LEARNED FORENSIC SOURCE SIMILARITY FOR UNKNOWN CAMERA MODELS

Owen Mayer, Matthew C. Stamm

Drexel University
Department of Electrical and Computer Engineering
Philadelphia, Pennsylvania USA

ABSTRACT

Information about an image's source camera model is important knowledge in many forensic investigations. In this paper we propose a system that compares two image patches to determine if they were captured by the same camera model. To do this, we first train a CNN based feature extractor to output generic, high level features which encode information about the source camera model of an image patch. Then, we learn a similarity measure that maps pairs of these features to a score indicating whether the two image patches were captured by the same or different camera models. We show that our proposed system accurately determines if two patches were captured by the same or different camera models, even when the camera models are unknown to the investigator. We also demonstrate the utility of this approach for image splicing detection and localization.

Index Terms— Multimedia Forensics, Camera Model Identification, Forensic Similarity, Image Splicing

1. INTRODUCTION

Digital images are used in many scenarios, such as in news reporting, courts of law, government intelligence, as well as in communication on variety of social media platforms. However, modern software makes it easy for images to be manipulated or their source to be masked. For example, it is common that images downloaded from online databases, such as Facebook, have their source masked through metadata removal. To address this, researchers have developed a number of multimedia forensic techniques that are used to verify the source and authenticity of digital images [1].

One important multimedia forensics task is camera model identification, where the camera model that was used to capture an image is identified. A variety of techniques exist for camera model identification [2]. These techniques employ a range of features including CFA demosaicing artifacts [3–7], sensor pattern noise [8], local binary patterns [9], noise models [10], and chromatic aberration [11]. More recently, convolutional neural networks (CNNs) have been shown to be powerful tools for camera model identification [12–16].

While these approaches for camera model identification are effective, a major drawback with many approaches is that they assume a closed set of camera models. That is they require prior knowledge, i.e. training data, from a source camera model in order to accurately identify images captured by that model. However, given the large number of camera models that exist, it is often unlikely that the camera model of an image under investigation exists within a training set. In many cases it is not feasible to scale an identification system to contain large numbers of training models.

This material is based upon work supported by the National Science Foundation under Grant No. 1553610.

Furthermore, in many scenarios an investigator may not be concerned with the exact camera model that was used to capture an image. Such scenarios include verifying whether a database of images were taken by the same camera model, discovery of intellectual property theft (e.g. unauthorized use of proprietary demosaicing strategies), or determining whether a splicing forgery has occurred within an image. When detecting splicing operations, camera model identification features can be used to detect when a spliced image is a composite of image content from two different camera models [17, 18]. Recent work by Bondi et al. shows that the outputs from a CNN-based camera model identification system can be iteratively clustered to detect and localize image splicing [19]. They found their method to be effective even when the spliced image does not contain content from camera models used to train the system.

In this paper, we propose a new system that determines if two image patches are captured by different camera models or from the same camera model. Our approach is different from camera model identification in that it does not determine the exact camera model that was used to capture either patch. The power of our proposed approach is that it is able to compare camera models that were not used to train the system. This allows an investigator to learn important information about images captured by *any* camera, and isn't limited by the set of camera models in the investigator's database.

To accomplish this, we develop a measure that describes the similarity between the camera models that were used to capture two image patches. We do this by first learning a CNN-based feature extractor that outputs generic, high level features useful for camera model identification. Then, we train a similarity network that maps pairings of these features to a score that indicates whether the two image patches were captured by the same camera model or two different camera models. We experimentally show that our proposed approach effectively differentiates between camera models, even if the source camera models were not used to train the system. Additionally, we show an example that demonstrates the promise of this approach for detecting an localizing image splicing manipulations.

2. PROBLEM FORMULATION & SYSTEM OVERVIEW

There are many scenarios where an investigator may wish to know information about the source camera model of an image, but may not have sufficient training data to identify the true source. To address this, we propose a method that compares two image patches, and renders a decision about whether the source camera models of these patches are the same or different. This comparison is performed without needing training information from the source camera model of either patch. This approach provides less information about the true camera model than a camera model identification method, but it allows an investigator to gain important information about images that are sourced from *any* camera model, not just those belonging to

an investigator's training set.

We assume that an investigator has access to a set of camera models, referred to as *known* camera models, from which they can gather training images. However, the investigator may be presented with images sourced from camera models that do not belong to this set, referred to as *unknown* camera models. As a result, it is important that our proposed system be able to learn a similarity measure from images belonging to known camera models, and also effectively compare images belonging to unknown camera models.

Our proposed approach compares two image patches and determines whether the source camera models are different or the same. To do this, we develop a comparison function $C : \mathbb{X} \times \mathbb{X} \rightarrow \{0, 1\}$, where $\mathbb{X}$ is the space of image patches. The mapping is such that for two inputs $X_1 \in \mathbb{X}$, $X_2 \in \mathbb{X}$, the comparison function will map to either 0 or 1. An output of 0 indicates that the source camera models of the two image patches are different, and an output of 1 indicates that the source camera model is the same, i.e.

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } X_1, X_2 \text{ from different camera models,} \\ 1 & \text{if } X_1, X_2 \text{ from the same camera model.} \end{cases} \quad (1)$$

This problem is tackled in three steps. First, a feature extractor function $f : \mathbb{X} \rightarrow \mathbb{R}^N$ is learned that maps an input image patch $X \in \mathbb{X}$ to an N-dimensional feature space, which encodes high level camera model information. Next, a feature similarity measure $S : \mathbb{R}^N \times \mathbb{R}^N \rightarrow [0, 1]$ is learned that maps pairs of features to a space bounded by 0 and 1, with 0 indicating that the source camera models are dissimilar and 1 indicating that they are similar. Finally, this measure is compared to a threshold η to decide whether the source camera models are the same or different.

The overall system,

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } S(f(X_1), f(X_2)) \leq \eta \\ 1 & \text{if } S(f(X_1), f(X_2)) > \eta \end{cases}, \quad (2)$$

is a combination of these three steps. The system inputs two image patches and maps each to a feature space that encodes high-level camera model information. Then, the pair of features are mapped to an output score indicating the similarity of their source camera models which is finally compared to a threshold.

3. PROPOSED APPROACH

To create our proposed system, we first train a convolutional neural network (CNN) to learn a feature extractor that produces generic, high-level features from image patches that are useful for camera model identification. These features are related to components of the image capturing pipeline that uniquely identify different camera models. Then, a similarity network is trained using pairs of these features to learn a similarity measure. The similarity network is a multilayer neural network that first maps the extracted features to a new feature space. It then compares the newly mapped features to each other to finally render a decision on whether the two input image patches are sourced from the same or different camera models. A view of the overall network architecture is shown in Fig. 1.

The whole system is learned in two phases. In the first learning phase, called *Learning Phase A*, the feature-extractor network is trained to identify a set of camera models $\mathcal{A}$. Then, during the second learning phase, called *Learning Phase B*, the similarity network is trained to learn similarity between pairs of high-level features extracted from a second set of camera models $\mathcal{B}$. The camera model sets $\mathcal{A}$ and $\mathcal{B}$ are disjoint to ensure that the similarity network learns to differentiate camera models that have not been learned by the feature extractor, including unknown camera models.

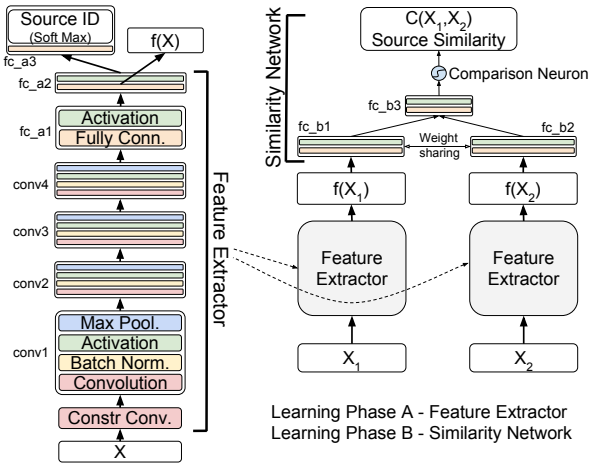


Fig. 1. Proposed system architecture.

3.1. Feature extractor - Learning Phase A

The first step in our approach is to learn an extractor for generic, high-level features that encode camera model information. To do this, we train a convolutional neural network (CNN) to identify camera models from image patches, following the approach in [13]. This training is called *Learning Phase A*. After training, a feature representation $f(X)$ of an image patch is extracted by feeding forward the patch X through the network, then extracting the neuron values, pre-activation, of a high-level fully connected layer. The feature representation $f(X)$ corresponds to a generic, high-level encoding of the patch's source camera model.

The feature extractor network, depicted on the left in Fig. 1, contains a constrained convolutional layer followed by four convolutional layers, labeled conv1-4. Each of the four non-constrained layers is composed of batch normalization, hyberbolic tangent activation function, and pooling steps. The constrained convolutional layer, introduced in [20], contains 3 kernels each of size 5x5. A constraint is placed on weights $\mathbf{w}$ for each kernel in the constrained layer such that

$$\begin{cases} \mathbf{w}(0,0) = -1 \\ \sum_{(l,m)\neq(0,0)} \mathbf{w}(l,m) = 1 \end{cases}. \quad (3)$$

That is, the constrained layer has kernels where the central value is -1, and the rest of the weights sum to 1. This constrained convolutional layer roughly corresponds to a prediction error filter, which jointly suppresses image content while allowing the network to learn camera model related pixel value dependencies introduced by the camera's image processing pipeline [16]. Details of the exact parameters of this network can be found in [13].

Following the convolutional layers are a series of 2 fully-connected neuron layers, each with 200 neurons that have hyberbolic tangent activation. These fully connected layers are labeled fc_a1 and fc_a2 in Fig. 1. A final fully-connected layer with $|\mathcal{A}|$ neurons and a softmax is used to identify the input camera model, where $|\mathcal{A}|$ is the number of camera models in the training camera model set $\mathcal{A}$.

To extract features $f(X)$ from an image patch X, we feed X forward through the trained network and record the neuron values, pre-activation, of layer fc_a2. The feature vector $f(X)$ has dimension 200 and encodes information about the source camera model of X.

3.2. Similarity network - Learning Phase B

The second step of our approach is to map pairings of the extracted features to a similarity score that indicates whether a pair of input im-

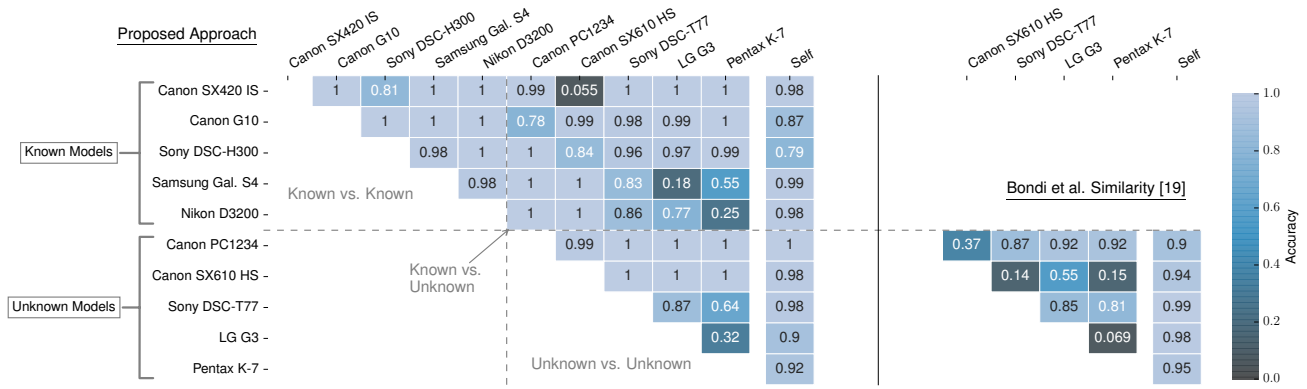


Fig. 2. Source comparison accuracy. The triangular matrices show rates of correctly detecting two patches as sourced from different camera models. The columns labeled "Self" show rates of correctly detecting two patches as sourced from the same camera model. Results of our proposed method are on the left. On the right, we compare to an approach adapted from Bondi et al. [19].

age patches are sourced from different camera models or the same model. To do this, we train a multilayer neural network to compare pairs of features that are extracted from the feature extractor described in Sec. 3.1. We call this network a similarity network. The construction of this network is modeled after image comparison neural networks in [21–23] called "Siamese" networks.

The architecture for this similarity network is depicted in Fig. 1 on the right. To start, we take two copies the pre-trained feature extractor network learned in Learning Phase A. The extracted features $f(X_1)$ and $f(X_2)$ from two image patches are used as inputs to the similarity network. The features $f(X_1)$ and $f(X_2)$ are separately connected to two layers *fc_b1* and *fc_b2*. Both layers contain 1024 neurons with a parametric ReLU activation function. These layers learn to map the input features to a new feature space. During training, the weights are shared between *fc_b1* and *fc_b2*.

Then, layers *fc_b1* and *fc_b2* are both fully connected to a single layer of neurons, labeled *fc_b3* in Fig. 1. This layer contains 64 neurons, each with parametric ReLU activation. This layer of neurons is then fully connected to a single neuron with sigmoid activation. The activation value of this neuron corresponds to the similarity value $S(f(X_1), f(X_2))$ in Eq. (2). An activation value above a threshold $\eta = 0.5$ indicates that the pair of input patches X_1 and X_2 are captured by the same camera model.

During Learning Phase B, the similarity network is trained using pairs of image patches with labels that correspond to whether the patches were captured by different camera models (0) or the same camera model (1). To ensure that the similarity network learns to compare camera models that are unknown, the set of camera models used in Learning Phase B is disjoint to the set of camera models used in Learning Phase A. Training is performed through iterative back propagation with binary cross entropy loss. During training, the weights of the feature extractor network are frozen, and only weights in the similarity network are learned. Additionally, 50% dropout is used on the output of layers *fc_b1* and *fc_b2* to help prevent overfitting. Together, the feature extractor and similarity network form a full system, which inputs two patches and outputs a decision on whether they are captured by the same or different camera models.

Additionally, to make a decision on an image patch we require that it have an entropy value above a certain threshold. This is done to ensure that each patch contains sufficient texture information to effectively encode its source camera model. Entropy h is defined by $h = -\sum_{k=0}^{255} p_k \ln(p_k)$, where p_k is the proportion of pixels in the patch that are of value k. Here, entropy is measured in nats.

4. EXPERIMENTAL RESULTS

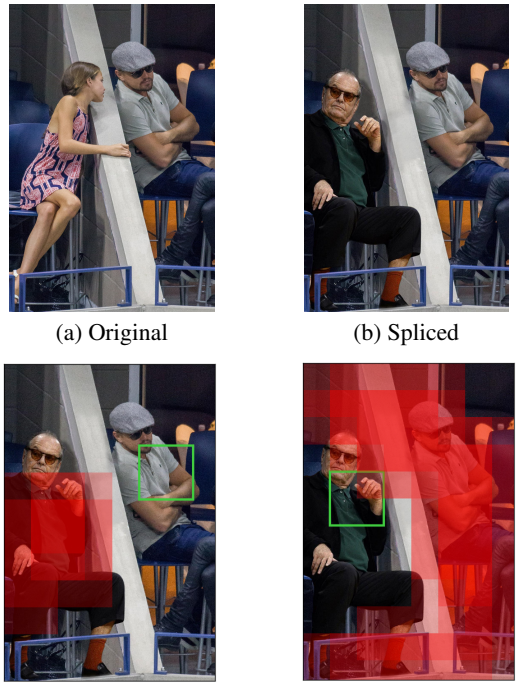
To assess the efficacy of our approach, we conducted two experiments that test our sytem under various investigation scenarios. To do this, we started with a database of images from 65 camera models, of which 20 are from the publicly available Dresden Image Database [24]. The remaining 45 camera models are from our own database composed of point-and-shoot, cellphone, and DSLR cameras. For each camera model in our database, we collected 300 images with diverse and varied scene content. The set of all camera models were split into 3 disjoint sets; 40 camera models were placed in set $\mathcal{A}$, 20 different models in set $\mathcal{B}$, and the remaining 5 in set $\mathcal{C}$.

The feature extractor network was trained (Learning Phase A) using image patches from camera models in $\mathcal{A}$. For each camera model we used 50,000 unique non-overlapping patches of size 256×256, with locations chosen at random, yielding 2,000,000 total training patches. Only the green color channel was used. The network was trained using stochastic gradient descent for 30 epochs, with an initial learning rate of 0.001 decreased by a factor of 2 every 3 epochs, decay of 0.0005, and momentum of 0.9. Training of the feature extractor was performed using Caffe on a GTX 1080TI GPU.

Next, the similarity network was trained (Learning Phase B), using features extracted from image patches sourced from camera models in $\mathcal{B}$. For each camera model, we used 20,000 unique non-overlapping patches, chosen at random, to extract features yielding a total of 400,000 training patches. As described in Sec. 3.2, only features from image patches whose entropy was greater than 4 nats were used. A training set was made composed of 250,000 random pairings of features from different camera models and 250,000 random pairings of features from the same camera model. The similarity network was trained using the Adam optimizer [25] for 60 epochs, with a learning rate of 0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-6}$. Training was performed using Keras with a Tensorflow backend on a GTX 1080TI GPU.

4.1. Source differentiation

In our first experiment, we assessed the performance of our system in determining whether two image patches were captured by different or the same camera models. We did this using three scenarios: 1) both patches captured by camera models known to the investigator, 2) one patch captured by a known camera model and the other by an unknown camera model, and 3) both patches captured by unknown camera models. These scenarios represent all scenarios that an investigator may encounter.



(a) Original (b) Spliced

(c) Host Reference (d) Foreign Reference

Fig. 3. Splicing detection example. An (a) original and (b) spliced image. In (c) and (d), patches are highlighted that are found to have a different source than the reference patch outlined in green.

To do this, we started with 10,000 unique patches each from 10 different camera models, yielding 100,000 total patches. Five of the camera models were known camera models in $\mathcal{A}$, and the other five were the unknown camera models in $\mathcal{C}$. The patches were randomly paired and then tested by our proposed comparison system. As described in Sec. 3.2, only image patches whose entropy was greater than 4 nats were used.

The results of this experiment are shown in Fig. 2. The triangular matrices show rates of correctly detecting two patches as captured by different camera models. The columns labeled "Self" show rates of correctly detecting two patches as sourced from the same camera model. For example, when one patch is captured by a Sony DSC-H300 and the other by an LG G3, our proposed approach correctly detects the camera models as different at a rate of 98%. When both patches are captured by a Samsung Galaxy S4, our proposed approach correctly detects patch sources as the same at a rate of 99%.

The overall accuracy achieved by our method was 91.1%. In scenarios where both source camera models were known, our system achieved 95.8% accuracy. In scenarios where both source camera models were unknown, our proposed system achieved 90.6% accuracy. In the scenarios where one camera model was known and the other unknown, our system achieved 84.0% accuracy. However, in these scenarios a few particular cases, such as the Canon SX420IS versus Canon SX630HS, greatly reduced the detection accuracy. Most unknown versus known cases were correctly detected at a rate of 99% or greater. The results of this experiment demonstrate that our proposed system accurately determines meaningful information about the source camera models of two patches, even if one or both of the camera models are unknown.

We note that there were a few cases that were difficult for our proposed approach to detect. These instances typically occurred when the two camera models were produced by same manufacturer, or when both of the camera models were a cellphone or DSLR. The latter illustrates the need for a diverse training set, since our known database is primarily composed of point-and-shoot camera models.

In addition, we compared to an approach adapted from Bondi et. al in [19]. Briefly, their method detects and localizes image splicing by iteratively clustering image patches into either original or forged sets. As part of their algorithm, they calculate the Bray-Curtis distance between the softmax scores of patches as output by a camera model identification CNN. This distance is used to determine set membership (i.e. original or forged content). We tested the efficacy of using this distance to compare the source of two image patches.

To compare to their approach, we first used their pre-trained convolutional neural network from [19] to output the softmax scores of image patches. Then, we calculated the Bray-Curtis distance between pairs of these scores, and used a threshold to detect whether the camera models were the same or different. Since all camera models in this experiment were unknown with respect to the Bondi et al. CNN (with the exception of Sony DSC-T77), we compare only to the unknown cases. We set the decision threshold so that the false alarm rate is equal to the false alarm rate of the unknown cases in our system (4.5%), to make a fair comparison.

In all cases our proposed method outperforms the method we adapted from Bondi et al, except when one patch was sourced by a Sony DSC-T77 and the other a Pentax K-7 camera. This shows that our proposed system more accurately compares the camera model source of image patches than the approach we adapted from [19]. We note that the Sony DSC-T77 is known relative to the Bondi et al. neural network yet is unknown to our system.

4.2. Splicing detection

An investigator may also be concerned with whether an image is a spliced forgery. In this experiment, we demonstrate the promise of our proposed approach for detecting spliced images that are a composite of content from two different camera models. To do this, we took a spliced image and split it into 256×256 patches with 50% overlap. We selected one patch as a reference, computed its comparison score to every other patch in the image, and then highlighted all of the patches that were detected as captured by a different source.

An example of this analysis is shown in Fig. 3 on an image downloaded from an online forum[1]. The original and spliced images are shown in (a) and (b). In (c) we selected a patch in the original part of the image as reference, and our method highlighted the foreign portion of the image a having a different source. In (d) we selected a patch in the foreign part of the image as reference, and our method highlighted the original parts of the image as having a different source. The results of this experiment demonstrate the promise of our proposed approach at detecting and localizing spliced images.

5. CONCLUSION

In this paper we propose a system to compare the source camera model of two image patches. To do this, we first train a CNN-based feature extractor to output generic, high-level features which encode information about the source camera model of an image patch. Then, we learn a similarity measure that maps pairs of these features to a score indicating whether the two image patches were captured by the same or different camera models. We show that our proposed system correctly determines if two image patches were captured by the same or different camera models, even when the camera models are unknown to the investigator.

[1] www.reddit.com/r/photoshopbattles

6. REFERENCES

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

[2] M. Kirchner and T. Gloe, "Forensic camera model identification," *Handbook of Digital Forensics of Multimedia Data and Devices*, pp. 329–374, 2015.

[3] A. Swaminathan, M. Wu, and K. R. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 91–106, 2007.

[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*. IEEE, 2015, pp. 1–6.

[5] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on cfa interpolation," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3. IEEE, 2005, pp. III–69.

[6] 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, 2009.

[7] X. Zhao and M. C. Stamm, "Computationally efficient demosaicing filter estimation for forensic camera model identification," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 151–155.

[8] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, 2008, pp. 1296–1299.

[9] G. Xu and Y. Q. Shi, "Camera model identification using local binary patterns," in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*. IEEE, 2012, pp. 392–397.

[10] T. H. Thai, R. Cogranne, and F. Retraint, "Camera model identification based on the heteroscedastic noise model," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 250–263, 2014.

[11] L. T. Van, S. Emmanuel, and M. S. Kankanhalli, "Identifying source cell phone using chromatic aberration," in *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE, 2007, pp. 883–886.

[12] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*. IEEE, 2016, pp. 1–6.

[13] B. Bayar and M. C. Stamm, "Augmented convolutional feature maps for robust CNN-based camera model identification," in *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[14] 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, 2017.

[15] 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," *Electronic Imaging*, vol. 2017, no. 7, pp. 67–76, 2017.

[16] B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Electronic Imaging*, vol. 2017, no. 7, pp. 77–86, 2017.

[17] A. E. Dirik and N. Memon, "Image tamper detection based on demosaicing artifacts," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009, pp. 1497–1500.

[18] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of CFA artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, 2012.

[19] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "Tampering detection and localization through clustering of camera-based CNN features," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 1855–1864.

[20] 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*. ACM, 2016, pp. 5–10.

[21] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[22] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'siamese' time delay neural network," in *Advances in Neural Information Processing Systems*, 1994, pp. 737–744.

[23] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[24] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, 2010.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.