https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-328

Design Principles of Convolutional Neural Networks for Multimedia Forensics

Belhassen Bayar and Matthew C. Stamm; Drexel University, Philadelphia, PA, bb632@drexel.edu, mstamm@coe.drexel.edu

Abstract

Convolutional neural networks (CNNs) have received significant attention due to their ability to adaptively learn classification features directly from data. While CNNs have helped cause dramatic advances in fields such as object and speech recognition, multimedia forensics is fundamentally different problem compared to other deep learning applications. Little work exists to guide the design of CNN architectures for forensic tasks. Furthermore, it is still unclear which forensic tasks can be performed using CNNs. In this work, we investigate the design of CNNs for multiple multimedia forensic applications. We show that CNNs are capable of performing image manipulation detection as well as camera model identification. Through a series of experiments, we systematically examine the influence of several important CNN design choices for forensic applications, such as the use of a constrained convolutional layer or fixed high-pass filter at the beginning of the CNN, the use of nonlinearity after the first layer, the choice of activation and pooling functions, etc. We show that different CNN design choices should be made for different forensic applications and identify design choices to maximize the performance of CNNs for manipulation detection and camera model identification.

Introduction

Multimedia information, such as digital images, is frequently used in numerous important settings, such as evidence in legal proceedings, criminal investigations, and military and defense scenarios. Since this information can easily be edited or falsified, information forensics researchers have developed a wide variety of methods to forensically determine the authenticity and source of multimedia data [27]. Many early forensic approaches were developed by theoretically or heuristically identifying a set of traces left in an image by a particular processing operation or source device. For example, techniques have been developed to detect specific traces left by resampling and resizing [22, 14], contrast enhancement [26], median filtering [15, 13], sharpening [2], and many other operations. Similarly, forensic algorithms have been developed to identify the model of an image's source camera using specific traces left by different elements of the camera's internal processing pipeline [28, 3, 8].

More recent data-driven forensic approaches have leveraged techniques from steganalysis research that capture local pixel dependencies using high dimensional feature sets [9, 20]. These approaches have been used to detect image editing [23] and perform camera model identification [4, 19]. While these techniques have shown significant improvements in manipulation detection or camera model identificaiton accuracy, researchers are still left with questions such as: Are these the best set of classification features for forensic tasks? Can forensic traces and classification features be learned directly from data?

Recently, researchers have shown significant interest in using techniques from deep learning to address problems in multimedia security such as image manipulation detection and steganalysis [1, 21]. Tools such as convolutional neural networks (CNNs) [17] show particular promise due to their ability to adaptively learn decision features from large sets of data. While CNNs have been successfully applied to computer vision problems such as object recognition [16, 29], lessons learned from this field do not necessarily translate to multimedia forensics. Because multimedia forensics is a fundamentally different problem, certain design principals that are successful when building CNNs to perform object recognition are suboptimal for multimedia forensic tasks.

In their standard form, CNNs tend to learn features related to the image's content, whereas, in image forensics tasks we need to suppress an image's content and capture pixel value dependencies finduced by an editing operation (e.g., image tampering detection task [23]) or the camera's image processing pipeline (e.g., camera model identification task using models of camera's demosaicing algorithm [28]). If CNNs are used in their existing form, this will lead to a classifier that identifies the objects and scenes associated with the camera as opposed to learn image forensics classification features. In response to this problem, two methods have emerged namely the predetermined high-pass filter based deep learning approach used in steganography [21] and the 'constrained convolutional' layer adaptive approach used to perform image manipulation detection [1].

Since the use of deep learning approaches for multimedia security applications is still in its infancy, little work exists to guide the design of CNN architectures for forensic tasks. In addition, no work has explicitly examined and compared the performance of different proposed network topologies. As a result, several open questions currently exist with regard to the design and training of CNNs for multimedia forensics. For example: Which problems in multimedia forensics can CNNs be successfully applied to? Multiple approaches have recently been proposed for the design of the initial CNN layer (i.e. high-pass filter or constrained convolutional layer). Which of these yields the best performance? Do certain other design parameters such as the pooling technique or choice of activation function have significant effects on the CNN's accuracy? Do these design choices vary depending on the forensic task being considered? What effect do different training techniques such as batch normalization and local contrast normalization have on the CNN's classification accuracy? In order to guide future research into the application of deep learning techniques for multimedia security, it is important to address these questions.

In this paper, we systematically investigate several CNN design choices, then use the results of our investigation to present

several guidelines for designing and training CNN architectures for multimedia forensics. Specifically, we investigate (1) the impact of the choice of the initial CNN layer, (2) the effect of different types of nonlinearity following the first layer (i.e. pooling, the absolute value function, nonlinear activation functions, etc.), (3) the performance of different pooling techniques, (4) the influence of network depth, (5) the effect of integrating a 1×1 layer into the CNN to learn associations across feature maps, (6) the influence of the choice of activation functions, and (7) the effect of different training approaches such as batch normalization and local contrast normalization.

Additionally, we demonstrate that CNNs can be designed to perform several different forensic tasks. Specifically, we investigate the design of CNNs for detecting multiple different image manipulations as well as for identifying the model of an image's source camera. The construction of a CNN for performing camera model identification is a new result that, to the best of our knowledge, has not yet been published. Our investigation reveals both general CNN design principles that are important regardless of the forensic task, along with other design choices that must be appropriately selected depending on the chosen forensic task.

Background

Convolutional neural networks (CNNs) are an extended version of neural networks (NNs) [17]. They have proven effective at extracting classification features directly from data with different types of signals such as images [16], text data [6] and speech [11], etc. A CNN's architecture, which is the set of parameters and components that we need to design a network, is based on stacking many hidden layers on top of one another which makes CNNs capable of learning hierarchical features. That is, they can learn higher-level features from a set of previously learned lower-level features.

In a CNN architecture the first convolutional layer operates as a features extractor which consists of a set of convolutional filters with a fixed size. These filters convolve in parallel with all regions of an input image with an overlapping distance called a stride. The output of each convolutional filter in a convolutional layer is a new learned representation of data known as feature map. Every entry in a feature map is equivalent to one neuron in a NN's layer. Subsequently, the following hidden convolutional layers similarly extract features from the input feature maps previously learned by a former convolutional layer. The output of these hierarchical feature extractors is fed to a fully-connected neural network that performs a classification task.

The analytical expression of the convolution within the CNN architecture is given in Eq. (1):

$$h_j^{(n)} = \sum_{k=1}^{K} h_k^{(n-1)} * w_{kj}^{(n)} + bj^{(n)}, \tag{1}$$

where $*$ denotes a 2-D convolution operation, $h_j^{(n)}$ is the j^{th} feature map output in the n^{th} hidden layer, $h_k^{(n-1)}$ is the k^{th} channel in the $(n-1)^{th}$ hidden layer, $w_{kj}^{(n)}$ is the k^{th} channel in the j^{th} filter in the n^{th} layer and $b_j^{(n)}$ is its corresponding bias term.

Recently, CNNs have been successful for addressing problems such as object recognition [16, 29]. These recent advances have been improved with the use of GPUs to overcome the computational expense of learning the large number of filters coefficients and fully-connected layers weights that a deep network involves. These weights and coefficients are initially seeded with random values, then learned through an algorithm called back-propagation. To introduce nonlinearity throughout the network, the convolutional and fully-connected layers are typically followed by an activation function. This enables each artificial neuron to act as a human neuron that suppresses the small values of a layer's output and activates the large ones. Furthermore, the set of hierarchical convolutional layers yields a large volume of feature maps which makes the CNN computationally very expensive. To address this issue, the convolutional layers are followed by another type of layer called pooling which is used to reduce the dimension of each feature map. This reduces the computational cost associated with training the network and decreases the chances of over-fitting by retaining the most representative features throughout CNN. There exist many types of pooling operations such as max, average and stochastic pooling. A max-pooling layer for instance, operates as a sliding window with a stride distance that retains the maximum value within the sliding window dimension.

The coefficients of the convolutional filters w_{ij}'s are automatically learned using an iterative algorithm which alternates between feedforward and back-propagation passes of the data. This iterative process will lead to minimize the average loss between the actual labels and the network outputs, i.e., $E = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_i^{*(k)} \log\left(y_i^{(k)}\right)$, where $y_i^{*(k)}$ and $y_i^{(k)}$ are respectively the true label and the network output of the i^{th} image at the k^{th} class with m training images and c neurons in the output layer. In fact, numerous solvers [24, 30, 7] could be used to solve the underlying optimization problem. In this work we consider the stochastic gradient descent (SGD) to train the CNN models [24]. The iterative update rule for the filters coefficients $w_{ij}^{(n)}$ in CNN during the back-propagation pass is given below:

$$\begin{aligned} \nabla w_{ij}^{(n+1)} &= m \cdot \nabla w_{ij}^{(n)} - d \cdot \varepsilon \cdot w_{ij}^{(n)} - \varepsilon \cdot \frac{\partial E}{\partial w_{ij}^{(n)}} \\ w_{ij}^{(n+1)} &= w_{ij}^{(n)} + \nabla w_{ij}^{(n+1)}, \end{aligned} \tag{2}$$

where $w_{ij}^{(n)}$ represents the i^{th} channel from the j^{th} kernel matrix in the hidden layer $h^{(n)}$that convolves with the i^{th} channel in the previous feature maps denoted by $h_i^{(n-1)}$, $\nabla w_{ij}^{(n)}$ denotes the gradient of $w_{ij}^{(n)}$ and ε is the learning rate. The letters m and d are respectively the momentum and the decay. The bias term $b_j^{(n)}$ in (1) is updated using the same equations presented in (2). The use of the *decay* and *momentum* strategy is mainly for fast convergence as explained by LeCun *et al.* in [18].

General experimental setup

When a forensic investigator designs a CNN architecture, they have at their disposal numerous possible choices of parameters, e.g., the number of convolutional layers, the choice of activation function, etc. Therefore, there exist a large number of design choices that one could make to build a CNN architecture where each element of the network has an impact on CNN's performance. In this work, we present through a set of experiments

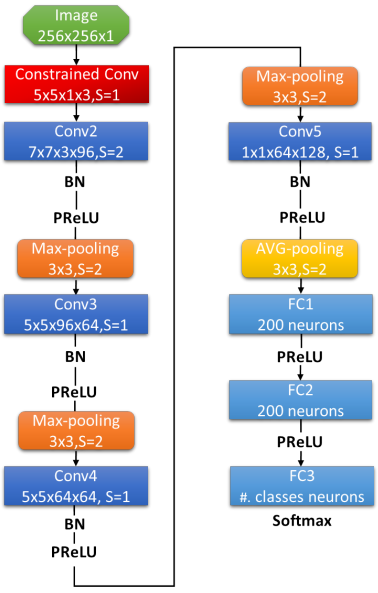

Figure 1. *Our proposed CNN architecture; BN:Batch-Normalization Layer; PReLU: Parametric Rectified Linear Unit Layer.*

the primary steps to design a CNN model capable of adaptively learning image forensics features directly from data. We first establish a baseline architecture shown in Fig. 1. From this figure, we can see that the baseline architecture consists of one constrained convolutional layer which is followed directly by four convolutional layers and three fully-connected layers. All the regular convolutional layers are followed by a batch normalization layer (BN) [12], a parametric rectified linear unit (PReLU) activation function [10] and a max-pooling layer.

One can notice that the constrained convolutional layer is not followed by any type of nonlinearity (e.g., pooling layer and/or an activation function) which will be explained through our experiments. Additionally, we can notice that in the "Conv5" layer we use 1×1 filters followed by an average pooling layer. We use these type of filters to learn new associations between feature maps. The first two fully connected layers contain 200 neurons and are followed by a PReLU activation function. The number of neurons in the output layer is equal to the number of classes in the considered forensics task. This layer is followed by a softmax activation function and the class of the input image corresponds to the highest activated neuron in the output layer. In this work, the input layer of our CNN is the green layer of an image patch sized 256×256 pixels.

In our experiments, we vary specific design choices in the baseline architecture and experimentally evaluate the performance of these choices on two different tasks, namely (1) image manipulation detection and (2) camera model identification. Only one design choice is varied at a time in order to understand the importance of that particular design choice. In every experiment, we train each CNN for 45 epochs, which is the number of times that every sample in the training data was trained. Additionally, while training CNNs, their testing accuracies on a separate testing database were recorded every 1000 iterations to produce figures in the experiments sections. In what follows, we give details about the different architectural design choices that we will test in our experiments.

We start our experiments by studying the first convolutional layer in a CNN architecture. This layer is the most important in a CNN's architecture since it extracts the lowest-level features from the input data. Then higher-level features will be extracted through the following hidden convolutional layers. As we mentioned above, CNNs in their existing form do not suppress the image's content and may lead to a classifier that identifies the scenes and objects associated with training data. To combat this issue, two alternatives have emerged to suppress an image's content and capture pixel values dependencies when CNN is used in a forensics task, i.e., high-pass filter (HPF) [21] and constrained convolutional layer [1]. More specifically, the following fixed HPF has been used in steganography [21]:

$$w^{(1)} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix}$$

as a first convolutional layer in CNN. This method has proven effective and promising at distriguishing between the stego and cover images.

However, this approach still suffers from the fact that a human intervention is required to choose a predetermined filter which is not adaptive. To address this problem, we have suggested in our recent work the constrained convolutional layer [1] which can jointly suppress an image's content and adaptively learn pixel values dependencies while training CNN to perform universal image manipulation detection. Each of the K filters $w_k^{(1)}$ in the first convolutional layer of the CNN have the following constraints placed on them:

$$\begin{cases} w_k^{(1)}(0,0) = -1 \\ \sum_{\ell,m\neq 0} w_k^{(1)}(\ell,m) = 1 \end{cases} \tag{3}$$

In this manner, we enforce the first convolutional layer in CNN to extract prediction-error features by learning a normalized linear combination of the central pixel value in terms of its local neighbors within the convolutional filter dimension. Pseudocode outlining the training process of the constrained convolutional layer is given in Algorithm 1. In this work we compare the performance of a CNN when both types of the first convolutional layer are used to extract the lowest-level forensics features from data, i.e., prediction-error features. Furthermore, when the "Constrained Conv" layer in Fig. 1 is replaced by a HPF layer, the new dimension of "Conv2" layer becomes $7 \times 7 \times 1 \times 96$.

In different deep learning applications, the first convolutional layer is typically followed by an activation function and a pooling layer. However, such rules may not apply to multimedia forensics tasks. Thus, in the second set of experiments we study the effect of introducing different nonlinear operations to the prediction-error features learned by the "Constrained Conv" layer used in the baseline architecture. That is, we would like to know if nonlinearity can help to learn a better representation of the data when introduced to the prediction-error features.

From Fig. 1, one can observe that we use an average-pooling layer after the 1×1 filters in the "Conv5" layer. Choosing the type of the pooling layer after the 1×1 convolutional filters is a

Algorithm 1 Training algorithm for constrained convolutional layer

Initilize w_k's using randomly drawn weights
i=1
while $i \leq max\_iter$ **do**
    Do feedforward pass
    Update filter weights through stochastic gradient descent and backpropagate errors
    Set $w_k(0,0)^{(1)} = 0$ *for all K filters*
    Normalize $w_k^{(1)}$*'s such that* $\sum_{\ell,m\neq 0} w_k^{(1)}(\ell,m) = 1$
    Set $w_k(0,0)^{(1)} = -1$ *for all K filters*
    i = i+1
    if training accuracy converges **then**
        exit
end

very critical task and has a noticeable impact on the performance of the CNN. These 1×1 filters learn the association between the highest-level feature maps in the network before the fully-connected layers perform classification. Therefore, it is important to choose a pooling layer that keeps the most representative features which will be fed to the fully-connected layers. Moreover, there exist other types of pooling that may perform better than average-pooling. Therefore, in the experimental part we study the impact of using a max-pooling layer after "Conv5".

One can also notice from Fig. 1 that in the baseline architecture we use the PReLU activation function [10]. When CNNs used with a PReLU activation function, it has experimentally demonstrated to surpass human-level performance on visual recognition challenge [25]. Earlier CNN architectures in computer vision [16, 29] use the ReLU activation function which have shown several advantages during training the network. Recently, Clever *et al.* [5], proposed the exponential linear units (ELU) activation function, which considerably speeds up learning and obtains less than 10% classification error compared to a ReLU network with the same architecture. Thus, we experimentally study the effect of using each of the above mentioned activation functions in the baseline architecture.

There is no systematic way of determining the necessary depth in a CNN architecture and one has to empirically choose the appropriate number of convolutional layers in the network. Therefore, in the experimental part we assess the performance of CNN when trained with different depths. We also study the impact of using 1×1 convolutional filters. Unlike the convolutional layers with larger filter size where each is followed by a max-pooling layer, we use an average-pooling layer after the 1×1 convolutional filters. We start by training a CNN architecture with one convolutional layer after the "Constrained Conv" layer, then we vary the network depth by increasing the number of convolutional layers while keeping the number of the fully-connected layers fixed. Thus, for every fixed depth we train a CNN both with and without a 1×1 convolutional layer whose number of filters is equal to 128. Additionally, every non-constrained convolutional operation is followed by a BN layer, a PReLU activation function and a pooling layer.

Similarly to the activation function, researchers in computer vision have developed several techniques to normalize the data throughout the CNN architecture. Early CNNs use the local response normalization (LRN) layer to normalize the data throughout the network. This type of layer normalizes the central coefficient within a sliding window in a feature map with respect to its neighbors. Furthermore, the LRN layer has proven to improve the testing performance of the network. In particular, it has improved the AlexNet model [16] testing accuracy by 1.4% to classify $1,000$ different objects.

Recently Ioffe *et al.*, proposed in [12] the batch normalization layer which dramatically accelerates the training of deep networks. This type of mechanism minimizes the internal covariate shift which is the change in the input distribution to a learning system. This is done by a zero-mean and unit-variance transformation of the data while training the CNN model. Since the input to each layer is affected by the parameters of all previous layers, even small changes get amplified. Thus, this type of layer addresses an important problem and increases the final accuracy of a CNN model. Additionally, the BN layer has proven to reduce the effect of overfitting in CNNs. Therefore, in our experiments we trained the baseline architecture with these two choices of normalization layer to assess the impact of each on CNN's performance.

Image manipulation detection

In this part, we present an experimental study about the impact of the different design choices of the CNN parameters on the final image manipulation detection rate. We used the baseline CNN architecture in Fig. 1 to adaptively extract image manipulation features directly from data and perform universal image tampering detection with five different editing operations listed in Table 1. For this forensics task, the output layer of CNN consists of 6 neurons, i.e. original versus five editing operations. As mentioned above, the baseline architecture of CNN is fixed then changed accordingly to each tested design setting described in the previous section.

Data collection & training parameters To run this set of experiments, we first built an experimental database that we downloaded from the 1st IEEE IFS-TC image forensics challenge website: `http://ifc.recod.ic.unicamp.br/fc.website/index.py?sec=5`. We collected $2,445$ images of size 1024×768 for the training and testing data. To train our CNNs we randomly selected $1,852$ images from our experimental database. Next, we divided these images into 256×256 pixel patches and retained all the nine central patches from the green layer of each image for our training database. We then created their corresponding edited patches using the five tampering operations listed in Table 1. In total, our training database consisted of $100,000$ patches in which $16,667$ patches were not tampered.

When training our CNN, we set the batch size equal to 64 and the parameters of the stochastic gradient descent as follows: $momentum = 0.9$, $decay = 0.0005$, and a learning rate $\varepsilon = 10^{-3}$ that decreases every 5 epochs by a factor $\gamma = 0.5$. We trained the CNN in each experiment for 45 epochs (approximately $70,000$ iterations). We subsequently created a testing database by dividing the remaining 593 images not used for the training into 256×256 pixel patches in the same manner described above. In total, our testing database consisted of $32,000$ patches where $5,337$ patches were not edited. Note that training and testing are disjoint. In what follows we describe the set of experiments that we ran to

Table 1: Used editing parameters to create our experimental database for CNN-based universal manipulation detection.

Editing operation	Parameter
Median Filtering (MF)	$K_{size} = 5 \times 5$
Gaussian Blurring (GB) with $\sigma = 1.1$	$K_{size} = 5 \times 5$
Additive White Gaussian Noise (AWGN)	$\sigma = 2$
Resampling (RS) using bilinear interpolation	Scaling $= 1.5$
JPEG compression	$QF = 70$

find the appropriate architecture to image manipulation detection.

Experimental results In the first set of experiments, we compared a CNN with the constrained convolutional layer [1] to the steganalysis approach suggested in [21] where the input image is first convolved with a predetermined HPF. Fig. 2 depicts the curves of the image manipulation detection rate versus the number of training epochs for both networks, i.e., the HPF-based CNN and constrained convolutional layer-based CNN. One can observe that the CNN with the constrained convolutional layer outperforms the HPF approach throughout the training epochs where the best achieved detection rates are respectively 98.70% and 97.99%. The HPF approach achieves a lower detection rate since it is a suboptimal solution of the trained network with a constrained convolutional layer. Thus, the constrained convolutional layer can capture image manipulation features that may not be captured using a hand-designed HPF.

In the baseline CNN architecture, the prediction-error feature maps adaptively learned by the constrained convolutional layer are directly convolved with the regular convolutional layer "Conv2". Therefore, in this part of experiments we study the effect of introducing nonlinearity after the "Constrained Conv" layer and whether this could lead to learn better representative image manipulation features. Normally, all the regular convolutional layers are followed by an activation function and a pooling layer. Thus, we assessed the performance of the baseline architecture in Fig. 1 after applying the changes listed in Table 2. We trained the new CNN architectures with the same training parameters used in the previous experiments.

We first trained our CNN architecture with a PReLU activation function and a max-pooling layer after the "Constrained Conv" layer. The best achieved detection rate with this new architecture is 95.48%, whereas, the baseline architecture achieved 98.70% without introducing the nonlinearity to the prediction-error features. Therefore, by adding an activation function and a pooling layer after the "Constrained Conv" layer, the CNN model has lost the salient features capable of detecting image manipulation with a higher rate.

Then, we assessed the CNN baseline architecture when adding only a max-pooling layer after the constrained convolutional layer. The best achieved detection rate with this architecture is 93.19%. We then study the impact of adding a symmetry operation to the prediction-error features extracted by the first constrained layer. Therefore, we trained our model in Fig. 1 with adding an absolute value layer after the "Constrained Conv" layer. The final detection rate significantly decreased to 94.90%. These results suggest that any type of nonlinearity introduced to

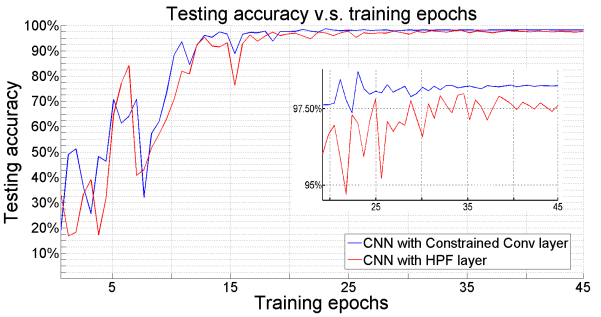

Figure 2. *CNN testing accuracy v.s. training iterations for image manipulation detection, blue: our proposed architecture, red: HPF-based CNN.*

Table 2: CNN testing accuracy with nonlinear operations after constrained convolutional layer for image manipulation detection.

Nonlinear operations	Accuracy
PReLU + Max-pooling	95.48%
Max-pooling	93.19%
Absolute value	94.90%
w/out nonlinearity	**98.70%**

the prediction-error features inhibits important representative features and drops significantly the overall detection rate. Therefore, it is very important to use a regular convolutional layer directly after the constrained convolutional operation.

The baseline network architecture includes one constrained convolutional layer, three regular convolutional layers followed by a 1×1 convolutional layer that learns the association across feature maps and three fully connected layers used for the classification task. However, the number of convolutional layers cannot be determined in a systematic way and one has to try different depths of the network to find the appropriate number of convolutional layers after the constrained one. Furthermore, we need to know how important is to learn association across the feature maps and whether the 1×1 convolutional filters can improve the final detection rate.

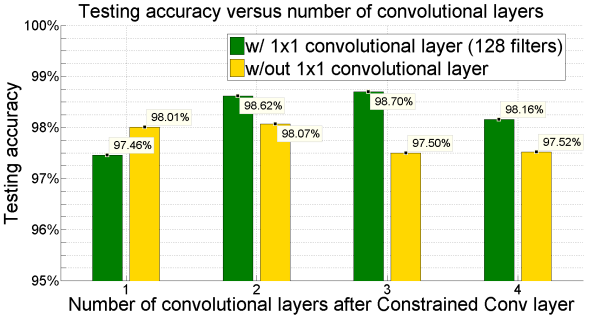

Figure 3. *Testing accuracy versus number of convolutional layers for image manipulation detection, green: last convolutional layer followed by* 1×1 *convolutional layer, yellow: last convolutional layer not followed by* 1×1 *convolutional layer.*

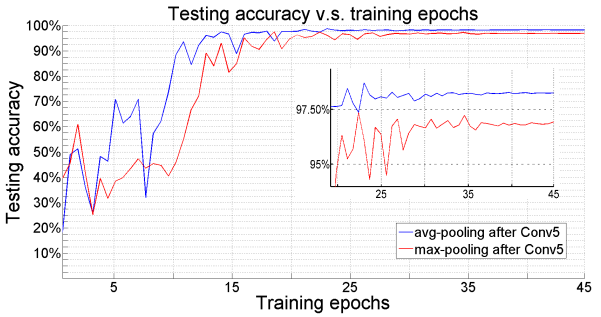

Figure 4. *CNN testing accuracy v.s. training iterations for image manipulation detection, blue: avg-pooling after Conv5, red: max-pooling after Conv5.*

Table 3: CNN testing accuracy with different activation functions for image manipulation detection.

Activation function	Accuracy
ELU	98.52%
ReLU	97.78%
PReLU	**98.70%**

Fig. 3 summarizes the performance of each possible depth. One can notice that as we increase the number of convolutional layers we can improve the overall detection rate until we reach the optimal depth. We can also notice that with two, three and four non-constrained convolutional layers, the 1×1 convolutional filters have importantly improved the final detection rate. However, if we use these 1×1 filters after the first non-constrained convolutional layer the detection rate decreases from 98.01% to 97.46%.

One can observe that the best performance is achieved when we use the baseline architecture in Fig. 1, i.e., three non-constrained convolutional layers followed by 1×1 convolutional filters after the "Constrained Conv" layer. Additionally, we can notice that when four convolutional layers are used after the "Constrained Conv" layer, where the number of used filters in the fourth added non-constrained convolutional layer is 48 with dimension $5 \times 5 \times 64$, the final detection rate has decreased by 0.54% compared to the baseline CNN architecture in Fig. 1.

In the previous set of experiments, we showed that the 1×1 convolutional filters followed by an average-pooling improves the overall accuracy of the proposed CNN. However, there exist other types of pooling that may perform better. Therefore, we trained the baseline architecture in Fig. 1 using a max-pooling layer after "Conv5" layer. Fig. 4 depicts the detection rate versus the training epochs using two choices of pooling layer. The overall detection rate has decreased from 98.70% to 97.45% when using the max-pooling layer after "Conv5". Thus, for image manipulation detection task, the average-pooling layer retains the most representative features from the deepest convolutional feature maps in the network.

Subsequently, we examined the impact of different choice of activation function on CNN's performance. We compared the performance of the proposed PReLU-based network in Fig. 1 to ELU and ReLU networks. To accomplish this, we trained the baseline architecture with the different mentioned above activation functions. We report the best achieved detection rates of these networks in Table 3. Fig. 5 shows the detection rate versus the training epochs curve for each choice of activation function. One can observe that the PReLU network outperforms the ELU and ReLU networks and reaches a higher constant detection rate in fewer number of epochs. We can notice that the PReLU network performance is respectively 0.92% better than the ReLU network and 0.18% better than the ELU networks.

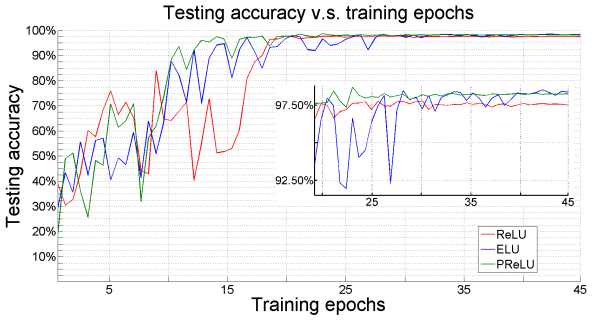

Figure 5. *CNN testing accuracy v.s. training epochs with different activation functions for image manipulation detection.*

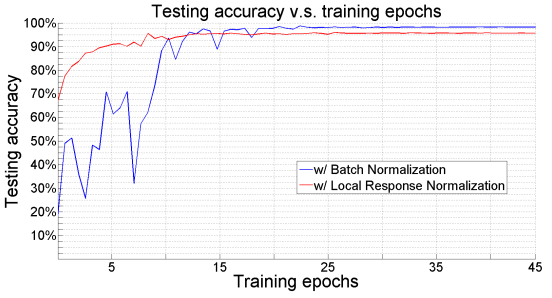

Figure 6. *CNN testing accuracy v.s. training epochs with batch normalization and local response normalization.*

Finally, we compared the proposed BN-based architecture in Fig. 1 to the same architecture that instead uses 5×5 LRN layer after each pooling operation. We trained both architectures for 45 epochs. We used a learning rate $\varepsilon = 10^{-4}$ for the LRN-based model since a higher choice of the learning rate yields very large filters coefficients, hence the training diverges. Fig. 6 depicts the detection rate versus the training epochs of each network where the best achieved rate with the LRN-based network is 95.92% whereas we could achieve 98.70% with the baseline architecture that uses the batch normalization layer. Fig. 7 shows the training loss versus the training epochs for both networks. One can observe that the BN layer leads to significantly lower loss compared to the LRN layer. Additionally, the BN-based CNN loss curve has less spikes throughout the training epochs which demonstrates the impact of this layer on reducing the effect of overfitting.

Summary In this part, we demonstrated through our experiments that a CNN architecture used in different deep learning applications may not be generalized and applied in image manipulation detection task. In fact, we have seen that the choice of the first convolutional layer is very crucial to CNN's performance. More specifically, the constrained convolutional layer has proven to help improving the detection rate compared to when CNN is

Table 4: Classification accuracy for each camera model in our database using a ReLU-based CNN architecture in Fig. 1.

Camera Model	Accuracy	Camera Model	Accuracy	Camera Model	Accuracy
Canon EOS SL1	98.97%	Kodak Easyshare C813	99.66%	Samsung S2	99.37%
Canon PC 1234	98.50%	LG G2	98.36%	Samsung S3	93.73%
Canon PC 1730	96.36%	LG G3	99.17%	Samsung S4	99.79%
Canon Powershot G10	97.65%	LG Nexus5	94.63%	Samsung S5	97.63%
Canon Powershot S100	98.94%	LG Realm	97.93%	Samsung S6 edge	99.40%
Canon Powershot A580	98.41%	Motorola Droid Maxx	98.44%	Sony A6000	95.92%
iPad Air 2	97.56%	Motorola Droid Turbo	94.91%	Sony Cybershot DSCT70	98.85%
iPhone 4	98.95%	Nikon D7100	93.18%	Sony Nex 5TL	78.14%
iPhone 4s	99.41%	Nokia Lumia 920	98.71%	Sony Nex 7	93.51%
iPhone 5	98.70%	Samsung Lite	99.53%	Blackberry Leap	95.65%
iPhone 5s	98.33%	Samsung Note3	98.04%	**Overall Accuracy**	**97.59%**
iPhone 6s	97.84%	Samsung Note4	97.55%	-	

used with a predetermined HPF. Additionally, unlike regular CNN architectures our experiments showed that features learned by the first convolutional layer should not be processed by any type of nonlinear operation. More explicitly, the prediction-error features which are the lowest-level image manipulation features are vulnerable to be destroyed by any type of nonlinearity. Furthermore, we found that the image manipulation detection rate could be improved when using deeper CNN architecture, however, it starts to decrease after we reach the appropriate number of convolutional layers. Moreover, when we use 1×1 convolutional filters after the highest-level feature maps the detection rate has also significantly increased for the different tested depths. We also have seen that the particular choice of the pooling layer and the activation function has an impact on CNN's performance. Finally, through our experiments we demonstrated the advantage of using BN layers in CNN, which reduces the effect of overfitting, compared to an architecture that instead uses LRN layers to normalize the data.

Camera model identification

While previous research has shown that CNNs can be used to perform image manipulation detection, researchers may naturally ask: What other forensic tasks can CNNs be used for? Here, we show that CNNs can be designed to determine the make and model of an image's source camera. To the best of our knowledge, this is the first paper to show that CNNs can be built to learn traces left in an image by its source camera, then use these traces to determine the source camera's make and model.

Our experimental results for image manipulation detection showed that one has to test each design choice of CNN to find the appropriate architecture for the considered forensics task. Thus, we are also interested in studying whether rules learned in the previous forensics task could be applied to camera model identification. Similarly to the image manipulation detection experiments, we train different CNN architectures to study the effect of each element in CNN on the CNN's performance to perform image's source identification. The number of neurons in the output layer of the CNN baseline architecture in Fig. 1 is equal to the number of camera models used to collect the training and testing databases.

Data collection & training parameters We built an experimental database by manually capturing images using the 34 different camera models listed in Table 4. At least 300 images were captured by each camera using its default settings to create a set of 17,641 images. To train our CNN, we randomly selected 14,113 images from our experimental database. Next, we divided these images into 256×256 pixel patches and retained the central 25 patches from the green layer of each image for our training database. In this database, each block corresponds to a new image that has its corresponding camera model label. In total, our training database consisted of 352,825 patches.

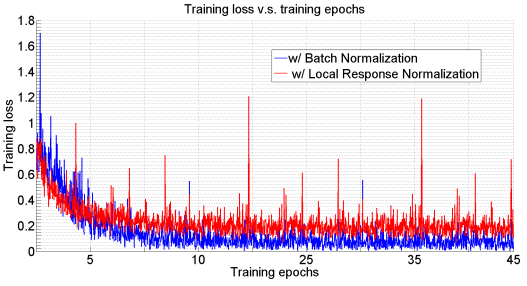

Figure 7. *CNN training loss v.s. training epochs with batch normalization and local response normalization functions.*

When training our CNN, we set the batch size equal to 64 and the parameters of the stochastic gradient descent as follows: $momentum = 0.9$, $decay = 0.0005$, and a learning rate $\varepsilon = 10^{-3}$ that decreases every 5 epochs by a factor $\gamma = 0.5$. We trained the CNN in each experiment for 45 epochs (approximately 250,000 iterations).

Finally, to evaluate the performance of our proposed approach, we created a testing database by dividing the 3,528 images not used for the training into 256×256 pixel patches in the same manner described above. In total, our testing database consisted of 88,200 patches. We then used our CNN to identify the source camera model of each image in the testing set.

Experimental results Similarly to the image manipulation experiments, we compared the constrained CNN in Fig. 1 to the approach suggested in [21] where the input image is first convolved with a HPF layer. Fig. 8, depicts the curve of the identification rate versus the training epochs for each network. One can observe that the constrained convolutional layer based CNN out-

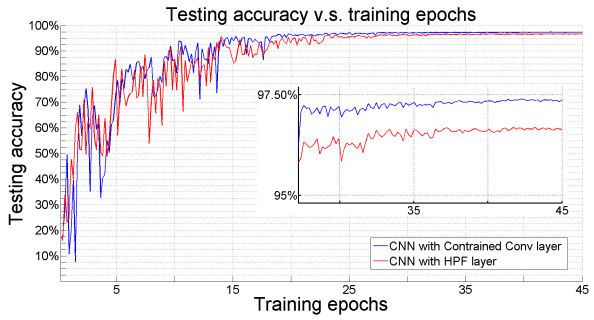

Figure 8. *CNN testing accuracy v.s. training epochs for camera model identification, blue: our proposed architecture, red: HPF-based CNN.*

Table 5: CNN testing accuracy with nonlinear operations after constrained convolutional layer for camera model identification.

Nonlinear operations	Accuracy
PReLU + Max-pooling	70.56%
Max-pooling	79.49%
Absolute value	94.83%
w/out nonlinearity	**97.39%**

performs the HPF approach where the best achieved identification rates are respectively 97.39% and 96.69%. As mentioned in the previous set of experiments, we also believe that the HPF-based CNN achieves lower performance because any predetermined filter is a suboptimal solution of the learned constrained convolutional filters. Additionally, these results show that if we make use of the constrained convolutional layer, a CNN can adaptively extract features related to pixel values dependencies induced by the camera's image processing pipeline, e.g., its demosaicing algorithm, compression, denoising, etc. However, we believe that the camera's fingerprint is dominated by the demosaicing algorithm.

Subsequently, we study the impact of using different nonlinear operations after the "Constrained Conv" layer on the CNN's identification rate for camera model identification. Thus, similarly to the image manipulation detection task where we experimentally showed that the prediction-error features are vulnerable to be destroyed by nonlinear operations, we would like to assess the baseline architecture's ability to perform image source camera model identification when nonlinearity is introduced to the camera's demosaicing features learned by the "Constrained Conv" layer.

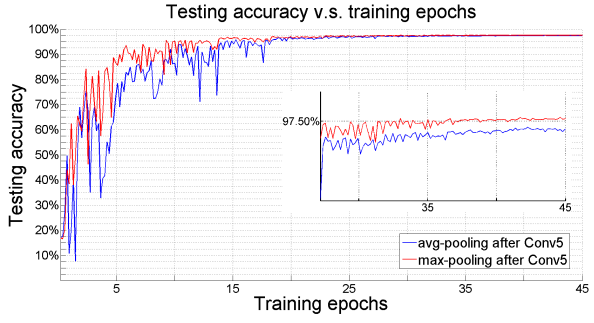

Figure 9. *CNN testing accuracy v.s. training iterations for camera model identification, blue: avg-pooling after Conv5, red: max-pooling after Conv5.*

Table 6: CNN testing accuracy with different activation functions for camera model identification.

Activation function	Accuracy
ELU	97.28%
ReLU	**97.59%**
PReLU	97.39%

Table 5 summarizes the performance of the baseline CNN after applying the nonlinear operations to the prediction-error features. First, we trained the network in Fig. 1 with a PReLU and max-pooling layer after "Constrained Conv" layer. The CNN identification rate has significantly decreased and the best identification rate of the network during the training is 70.56%. We repeated the same experiment by adding only a max-pooling layer. The final identification rate has also decreased compared to the baseline architecture and it didn't achieve better than 79.49% during training. Finally, we study the impact of using an absolute value layer on the learned prediction-error features. The network performance has also decreased and the best achieved identification rate is 94.83%. From Tables 2 and 5 one can notice that the prediction-error features learned in image's source identification task are more vulnerable to be destroyed by nonlinear operations compared to the image manipulation detection task. Thus, through our experiments we were able to demonstrate that for both considered forensics tasks, if the "Constrained Conv" layer is followed by a nonlinear operation then most of the representative features will be destroyed and the performance of the network significantly decreases.

We then compared the performance of the baseline architecture in Fig. 1 to the same architecture that instead uses a max-pooling layer after the 1×1 convolutional filters in "Conv5" layer. Fig. 9 depicts the identification rate versus the training epochs curves for both choices of pooling. Unlike the image manipulation detection task, when a max-pooling layer is used after the "Conv5" layer in the CNN baseline architecture, the performance of CNN to identify the image's source is improved and can achieve 97.57% accuracy. This result demonstrates again that the design choices of CNN in multimedia forensics depend on the forensics detection task.

In the previous experiment, we have shown that for image's source identification the choice of the pooling layer should be different when CNN is used for image manipulation detection. Therefore, one particular design of CNN could not be generalized for another forensics tasks. Thus, in this part of the experiments we assessed the performance of the baseline architecture with different choices of activation function. Fig. 10 shows the identification rate versus the training epochs for the different choices of activation function, i.e., ELU, ReLU and PReLU. In Table 6, we report the best achieved camera model identification rate for each choice of activation function. One can observe that the ReLU-based CNN outperforms the ELU and PReLU networks where their best achieved identification rates are respectively 97.59%, 97.28% and 97.39%.

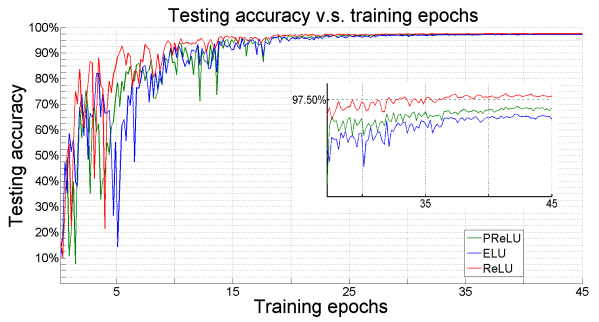

Figure 10. *CNN testing accuracy v.s. training epochs with different activation functions for camera model identification.*

Finally, Table 4 shows the identification rate for each camera model in our database using the ReLU-based baseline CNN. One can notice that the CNN can identify the image's source with an accuracy typically higher than 97% for each model. This demonstrates the ability of the constrained convolutional layer to capture good camera's fingerprints while suppressing the image content. Additionally, these results are very promising since we are using reasonably small input image patches from a single image channel.

Summary In this set of experiments, we have shown that some rules learned from the image manipulation detection task could be generalized for camera model identification. More specifically, the choice of the first convolutional layer should be the same for both forensics CNN detectors. In fact, when a CNN is used with the constrained convolutional layer, we can achieve better identification rate compared to a CNN that uses a fixed HPF layer on top of the network. Furthermore, our results show that when nonlinearity is introduced to the camera's demosaicing features learned by "Constrained Conv" layer the overall identification rate significantly decreases. Unlike the image manipulation detection experiments where the PReLU-based baseline CNN architecture performs the best, in these experiments we found that for source camera model identification a ReLU-based CNN outperforms the other choices of activation functions we examined. Finally, we have seen through our experiments that the choice of the pooling layer after the 1×1 convolutional filters could differs between both tasks. Our results showed that when a max-pooling layer is used after "Conv5" the overall identification rate is improved.

Conclusion

In this paper, we presented the primary steps to design a CNN architecture for multimedia forensics tasks. We showed that CNNs could be used to perform universal image manipulation detection and camera model identification. Through our experiments, we demonstrated the effect of using each element in a CNN architecture on the performance of CNN. Given that image forensics is fundamentally different problem compared to object recognition, several rules used to design CNNs should not be applied when a forensics investigator builds a deep network. To learn image forensics features using a CNN we first need to suppress the image's content, otherwise, it will lead to a classifier that identifies objects and scenes associated with the training images. To address this issue, two competing approaches, which suggest to replace the first convolutional layer in the network, have emerged, i.e., predetermined high-pass filter layer [21] versus constrained convolutional layer [1]. We studied the impact of each of these approaches on the performance of CNN. Our experimental results showed that the constrained convolutional layer improves the identification rate of CNN for both forensics tasks compared to the HPF-based approach.

Furthermore, we studied the effect of introducing different nonlinear operations to the features learned by the constrained convolutional layer. We demonstrated through our experiments that for both forensics tasks any type of nonlinearity would destroy these representative features which decreases the overall identification rate. Additionally, to further improve the performance of CNN, we showed through the experiments that we can accomplish this by increasing the network depth. Moreover, we have seen that if 1×1 convolutional filters are used to learn feature maps association from the highest-level convolutional features in the network, this will lead to a better CNN's performance. Finally, our results showed that some rules learned when designing a CNN for image manipulation detection cannot be applied to camera model indentification task. In fact, we have seen that the best performance in these two forensics tasks was achieved with different choices of activation function and pooling layers.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1553610. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10. ACM, 2016.

[2] G. Cao, Y. Zhao, R. Ni, and A. C. Kot. Unsharp masking sharpening detection via overshoot artifacts analysis. *IEEE Signal Processing Letters*, 18(10):603–606, 2011.

[3] H. Cao and A. C. Kot. Accurate detection of demosaicing regularity for digital image forensics. *IEEE Transactions on Information Forensics and Security*, 4(4):899–910, 2009.

[4] C. Chen and M. C. Stamm. Camera model identification framework using an ensemble of demosaicing features. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[8] T. Filler, J. Fridrich, and M. Goljan. Using sensor pattern noise

for camera model identification. In *2008 15th IEEE International Conference on Image Processing*, pages 1296–1299. IEEE, 2008.

[9] J. Fridrich and J. Kodovskỳ. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[11] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] X. Kang, M. C. Stamm, A. Peng, and K. J. R. Liu. Robust median filtering forensics using an autoregressive model. *IEEE Transactions on Information Forensics and Security,*, 8(9):1456–1468, Sept. 2013.

[14] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *Proceedings of the 10th ACM Workshop on Multimedia and Security*, MM&Sec '08, pages 11–20, New York, NY, USA, 2008. ACM.

[15] M. Kirchner and J. Fridrich. On detection of median filtering in digital images. In *IS&T/SPIE Electronic Imaging*, pages 754110–754110. International Society for Optics and Photonics, 2010.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[18] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[19] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva. A study of co-occurrence based local features for camera model identification. *Multimedia Tools and Applications*, pages 1–17, 2016.

[20] T. Pevny, P. Bas, and J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2):215–224, June 2010.

[21] L. Pibre, P. Jérôme, D. Ienco, and M. Chaumont. Deep learning for steganalysis is better than a rich model with an ensemble classifier, and is natively robust to the cover source-mismatch. *arXiv preprint arXiv:1511.04855*, 2015.

[22] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, Feb. 2005.

[23] X. Qiu, H. Li, W. Luo, and J. Huang. A universal image forensic strategy based on steganalytic model. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 165–170. ACM, 2014.

[24] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[26] M. C. Stamm and K. J. R. Liu. Forensic detection of image manipulation using statistical intrinsic fingerprints. *IEEE Trans. on Information Forensics and Security*, 5(3):492 –506, 2010.

[27] M. C. Stamm, M. Wu, and K. J. R. Liu. Information forensics: An overview of the first decade. *IEEE Access*, 1:167–200, 2013.

[28] A. Swaminathan, M. Wu, and K. J. R. Liu. Nonintrusive component forensics of visual sensors using output images. *IEEE Transactions on Information Forensics and Security*, 2(1):91–106, 2007.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[30] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Author Biography

Belhassen Bayar received the B.S. degree in Electrical Engineering from the Ecole Nationale d'Ingénieurs de Tunis (ENIT), Tunisia, in 2011, and the MS degree in Electrical and Computer Engineering from Rowan University, New Jersey, in 2014. After graduating from ENIT, he worked as a Research Assistant at the University of Arkansas at Little Rock (UALR). In Fall 2014, he joined Drexel University, Pennsylvania, where he is currently a PhD candidate with the Department of Electrical and Computer Engineering. Bayar won the Best Paper Award at the IEEE International Workshop on Genomic Signal Processing and Statistics in 2013. In summer 2015 he interned at Samsung Reasearch America in Mountain View, California. His main research interests are in image forensics, machine learning and signal processing.

Matthew C. Stamm received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Maryland, College Park in 2004, 2011, and 2012, respectively. He is an Assistant Professor with in the Department of Electrical and Computer Engineering at Drexel University. From 2004 to 2006, he was a Radar Systems Engineer with the Johns Hopkins University Applied Physics Laboratory. His research interests include multimedia forensics, signal processing, information security, and machine learning.