A Counter-Forensic Method for CNN-Based Camera Model Identification

Conference Paper · July 2017

DOI: 10.1109/CVPRW.2017.230

Luca Bondi
Politecnico di Milano
32 PUBLICATIONS **897** CITATIONS
SEE PROFILE

Stefano Tubaro
Politecnico di Milano
442 PUBLICATIONS **5,599** CITATIONS
SEE PROFILE

Edward J. Delp
Purdue University
626 PUBLICATIONS **14,873** CITATIONS
SEE PROFILE

A Counter-Forensic Method for CNN-Based Camera Model Identification

David Güera
Purdue University
West Lafayette, Indiana

Yu Wang
Purdue University
West Lafayette, Indiana

Luca Bondi
Politecnico di Milano
Milan, Italy

Paolo Bestagini
Politecnico di Milano
Milan, Italy

Stefano Tubaro
Politecnico di Milano
Milan, Italy

Edward J. Delp
Purdue University
West Lafayette, Indiana

Abstract

An increasing number of digital images are being shared and accessed through websites, media, and social applications. Many of these images have been modified and are not authentic. Recent advances in the use of deep convolutional neural networks (CNNs) have facilitated the task of analyzing the veracity and authenticity of largely distributed image datasets. We examine in this paper the problem of identifying the camera model or type that was used to take an image and that can be spoofed. Due to the linear nature of CNNs and the high-dimensionality of images, neural networks are vulnerable to attacks with adversarial examples. These examples are imperceptibly different from correctly classified images but are misclassified with high confidence by CNNs. In this paper, we describe a counter-forensic method capable of subtly altering images to change their estimated camera model when they are analyzed by any CNN-based camera model detector. Our method can use both the Fast Gradient Sign Method (FGSM) or the Jacobian-based Saliency Map Attack (JSMA) to craft these adversarial images and does not require direct access to the CNN. Our results show that even advanced deep learning architectures trained to analyze images and obtain camera model information are still vulnerable to our proposed method.

1. Introduction

The recent increase in the number of digital images that are being uploaded and shared online has given rise to unique privacy and forensic challenges [1]. Among those challenges, verifying the integrity and authenticity of these widely circulated pictures is one of the most critical and complex tasks [2, 3].

In the last few years, the digital media forensic community has explored several techniques to evaluate the truthfulness of digital images and media [4, 5]. Due to its multiple applicable scenarios, research efforts have focused on camera model identification [6, 7, 8, 9, 10]. Determining the camera model used to take a picture can be very important in criminal investigations such as copyright infringement cases or where it is required to identify the authors of pedo-pornographic material.

Camera model identification can also be considered an important preliminary step to reduce the set of camera instances when we try to detect a unique camera instance rather than just the make and model [8]. In addition, being able to identify the camera model by inspecting small image regions is a viable method to uncover manipulation operations that could have been done to the image (*e.g.* splicing) [11].

Current camera model identification detectors make use of the fact that each camera model completes a distinctive set of tasks on each image when the device acquires the image. Examples of these tasks include the use of different JPEG compression schemes, application of proprietary methods for CFA demosaicing, and “defects” in the optical image path. Due to these characteristic operations, a singular “footprint” is embedded in each picture. This information can be utilized to identify the camera model, and perhaps the exact camera, that has been used to capture an image or record a video sequence.

Due to the inherent and growing complexity of the image acquisition pipeline of modern image capturing devices, it is a difficult challenge to adequately model the set of operations that a camera has to execute to capture an image. Successful attempts that use hand-crafted features to model the traces left by some of these operations can be found in [7, 12, 13, 14, 15, 10, 16, 17].

The use of deep learning techniques for image and video classification tasks [18, 19, 20] has shown that it is also possible to learn characteristic features that model a problem space directly from the data itself. This offers a viable path to leverage the growing amount of available image data.

These modern approaches are data-driven in that they learn directly from the data rather than imposing a predetermined analytical model.

The data-driven model has recently proved valuable for forensics applications [21, 22, 23, 24]. Initial exploratory solutions targeting camera model identification [25, 26, 27] show that it is possible to use CNNs to learn discriminant features directly from the observed known images, rather than having to use hand-crafted features. The use of CNNs also makes it possible to capture characteristic traces left by non-linear and hard to model operations present in the acquisition pipeline.

With the introduction of CNNs as detectors for camera model identification, a new vector for counter-forensic attacks is presented for a malevolent skilled individual. The idea of counter-forensics was first introduced in [28], where the authors presented the concept of fighting against image forensics with a practical application, namely a method for resampling an image without introducing pixel correlations. An up-to-date survey of the last counter-forensics advances can be found in [29].

Before exploring the vulnerabilities of CNN-based camera model detectors, it is important to note that detectors that rely on hand-crafted features are not immune to similar counter-forensics attacks. As explained in [30], digital camera fingerprints are vulnerable to forging. In particular, if an attacker obtains access to images from a given camera, they can estimate its fingerprint and “paste” it into an arbitrary image to make it look as if the image came from the camera with the stolen fingerprint. An early attempt to investigate such counter-forensic methods appeared in [31].

As presented in [32], several machine learning models, including state-of-the-art convolutional neural networks, are vulnerable to adversarial attacks. This means that these machine learning models misclassify images that are only slightly different from correctly classified images. In many cases, an ample collection of models with different architectures trained on different subsets of the training data misclassify the same adversarial example [33].

Although there are techniques such as adversarial training [32] or defensive distillation [34] that can slightly reduce the incidence of adversarial examples in CNN-based detectors, defending against adversarial examples is still an on-going challenge in the deep learning community. Adversarial attacks are hard to defend against because they require machine learning models that produce correct outputs for every possible input. The imposition of linear behavior when presented with inputs similar to the training data, though desirable, is precisely the main weakness of CNNs [33]. Due to the massive amount of possible inputs that a CNN can be presented with, it is remarkably simple to find adversarial examples that look unmodified to us but are misclassified by the network. Designing a truly adaptive defense against adversarial images remains an open problem.

In this paper, we propose a counter-forensic method to subtly change an image to induce an error in its estimated camera model when analyzed by a CNN-based camera model detector. We leverage the recent developments to rapidly generate adversarial images. We test our counter-forensic method, using two well established adversarial image crafting techniques [33, 35], against an advanced deep learning architecture [36] carefully trained on a reference camera model dataset. Our results show that even modern and properly trained CNNs are susceptible to simple adversarial attacks. Note that our method only requires access to the predictions of the CNN-based camera model identification detector and does not need access to the weights of the CNN.

2. CNN-Based Camera Model Identification

In this section, we provide a brief overview of convolutional neural networks sufficient to understand the rest of this paper and show how they can be used as camera model detectors. For a more detailed description, please refer to one of the several available tutorials in the literature [37, 38].

Convolutional neural networks are a special type of neural networks, biologically inspired by the human visual cortex system, that consist of a very high number of interconnected nodes, or neurons. The architecture of a CNN is designed to take advantage of the 2D structure of an input image. This is achieved with local connections and tied weights followed by some forms of pooling which results in translation invariant features. The nodes of the network are organized in multiple stacked layers, each performing a simple operation on the input.

The set of operations in a CNN typically comprises convolution, intensity normalization, non-linear activation and thresholding, and local pooling. By minimizing a cost function at the output of the last layer, the weights of the network are tuned so that they are able to capture patterns in the input data and extract distinctive features.

In a CNN, the features are learned using backpropagation [39] coupled with an optimization method such as gradient descent [40] and the use of large annotated training datasets. The shallower layers of the networks usually learn low-level visual features such as edges, simple shapes and color contrast, whereas deeper layers combine these features to identify complex visual patterns. Finally, fully-connected layers coupled with a softmax layer are commonly used to generate an output class label that minimizes the cost function.

For example, in the context of image classification, the last layer is composed of N nodes, where N is the number of classes, that define a probability distribution over the N visual category. The value of a given node $p_i,\ i = 1, \ldots, N$

belonging to the last layer represents the probability of the input image to belong to the visual class c_i.

To train a CNN model for a specific image classification task we need to define the hyperparameters of the CNN, which range from the sequence of operations to be performed, to the number of layers or the number and shape of the filters in convolutional layers. We must also define a proper cost function to be minimized during the training process. Finally, a dataset of training and test images, annotated with labels according to the specific task (*e.g.* camera models in our work) needs to be prepared.

Figure 1 shows an example of a CNN-based pipeline for camera model identification similar to the one presented in [26]. To train the CNN architecture, we use a given set of training and validation labeled image patches coming from N known camera models. For each color image I, associated to a specific camera model L, K non-overlapping patches P_k, $k \in [1, K]$, of size 32×32 pixels are randomly extracted. Each patch P_k inherits the same label L of the source image. As trained CNN model $\mathcal{M}$, we select the one that provides the smallest loss on validation patches.

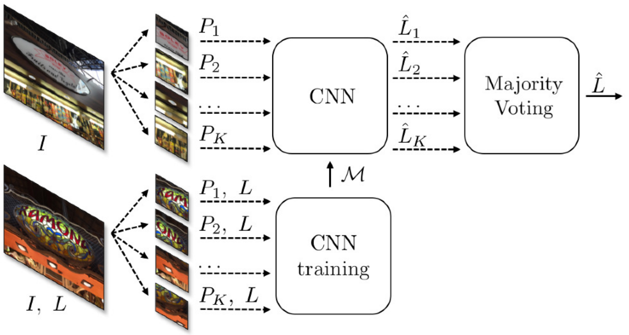


Figure 1. Example of a pipeline for camera model identification. The patches extracted from each training image I (bottom) inherit the same label L of the image. These patches are used in the CNN training process. For each patch P_k from the image I under analysis (top), a candidate label $\hat{L}_k$ is produced by a trained CNN model $\mathcal{M}$. The predicted label $\hat{L}$ for analyzed image I is obtained by majority voting.

When a new image I is under analysis, the camera model used to acquire it is estimated as follows. A set of K patches is obtained from image I as described above. Each patch P_k is processed by CNN model $\mathcal{M}$ in order to assign a label $\hat{L}_k$ to each patch. The predicted model $\hat{L}$ for image I is obtained through majority voting on $\hat{L}_k$, $k \in [1, K]$.

3. Proposed Method

Figure 2 shows the block diagram of our proposed counter-forensic method. Our method consists of an adversarial image generator module that can be added to a CNN-based camera model evaluation pipeline. In Figure 2, we assume a similar structure to the previously presented pipeline in Section 2. Our adversarial image generator module takes as input the set of K patches that have been extracted from the image I that is being analyzed. When presented with new image patches, our module can work in two different modes.

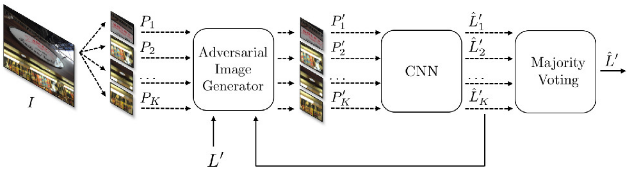


Figure 2. Block diagram of our proposed method.

In the first operation mode, the adversarial image generator module does an untargeted image manipulation, that is, it does not try to perturb the image patches to produce a specific misclassification class. Instead, we use the derivative of the loss function of the CNN with respect to the input image patches to add a perturbation to the images. The derivative is computed using backpropagation with the labels $\hat{L}'_k$, $k \in [1, K]$ that are given by the CNN detector when it first processes the unmodified image patches. This procedure is known as the fast gradient sign method (FGSM) [33].

In the second operation mode, the adversarial image generator module does a targeted image manipulation. In this case, we try to perturb the image patches to produce a specific misclassification class L', different from the true real label L that is associated with the analyzed image I and its associated P_k patches. In this mode of operation, we exploit the forward derivative of a CNN to find an adversarial perturbation that will force the network to misclassify the image patch into the target class by computing the adversarial saliency map. Starting with an unmodified image patch, we perturb each feature by a constant offset ϵ. This process is repeated iteratively until the target misclassification is achieved. This procedure is known as the Jacobian-based saliency map attack (JSMA) [35].

We present a detailed overview of both FGSM and JSMA techniques as follows.

3.1. Fast Gradient Sign Method

In [33], the fast gradient sign method was introduced for generating adversarial examples using the derivative of the loss function of the CNN with respect to the input feature vector. Given an input feature vector (*e.g.* an image), FGSM perturbs each feature in the direction of the gradient by magnitude ϵ, where ϵ is a parameter that determines the perturbation size. For a network with loss $J(\Theta, x, y)$, where Θ represents the CNN predictions for an input x and y is the correct label of x, the adversarial example is generated as

$$x^* = x + \epsilon \text{sign}(\nabla_x J(\Theta, x, y))$$

With small ϵ, it is possible to generate adversarial images that are consistently misclassified by CNNs trained using the MNIST and CIFAR-10 image classification datasets with a high success rate [33].

3.2. Jacobian-Based Saliency Map Attack

In [35], an iterative method for targeted misclassification was proposed. By exploiting the forward derivative of a CNN, it is possible to find an adversarial perturbation that will force the network to misclassify into a specific target class. For an input x and a convolutional neural network C, the output for class j is denoted $C_j(x)$. To achieve an output of target class t, $C_t(x)$ must be increased while the probabilities $C_j(x)$ of all other classes $j \neq t$ decrease, until $t = \arg\max_j C_j(x)$. This is accomplished by exploiting the adversarial saliency map, which is defined as

$$S(x,t)[i] = \begin{cases} 0, \text{if } \frac{\partial C_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial C_j(x)}{\partial x_i} > 0 \\ (\frac{\partial C_t(x)}{\partial x_i}) | \sum_{j \neq t} \frac{\partial C_j(x)}{\partial x_i} |, \text{otherwise} \end{cases}$$

for an input feature i. Because we work with images in this paper, in our case each input feature i corresponds to a pixel i in the image input x. Starting with a normal sample x, we locate the pair of pixels $\{i, j\}$ that maximize $S(x,t)[i] + S(x,t)[j]$, and perturb each pixel by a constant offset ϵ. This process is repeated iteratively until the target misclassification is achieved. This method can effectively produce MNIST dataset examples that are correctly classified by human subjects but misclassified into a specific target class by a CNN with a high confidence.

3.3. Implementation Details

To implement our counter-forensic method, we have used the software library *cleverhans* [41]. The library provides standardized reference implementations of adversarial image generation techniques and adversarial training. The library can be used to develop more robust CNN architectures and to provide standardized benchmarks of CNNs performance in an adversarial setting. As noted in [41], benchmarks constructed without a standardized implementation of adversarial image generation techniques are not comparable to each other, because a good result may indicate a robust CNN or it may merely indicate a weak implementation of the adversarial image generation procedure.

4. Experimental Results

In this section, we evaluate our proposed method and compare the results of the two techniques for generating the adversarial images. First, we create a reference dataset specially designed to exploit the traces left by the operations of the acquisition pipeline of different image capturing devices. Then, we train an advanced deep learning architecture to have a baseline to compare the accuracy results in the presence of adversarial images. Finally, we generate several adversarial image examples to demonstrate the performance of our proposed method.

4.1. Experimental Setup

As part of DARPA's MediFor Program, PAR Government Systems collected an initial dataset of 1611 images acquired by 10 different camera models, ranging from DSLRs to phone cameras, with a mixture of indoor and outdoor flat-field scenes. We focus on a flat-field image dataset because flat-field images are more difficult to modify without inserting visual distortions due to the absence of texture content.

Throughout the rest of the paper, we refer to this dataset as PRNU-PAR. Using the PRNU-PAR dataset, we create a patch dataset, composed by image patches of 32 × 32 pixels randomly extracted from the original images. Specifically, 500 patches are uniformly sampled from each original image in the PRNU-PAR dataset, which results in a patch dataset that contains 805,500 patches in total. The training, validation and test sets are created following a 70/20/10 split, while we ensure that the patches in each dataset split only contain patches from different images.

Table 1 shows the statistics of the patch dataset. As can be seen, due to the difference in the number of images per camera model class in the PRNU-PAR dataset, our dataset of image patches has an unequal number of patches for each of the camera models.

Camera Model	Training	Validation	Test
AS-One	90000	25500	12500
ES-D5100	37500	10500	5000
MK-Powershot	35000	10000	5000
MK-s860	35500	10000	5000
PAR-1233	71000	20000	10000
PAR-1476	107000	30500	15000
PAR-1477	70000	20000	9500
PAR-A015	40500	11500	5500
PAR-A075	26000	7000	3500
PAR-A106	54000	15500	7500

Table 1. Number of image patches per camera class for each of the different dataset splits.

Figure 3 shows a representative example of the images that are present in the PRNU-PAR dataset next to one of their randomly extracted patches. In this case, both camera models PAR-A075 and PAR-A106 have been used to capture images of a cloudy sky. Other camera models such AS-One or ES-D5100 have taken images of a white screen. All the image scenes that are captured in the PRNU-PAR dataset are mostly flat and bright.

As it has been shown in the literature [7], these largely uniform images are ideal candidates to be used for the extraction of the “fingerprint” (*e.g.* the characteristic PRNU noise of the camera model) left in the image by the camera.

Figure 3. Example of images from the training set of the patch dataset. (Top) Image from camera model PAR-A075 and one of the randomly selected patches associated with it. (Bottom) Image from camera model PAR-A106 and one of the randomly selected patches associated with it.

4.2. CNN Architecture

In order to do a fair evaluation of our counter-forensic method, we use a CNN-based camera model detector that has been trained to achieve state-of-the-art accuracy results in the patch dataset.

CNN architecture designs have tended to explore deeper models. Networks which can be hundreds of layers deep are now commonplace in the literature. This design trend has been motivated by the fact that for many applications such as image classification tasks, an increase in the depth of the CNN architecture translates into higher accuracy performance if sufficient amounts of training data are available.

A first approach to design a CNN architecture may be to simply stack convolutional or fully-connected layers together. This naive strategy works initially, but gains in accuracy performance quickly diminish the deeper this kind of architecture becomes. This phenomenon is due to the way in which conventional CNNs are trained through backpropogation. During the training phase of a CNN, gradient information must be propagated backwards through the network. This gradient information slightly diminishes as it passes through each layer of the neural network. For a CNN with a reduced number of layers, this is not a problem. For an architecture with a large number of layers, the gradient signal essentially becomes noise by the time it reaches the first layer of the network again.

The problem is to design a CNN in which the gradient information can be easily distributed to all the layers without degradation. ResNets and DenseNets are modern CNN architectures that try to address this problem.

A Residual Network [42], or ResNet is a deep CNN which tackles the problem of the vanishing gradient using a straightforward approach. It adds a direct connection at each layer of the CNN. In previous CNN models, the gradient always has to go through the activations of the layers, which modify the gradient information due to the nonlinear activation functions that are commonly used. With this direct connection, the gradient could theoretically skip over all the intermediate layers and be propagated through the network without being disturbed.

A Dense Network [36], or DenseNet generalizes the idea of a direct connection between layers. Instead of only adding a connection from the previous layer to the next, it connects every layer to every other layer. For each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers. The increased number of connections ensures that there is always a direct route for the information backwards through the network. The connectivity pattern of DenseNets yields state-of-the-art accuracies on the CIFAR10 image classification dataset, which is composed by images of 32×32 pixels in size.

Motivated by the accuracy performance of DenseNet in the CIFAR10 dataset and the fact that we also work with image patches of 32×32 pixels, we select a DenseNet model with 40 layers as our CNN camera model detector. To prevent the network from growing too wide and to improve the parameter efficiency, we limit the growth rate of the network, this is, the maximum number of input feature-maps that each layer can produce, to $k = 12$. To train the CNN, we use the Adam optimizer with a learning rate of 0.0001 and a batch size of 512 images. After 5 training epochs, we reach a plateau in the accuracy in our validation set. Table 2 shows the single patch accuracy results for our training, validation and test splits of the patch dataset.

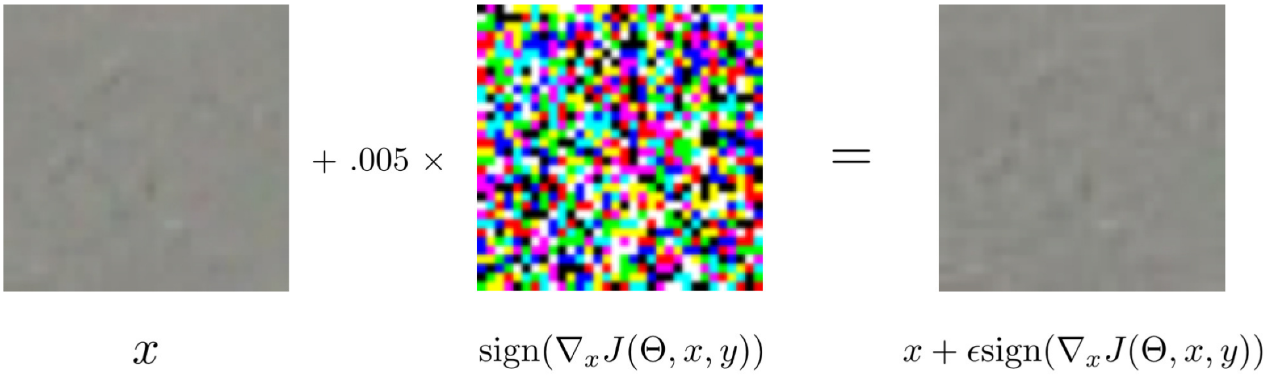
Dataset Split	**Train**	**Validation**	**Test**
Accuracy (%)	99.8	98.7	97.7

Table 2. Single patch accuracy results for our training, validation and test splits of the patch dataset.

4.3. Adversarial Image Generation

In order to evaluate the performance of our counter-forensic method, we test the DenseNet model trained on the patch dataset using untargeted attacks with FGSM and targeted attacks with JSMA. To properly evaluate our method, we only perturb images from the test split which were correctly classified by our CNN in their original states.

To be clear, what we refer as the average confidence score in this paper is the average value of the probability that is associated with the candidate camera model label for each of the image patches in the test split. The probability


+ .005 ×
=
x
$\text{sign}(\nabla_x J(\Theta, x, y))$
$x + \epsilon\text{sign}(\nabla_x J(\Theta, x, y))$

Detected camera model: AS-One
97.9% confidence

Detected camera model: ES-D5100
7.8% confidence

Detected camera model: PAR-1476
99.4% confidence

Figure 4. An example of untargeted fast adversarial image generation using FGSM applied to our trained DenseNet model on the patch dataset. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change DenseNet's classification of the image patch.

for each candidate camera model label corresponds with the highest probability value assigned by the softmax layer of our trained DenseNet model.

For untargeted attacks with FGSM, we report in Table 3 the error rate and the average confidence score on the test split of the patch dataset for different values of ϵ which have been shown to generate high misclassified adversarial images while not producing appreciable visual changes. We find that using $\epsilon = 0.005$ offers the best compromise between error rate and visual changes in the image, causing the trained DenseNet model detector to have a error rate of 93.1% with an average confidence of 95.3% on the patch test split. It should be noted that as we increase the value of ϵ, the manipulations become more visually apparent.

Figure 4 shows an example of the adversarial images that our proposed method can generate when we use FGSM. The modifications done to the images by FGSM are performed on 32-bit floating point values, which are used for the input of the DenseNet model. The gradient computed for Figure 4 uses 8-bit signed integers. To publish the sign of the gradient image in the paper, we have done a custom conversion from 8-bit signed integers to 8-bit unsigned integers. To increase the range of each color channel, we represent the -1s values as 0 and the 1s as 255. For the possible 0's, we have treated them as positive values (they are represented by 255).

For targeted attacks with JSMA, we report in Table 4 the error rate and the average confidence score for each possible camera model target class. Figure 5 shows an example of the images that JSMA allows us to generate when we perform a targeted attack. In this case, an image patch captured by camera ES-D5100 that is correctly classified when is analyzed by our trained DenseNet model is manipulated to be misclassified as an image patch that had been generated by

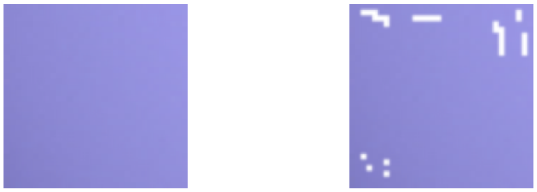
ϵ **value**	**Error rate (%)**	**Confidence Score (%)**
0.001	91.4	97.7
0.002	91.7	97.2
0.003	92.2	96.7
0.004	92.7	95.8
0.005	93.1	95.3
0.006	94.1	95.1
0.007	94.5	94.2
0.008	95.3	93.6
0.009	95.9	93.0
0.01	96.2	92.3

Table 3. Error rate and confidence score values of our trained DenseNet model after an untargeted attack with FGSM to the test split with different values of ϵ.

Target Camera Model	**Error rate (%)**	**Confidence Score (%)**
AS-One	99.5	87.7
ES-D5100	99.3	88.6
MK-Powershot	99.3	88.4
MK-s860	99.7	88.5
PAR-1233	99.7	87.9
PAR-1476	99.4	88.1
PAR-1477	99.5	88.2
PAR-A015	99.6	88.4
PAR-A075	99.3	87.8
PAR-A106	99.2	87.9

Table 4. Error rates and confidence scores of our trained DenseNet model for each possible target camera model after applying a targeted attack with JSMA to the test split.

camera model PAR-1233. It is important to appreciate that although JSMA allows us to generate image patches that get misclassified into a specific camera model with high error rates and confidence scores, the modifications that it applies to the images can usually be spotted through visual inspection. This effect is due to the fact that JSMA crafts the adversarial images by flipping pixels to their minimum or maximum values. Because our patch dataset is composed of image patches with mostly flat scene content, the effect can be clearly observed, for example, in the upper corners of the manipulated image patch in Figure 5.



Detected camera model: ES-D5100
98.1% confidence

Detected camera model: PAR-1233
89.6% confidence

Figure 5. An example of targeted adversarial image generation using JSMA applied to our trained DenseNet model on the patch dataset. (Left) Original image patch correctly classified as ES-D5100. (Right) Altered image patch with target camera model PAR-1233

5. Conclusions

This paper described a counter-forensic method to subtly alter images to change their estimated camera model when they are analyzed by a CNN-based camera model detector. We tested our method on a reference dataset with images from multiple cameras that show highly similar indoor and outdoor scenes. The results demonstrate that we can generate imperceptibly altered adversarial images that are misclassified with high confidence by the CNN. In the future, we will extend our method to apply it to video sequences and we will explore viable adversarial example detection methods and defense techniques to increase the robustness of CNN-based camera model detectors.

6. Acknowledgments

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-16-2-0173. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL or the U.S. Government.

References

[1] K. Liang, J. K. Liu, R. Lu, and D. S. Wong, "Privacy concerns for photo sharing in online social networks," *IEEE Internet Computing*, vol. 19, no. 2, pp. 58–63, March 2015.

[2] H. Farid, "Seeing is not believing," *IEEE Spectrum*, vol. 46, no. 8, pp. 44–51, August 2009.

[3] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, "Vision of the unseen: Current trends and challenges in digital image and video forensics," *ACM Computing Surveys*, vol. 43, no. 4, pp. 26:1–26:42, October 2011.

[4] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, p. 22, January 2013.

[5] M. C. Stamm, M. Wu, and K. J. R. Liu, "Information Forensics: An Overview of the First Decade," *IEEE Access*, vol. 1, pp. 167–200, May 2013.

[6] M. Kharrazi, H. T. Sencar, and N. Memon, "Blind source camera identification," *Proceedings of the IEEE International Conference on Image Processing*, pp. 709–712, October 2004, Singapore.

[7] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," *Proccedings of the IEEE International Conference on Image Processing*, pp. 1296–1299, October 2008, San Diego, CA.

[8] M. Kirchner and T. Gloe, "Forensic camera model identification," *Handbook of Digital Forensics of Multimedia Data and Devices*. Chichester, UK: John Wiley & Sons, Ltd, 2015, pp. 329–374.

[9] P. J. Chiang, N. Khanna, A. K. Mikkilineni, M. V. O. Segovia, S. Suh, J. P. Allebach, G. T. C. Chiu, and E. J. Delp, "Printer and scanner forensics," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 72–83, March 2009.

[10] N. Khanna, A. K. Mikkilineni, A. F. Martone, G. N. Ali, G. T.-C. Chiu, J. P. Allebach, and E. J. Delp, "A survey of forensic characterization methods for physical devices," *Digital Investigation*, vol. 3, pp. 17–28, 2006.

[11] A. Swaminathan, M. Wu, and K. J. R. Liu, "Digital image forensics via intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 101–117, March 2008.

[12] K. Choi, E. Lam, and K. Wong, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express*, vol. 14, no. 24, pp. 11 551–11 565, November 2006.

[13] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," *Proceedings of the IEEE International Conference on Image Processing*, pp. III–69–72, September 2005, Genova, Italy.

[14] H. Cao and A. C. Kot, "Accurate detection of demosaicing regularity for digital image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 899–910, December 2009.

[15] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Demosaicing strategy identification via eigenalgorithms," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2659–2663, May 2014, Florence, Italy.

[16] N. Khanna, A. K. Mikkilineni, and E. J. Delp, "Forensic camera classification: Verification of sensor pattern noise approach," *Forensic Science Communications*, vol. 11, no. 1, January 2009.

[17] ——, "Scanner identification using feature-based processing and analysis," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 123–139, March 2009.

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, June 2014, Columbus, OH.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[20] D. Mas Montserrat, Q. Lin, J. Allebach, and E. J. Delp, "Training object detection and recognition CNN models using data augmentation," *Proceedings of the IS&T International Symposium on Electronic Imaging*, January 2017, Burlingame, CA.

[21] M. Buccoli, P. Bestagini, M. Zanoni, A. Sarti, and S. Tubaro, "Unsupervised feature learning for bootleg detection using deep learning architectures," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 131–136, December 2014, Atlanta, GA.

[22] C. Jiansheng, K. Xiangui, L. Ye, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, November 2015.

[23] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pp. 5–10, June 2016, Vigo, Spain.

[24] G. Xu, H. Z. Wu, and Y. Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, May 2016.

[25] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 1–6, December 2016, Abu Dhabi, United Arab Emirates.

[26] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "First steps toward camera model identification with convolutional neural networks," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259–263, March 2017.

[27] L. Bondi, D. Güera, L. Baroffio, P. Bestagini, E. J. Delp, and S. Tubaro, "A preliminary study on convolutional neural networks for camera model identification," *Proceedings of the IS&T International Symposium on Electronic Imaging*, January 2017, Burlingame, CA.

[28] M. Kirchner and R. Böhme, "Tamper hiding: Defeating image forensics," *Proceedings of the International Workshop on Information Hiding*, pp. 326–341, June 2007, Saint-Malo, France.

[29] R. Böhme and M. Kirchner, "Counter-forensics: Attacking image forensics," *Digital Image Forensics: There is More to a Picture than Meets the Eye*, H. T. Sencar and N. Memon, Eds. New York, NY: Springer New York, 2013, pp. 327–366.

[30] M. Goljan, J. Fridrich, and M. Chen, "Sensor noise camera identification: Countering counter-forensics," pp. 75 410S–75 410S, January 2010, San Jose, CA.

[31] T. Gloe, M. Kirchner, A. Winkler, and R. Böhme, "Can we trust digital image forensics?" *Proceedings of the ACM International Conference on Multimedia*, pp. 78–86, September 2007, Augsburg, Germany.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *Proceedings of the International Conference on Learning Representations*, April 2014, Banff, Canada.

[33] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *Proceedings of the International Conference on Learning Representations*, May 2015, San Diego, CA.

[34] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 582–597, May 2016, San Jose, CA.

[35] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *Proceedings of the IEEE European Symposium on Security and Privacy*, pp. 372–387, March 2016, Saarbrücken, Germany.

[36] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv:1608.06993*, August 2016.

[37] C.-C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, November 2016.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[39] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, January 2009.

[40] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of the International Conference on Computational Statistics*, pp. 177–186, August 2010, Paris, France.

[41] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1.0.0: an adversarial machine learning library," *arXiv:1610.00768*, October 2016.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, December 2015.