond to difficult L-BFGS adversarial to detect) usually reflects inter-class visual similarities that are independent from the adversarial nature of the input image (see Table 3). The ROC curve for errors is the worst, indicating that our approach confuse correctly and incorrectly classified authentic images. As mentioned before, the detection of authentic images incorrectly classified (errors) would be a desirable property of our approach, but it is not our main goal. We also report in Table 2 the detection accuracy obtained by the proposed methods when using the threshold value yielding equal error rates (EER) on both 'confident' / 'not confident' classes.

In the following, we focus on the DW-kNN score. In Figure 3, we report the true positive (for correctly classified authentic images) and false positive (for adversarial images and authentic errors) rates distributions as a function of the discriminant threshold applied on the score $s(x, f(x))$. Please note that for our method positive

pool5

Processing	Score	L-BFGS	FGS	Aggr.	Errors
None	kNN	70.7	69.9	70.3	58.1
	W-kNN	71.2	70.8	71.0	59.6
	DW-kNN	71.0	69.9	70.4	58.6
L2Norm	kNN	79.2	74.3	76.7	60.7
	W-kNN	81.4	76.4	78.9	62.9
	DW-kNN	81.7	76.6	79.1	61.6
PCA + Whiten	kNN	86.4	83.4	84.9	62.9
	W-kNN	85.9	**83.8**	84.8	**65.0**
	DW-kNN	**86.5**	83.5	**85.0**	63.6

fc6

Processing	Score	L-BFGS	FGS	Aggr.	Errors
None	kNN	76.4	**88.8**	**82.2**	65.8
	W-kNN	**77.0**	86.9	81.7	68.0
	DW-kNN	76.3	88.6	82.0	65.2
L2Norm	kNN	69.6	84.9	76.5	65.6
	W-kNN	70.2	86.8	77.7	71.3
	DW-kNN	69.4	84.7	76.3	65.4
PCA + Whiten	kNN	67.5	84.8	75.2	66.2
	W-kNN	68.6	86.6	76.6	**72.0**
	DW-kNN	67.9	84.4	75.3	66.0

fc7

Processing	Score	L-BFGS	FGS	Aggr.	Errors
None	kNN	**77.0**	87.8	**82.1**	67.0
	W-kNN	73.2	**88.8**	80.3	69.7
	DW-kNN	76.7	87.8	81.9	66.7
L2Norm	kNN	64.7	86.2	73.9	64.7
	W-kNN	64.1	87.8	74.1	69.9
	DW-kNN	64.7	86.1	73.9	64.8
PCA + Whiten	kNN	63.4	86.3	73.1	64.6
	W-kNN	62.7	88.0	73.2	**70.5**
	DW-kNN	63.6	85.9	73.1	64.8

Table 2: Detection accuracy in the equal error rate (EER) score threshold setting for various activation layers, score functions, and features processing. We report results for both type of adversarial (L-BFGS and FGS) and the aggregated accuracy (Aggr.). In the last column, we report the detection rate of erroneous classifications not due to adversarial examples.

means non-adversarial. The results show that using very low discrimination score values (about 0.002), it is possible to correctly filter out more than 50% of the adversarial examples created by L-BFGS and more than 40% of the ones created by FGS, while retaining more than 98% of authentic images correctly classified. As a positive side-effect, we also discard around 10% of images the DNN would misclassify. The low threshold values reveal that while the DW-kNN score would not

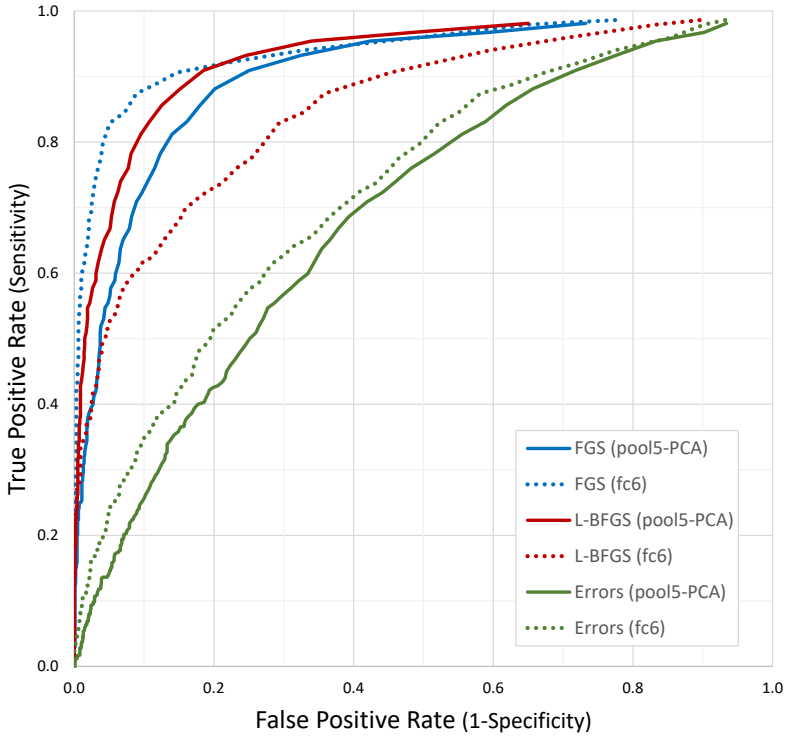


Fig. 2: Receiver operating characteristic (ROC) curves of the binary classification ('prediction is right' or 'prediction is wrong') for the various types of images. The curves are obtained varying the discrimination threshold on the score assigned by the DW-kNN classifier to the class predicted by the CNN. We only report the curves of the best performing configurations, that are *fc6* with no processing, and *pool5* with PCA and whitening. Notice that when using *pool5* we obtain a detection less sensible to a particular adversarial generation process.

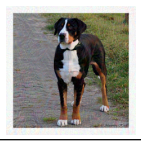
be effective in classifying the images, values below 0.003 are unlikely for authentic images.

The same results can be seen from the score densities reported in Figure 4, in which we can observe a distinction between the score densities of adversarial images and the ones of authentic images. Some simple statistics on those densities (such as the mean) could be computed on-line in the system hosting the model to isolate a particular source of adversarial examples, hence denying the access to the service to an attacker.

Finally in Table 3, we report examples of successful detections and failures of our approach applied to the generated adversarial images. We observed that higher kNN scores (which correspond to difficult adversarial to detect) usually reflects inter-class visual similarities that are independent from the adversarial nature of the input image.

Algorithm	Adv. Image	Actual	Predicted	Predicted NN	s
L-BFGS		bikini, two-piece	pomegranate		0.01
FGS		brassiere, bra, bandeau	Chihuahua		0.01
FGS		revolver, six-gun, six-shooter	mousetrap		0.00
L-BFGS		assault rifle, assault gun	Border terrier		0.00

(a) Examples of good detections of adversarial images with content that might be filtered. Low scores reflect a low confidence of image authenticity.

Algorithm	Adv. Image	Actual	Predicted	Predicted NN	s
FGS		chime, bell, gong	barometer		0.13
L-BFGS		basenji	Arctic fox, white fox, Alopex lagopus		0.13
FGS		Greater Swiss Mountain dog	Bernese mountain dog		0.11
FGS		jeep, landrover	pickup, pickup truck		0.11

(b) Examples of bad detections of adversarial images. Those are adversarial images for which our approach wrongly assigned a high score. However, this is mainly due to the visual similarity between the actual and fooled class.

Table 3: Examples of detections of adversarial images obtained by our best approach (*pool5*+PCA+DW-kNN). From left to right, columns respectively report: the adversarial generation algorithm, the generated adversarial image, its original class, the class predicted by the DNN, the nearest neighbor image (in terms of L2 distance between average-pooled *pool5* activations) belonging to the predicted class, and the DW-kNN score s for the predicted class. A low score indicates that the adversarial is correctly detected (a) while a high score means that our approach is wrongly confident about the prediction of the CNN (b). The results show that high scoring adversarials examples often share some common visual aspects and semantic with the predicted (adversarial) class, resulting in a more challenging detection.

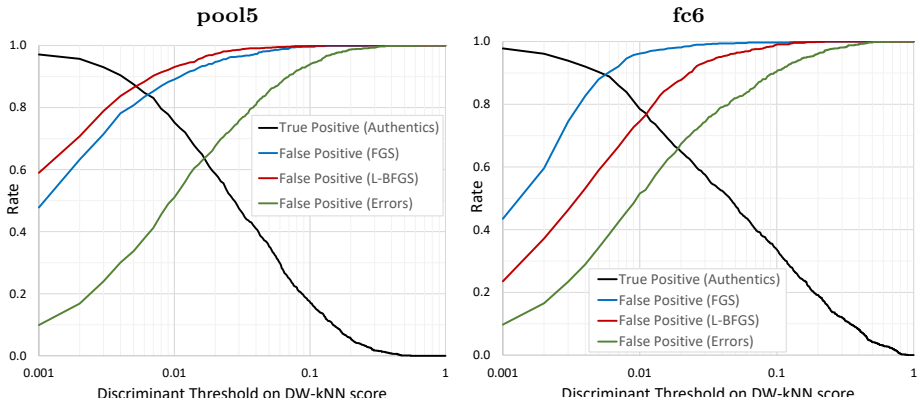


Fig. 3: True positive and false positive rates using as discrimination threshold between correctly and incorrectly classified images the score assigned by the DW-kNN classifier to the class predicted by the CNN. The *pool5* (on the left) and *fc6* (on the right) layers have been used as features with PCA and Whitening processing.

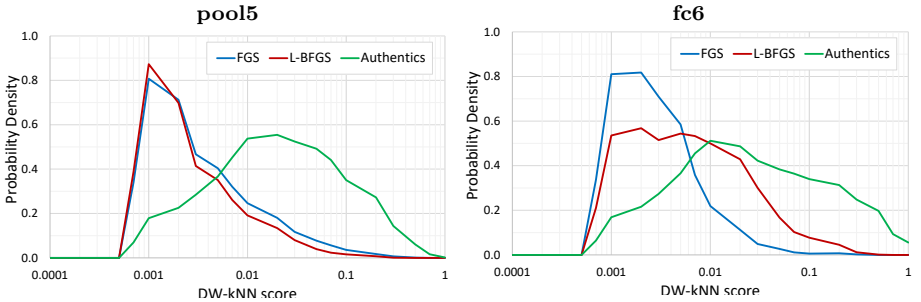


Fig. 4: Density of the DW-kNN scores for both adversarial and authentic images. We report densities of scores using *pool5* (on the left) and *fc6* (on the right) as feature and PCA+Whitening as processing.

5.2 *InceptionV3* Network on Kaggle NIPS 2017 Adversarial Competition Dataset

In the following experiments, we applied our detection algorithm to a more recent model, that is InceptionV3 [46], which has been used as baseline classifier in the NIPS 2017 Adversarial Attacks and Defenses Kaggle Competitions [8]. Thus, instead of selecting random images from the ILSVRC2012 validation set, we used the DEV images set released for the competition. The DEV set is composed of 1,000 images that are not part of the ImageNet dataset, yet they have been manually labelled using the ILSVRC2012 labels.

We followed the same methodology as in Section 5.1: we set apart from the DEV set the images that were incorrectly classified by InceptionV3; for the remaining images, we applied four different perturbation schemes (we generated the last two adversarial perturbations using the *cleverhans* library [30]):

noop the image is unchanged;

random_noise the image is perturbed adding random gaussian noise in the interval $[-16, +16]$;

fgsm the image is perturbed with FGSM (see Section 3.2) with $\epsilon = 16$ choosing a random target class;

iter_step_class the image is perturbed with 20 iterations of FGSM algorithm with $\epsilon = 1$; the target class is randomly chosen and kept fixed for all the iterations; the total perturbation is clipped to be in the interval $[-16, +16]$.

Since ***random_noise*** images are not meant to directly attack the classifier, we considered them authentic images. As in Section 5.1, we removed failed adversarial images produced by ***fgsm*** or ***iter_step_class***, i.e. adversarial perturbed images for which the prediction of the network has not changed. We also left out images produced by ***random_noise*** that are misclassified by the network, in order not to tamper the analysis of our adversarial detection approach with errors committed naturally by the classifier. From this procedure, we obtained a test set comprising correctly classified authentic images and successfully generated adversarial images.

We performed a thorough exploration on the choice of the deep features to be extracted from the InceptionV3 model. We extracted convolutional features after each inception module, and we obtained compact representations applying global average pooling. We also performed PCA dimensionality reduction to 256 components, since it had been proved beneficial for convolutional features in previous experiments (see Section 5.1).

5.2.1 Results

In Table 4, we report the equal-error-rate (EER) accuracy obtained on the test set for each layer used to extract deep features and considering each kNN scoring scheme. We noticed that the effectiveness of the detection increases adopting higher-level layers of the network, until we reach the fooled layers, where the accuracy drops. The first layers of the network produce activations not representative enough of class-level semantic concepts, while the last layers are steered by the adversarial crafting algorithms. This behavior suggests that current adversarial crafting algorithms do not steer all the internal semantic representations inside the network. Thus, we are able to find a good compromise between representativeness and robustness to adversarial manipulation.

In Figure 5, we report the true positive (correctly classified authentic images predicted as non-adversarial) and false positive (adversarial images and authentic errors not detected by our method) rates distributions as a function of the discriminant threshold applied on the score (positive means non-adversarial). Although lower-level features seem not to affect the detection of less perturbed images (i.e. FGSM), we noticed that they play a fundamental role to detect strong adversarial such as ***iter_target_class*** (Figure 5 on the left). This is reasonable given that the stronger attack we analyzed, i.e. ***iter_target_class***, tends to affect multiple layers in the last stage of the network, thus being better detected using more internal lower-level activations, while FGSM can be easily detected using higher-level layers.

kNN

Features	***fgsm***	***iter_target_class***	**Aggr.**	**Errors**
maxpool_5a_3x3	50.1	51.9	51.1	31.1
mixed_5b	53.0	54.4	53.8	35.4
mixed_5c	56.6	58.4	57.6	41.2
mixed_5d	57.6	59.8	58.9	43.2
mixed_6a	62.6	65.7	64.4	51.5
mixed_6b	62.3	72.7	68.2	52.8
mixed_6c	70.7	80.6	76.3	60.5
mixed_6d	70.9	**83.0**	**77.8**	64.5
mixed_6e	72.5	74.2	73.5	71.3
mixed_7a	73.0	77.2	75.4	71.4
mixed_7b	73.7	70.5	71.9	75.2
mixed_7c	**74.0**	52.0	61.5	**78.5**

W-kNN

Features	***fgsm***	***iter_target_class***	**Aggr.**	**Errors**
maxpool_5a_3x3	58.2	60.8	59.7	45.6
mixed_5b	59.5	63.0	61.5	49.1
mixed_5c	62.2	66.9	64.9	54.6
mixed_5d	63.8	68.7	66.6	57.5
mixed_6a	68.7	74.4	72.0	61.7
mixed_6b	67.8	79.9	74.7	63.1
mixed_6c	71.0	82.5	77.6	68.2
mixed_6d	71.7	**84.6**	**79.1**	69.4
mixed_6e	74.3	75.6	75.1	72.1
mixed_7a	**75.5**	78.4	77.2	75.0
mixed_7b	75.1	71.3	72.9	76.5
mixed_7c	74.4	52.3	61.9	**80.5**

DW-kNN

Features	***fgsm***	***iter_target_class***	**Aggr.**	**Errors**
maxpool_5a_3x3	58.3	60.9	59.8	45.7
mixed_5b	59.5	63.0	61.5	49.1
mixed_5c	62.2	66.9	64.9	54.5
mixed_5d	63.8	68.8	66.6	57.8
mixed_6a	68.8	74.4	72.0	62.7
mixed_6b	68.0	80.0	74.8	64.0
mixed_6c	71.4	82.3	77.6	69.2
mixed_6d	71.6	**85.0**	**79.2**	69.5
mixed_6e	72.8	75.5	74.4	72.6
mixed_7a	74.4	78.3	76.7	73.1
mixed_7b	74.4	71.4	72.7	76.1
mixed_7c	**74.5**	52.8	62.2	**79.0**

Table 4: Equal error rate (EER) detection accuracies for each (scoring scheme, layer) combination. The accuracies are computed putting together each subset (***fgsm*** or ***iter_target_class***) with the set authentic images. The **Aggr.** column reports a weighted mean of the accuracies on all subsets. The **Errors** column report the recover accuracy for natural (non-adversarial) errors committed by the network.

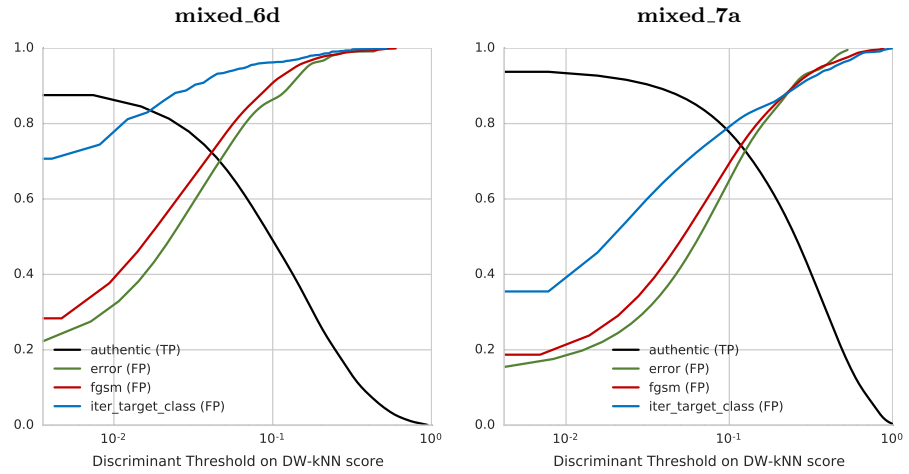


Fig. 5: True positive and false positive rates when varying the score threshold. The scores are computed using ***mixed_6d*** (on the left) and ***mixed_7a*** (on the right) of the InceptionV3 classifier. DW-kNN scoring scheme was selected for both.

Weighted scoring schemes (W-kNN and DW-kNN) seem to outperform the naive kNN scheme, specially when combined with low level features (see Figure 6). Overall, the best performance is obtained using the ***mixed_6d*** layer and the DW-kNN scheme, reaching a mean EER accuracy of 79.2%. Moreover, our method is able to recover around 80% of natural errors committed by the network. This rate is considerably higher than the ones presented in Table 2 due to the presence of ***random_noise*** images, that produced lots of easily recoverable misclassifications.

6 Conclusions and Future Work

In this paper, we presented an approach to detect adversarial examples crafted for fooling deep neural network classifiers. The overall goal is filtering out malicious images. Being deep learning methods based on representation learning, we decided to consider activations of neurons in hidden layers (the representation learned) in order to detect adversarials. In particular, we inspect the activations of intermediate layers for both adversarial and authentic inputs, and we defined an authenticity confidence score based on kNN similarity searching among the images used for training. Experiments on the two most cited adversarial generation techniques (L-BFGS and FGSM) on the very same neural network used in the original papers (i.e., Overfeat, InceptionV3) have been carried out. In the experiments, we considered various kNN score functions and hidden layers.

The proposed approach allows to filter about 80% of adversarial examples retaining more than 90% of the correctly classified authentic images (see Figure 3). We also showed that the probability density function of our authenticity confidence obtained over adversarial examples significantly differs from the one obtained for authentic images. Moreover, some examples are suggesting that hard adversarial

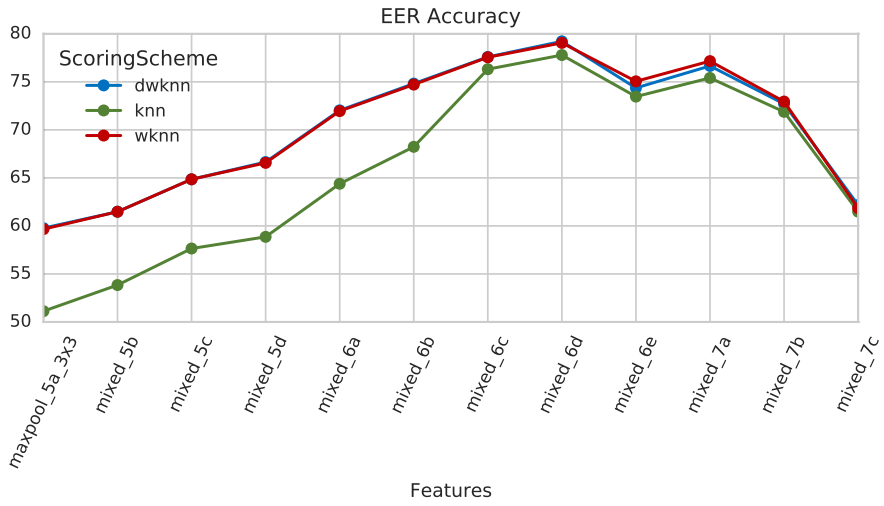


Fig. 6: The Equal Error Rate accuracy obtained on the NIPS DEV set using different intermediate activations of the InceptionV3 as deep features. Note that the effectiveness of the detection gets better when using deeper representations, and rapidly decreases when we reach the fooled layers at the end of the network.

examples are the ones for which actual and target classes are similar or have similar visual patterns.

With respect to the previous work we presented at CBMI 2017, we also tested our method on the InceptionV3 DNN, generating adversarial images from the NIPS 2017 Adversarial Attacks and Defenses Kaggle Competitions. Moreover, we compared the results obtained using the features obtained from each internal layer of the InceptionV3. Best results were obtained using layer ***mixed_6d*** with an overall EER Accuracy of 80%. These results confirm the effectiveness of our approach and its general applicability to deep neural network classifiers.

Acknowledgements 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 the project ESPRESS (Smartphone identification based on onboard sensors for security applications) co-funded by Fondazione Cassa di Risparmio di Firenze (Italy) within the Scientific Research and Technological Innovation framework. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

1. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., Vairo, C.: Deep learning for decentralized parking lot occupancy detection. Expert Systems with Applications **72**, 327–334 (2017)

2. Amato, G., Falchi, F., Gennaro, C., Rabitti, F.: Yfcc100m-hnfc6: A large-scale deep features benchmark for similarity search. In: International Conference on Similarity Search and Applications, pp. 196–209. Springer (2016)
3. Amato, G., Falchi, F., Vadicamo, L.: Visual recognition of ancient inscriptions using convolutional neural network and fisher vector. Journal on Computing and Cultural Heritage (JOCCH) **9**(4), 21 (2016)
4. Amerini, I., Uricchio, T., Ballan, L., Caldelli, R.: Localization of jpeg double compression through multi-domain convolutional neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1865–1871 (2017). DOI 10.1109/CVPRW.2017.233
5. Baraldi, L., Grana, C., Cucchiara, R.: Hierarchical boundary-aware neural encoder for video captioning. arXiv preprint arXiv:1611.09312 (2016)
6. Bayar, B., Stamm, M.C.: A deep learning approach to universal image manipulation detection using a new convolutional layer. In: 4th ACM Workshop on Information Hiding and Multimedia Security, pp. 5–10 (2016)
7. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013). DOI 10.1109/TPAMI.2013.50. URL `http://dx.doi.org/10.1109/TPAMI.2013.50`
8. Brain, G.: NIPS 2017: Competition on Adversarial Attacks and Defenses. `https://www.kaggle.com/nips-2017-adversarial-learning-competition` (2017). [Online; accessed 19-January-2018]
9. Carrara, F., Esuli, A., Fagni, T., Falchi, F., Fernández, A.M.: Picture it in your mind: Generating high level visual representations from textual descriptions. arXiv preprint arXiv:1606.07287 (2016)
10. 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, p. 38. ACM (2017)
11. Chandrasekhar, V., Lin, J., Morère, O., Goh, H., Veillard, A.: A practical guide to cnns and fisher vectors for image instance retrieval. arXiv preprint arXiv:1508.02496 (2015)
12. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: Icml, vol. 32, pp. 647–655 (2014)
13. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence **38**(2), 295–307 (2016)
14. Dong, J., Li, X., Snoek, C.G.: Word2visualvec: Cross-media retrieval by visual feature prediction. arXiv preprint arXiv:1604.06838 (2016)
15. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
16. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. In: European Conference on Computer Vision, pp. 241–257. Springer (2016)
17. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 (2017)

18. Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P.: Adversarial perturbations against deep neural networks for malware classification. arXiv preprint arXiv:1606.04435 (2016)
19. Gu, S., Rigazio, L.: Towards deep neural network architectures robust to adversarial examples. arXiv preprint arXiv:1412.5068 (2014)
20. 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)
21. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
22. Klarreich, E.: Learning securely. Commun. ACM **59**(11), 12–14 (2016). DOI 10.1145/2994577. URL `http://doi.acm.org/10.1145/2994577`
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
24. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
25. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
26. Li, X., Uricchio, T., Ballan, L., Bertini, M., Snoek, C.G., Bimbo, A.D.: Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. ACM Computing Surveys (CSUR) **49**(1), 14 (2016)
27. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267 (2017)
28. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv preprint arXiv:1610.08401 (2016)
29. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
30. Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., Hambardzumyan, K., Juang, Y.L., Kurakin, A., Sheatsley, R., Garg, A., Lin, Y.C.: cleverhans v2.0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768 (2017)
31. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697 (2016)
32. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on, pp. 372–387. IEEE (2016)
33. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814 (2016)
34. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Security and Privacy (SP), 2016 IEEE Symposium on, pp. 582–597. IEEE (2016)
35. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: British Machine Vision Conference (2015)

36. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 512–519. IEEE (2014)
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). DOI 10.1007/s11263-015-0816-y
38. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
39. 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 (2013)
40. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2014)
41. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 806–813 (2014)
42. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
44. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research **15**(1), 1929–1958 (2014)
45. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
46. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
47. 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)
48. Tabacof, P., Valle, E.: Exploring the space of adversarial images. In: Neural Networks (IJCNN), 2016 International Joint Conference on, pp. 426–433. IEEE (2016)
49. Tuama, A., Comby, F., Chaumont, M.: Camera model identification with the use of deep convolutional neural networks. In: Information Forensics and Security (WIFS), 2016 IEEE International Workshop on (2016)
50. Ying, Z., Goha, J., Wina, L., Thinga, V.: Image region forgery detection: A deep learning approach. In: Singapore Cyber-Security Conference (SG-CRC), pp. 1–11 (2016)