Reliability Map Estimation For CNN-Based Camera Model Attribution

David Güera
Purdue University
West Lafayette, Indiana

Sri Kalyan Yarlagadda
Purdue University
West Lafayette, Indiana

Paolo Bestagini
Politecnico di Milano
Milan, Italy

Fengqing Zhu
Purdue University
West Lafayette, Indiana

Stefano Tubaro
Politecnico di Milano
Milan, Italy

Edward J. Delp
Purdue University
West Lafayette, Indiana

Abstract

Among the image forensic issues investigated in the last few years, great attention has been devoted to blind camera model attribution. This refers to the problem of detecting which camera model has been used to acquire an image by only exploiting pixel information. Solving this problem has great impact on image integrity assessment as well as on authenticity verification. Recent advancements that use convolutional neural networks (CNNs) in the media forensic field have enabled camera model attribution methods to work well even on small image patches. These improvements are also important for determining forgery localization. Some patches of an image may not contain enough information related to the camera model (e.g., saturated patches). In this paper, we propose a CNN-based solution to estimate the camera model attribution reliability of a given image patch. We show that we can estimate a reliability-map indicating which portions of the image contain reliable camera traces. Testing using a well known dataset confirms that by using this information, it is possible to increase small patch camera model attribution accuracy by more than 8% *on a single patch.*

1. Introduction

Due to the widespread availability of inexpensive image capturing devices (e.g., cameras and smartphones) and user-friendly editing software (e.g., GIMP and Adobe PhotoShop), image manipulation is very easy. For this reason, the multimedia forensic community has developed techniques for image authenticity detection and integrity assessment [1, 2, 3].

Among the problems considered in the forensic literature, one important problem is camera model attribution, which consists in estimating the camera model used to acquire an image [4]. This proves useful when a forensic analyst needs to link an image under investigation to a user [5], or to detect possible image manipulations [6, 7] (e.g., splicing of pictures from different cameras).

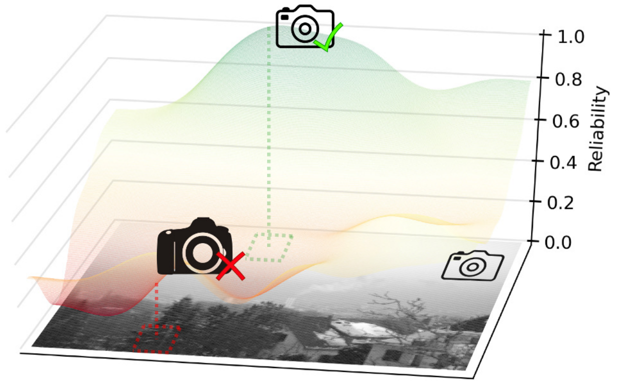


Figure 1. Reliability map representation for an example image taken with a given camera. In this case, patches belonging to the sky (green box) are more likely to provide accurate camera model attribution than patches containing textures (red box).

Linking an image to a camera can in principle be trivially done exploiting image header information (e.g., EXIF data). It is also true that image headers are not reliable (e.g., anyone can tamper with them) or not always available (e.g., decoded images and screen captures). Therefore, the need for a series of blind methodologies has led to the development of pixel-based only information extraction methods.

These methods leverage the fact that image acquisition pipeline is slightly different for each camera model and manufacturer (e.g., different sensors and color equalization techniques). Therefore, each image contains characteristic "fingerprints" that enable one to understand which pipeline has been used and hence the camera model. Among these techniques, exploiting photo sensor non uniformity (PRNU) is particularly robust and enables camera instance identification [8, 9]. Other methods exploit traces left by color filter array (CFA) interpolation [10, 11, 12], camera lenses [13], histogram equalization [14] or noise [15]. Alternatively, a series of methods extracting statistical features from the pixel-domain and exploiting supervised machine-learning classifier have also been proposed [16, 17, 18].

Due to the advancements brought by deep learning techniques in the last few years, the forensic community is also exploring convolutional neural networks (CNNs) for camera model identification [19]. Interestingly, the approach in [20] has shown the possibility of accurately estimating the camera model used to acquire an image by analyzing a small portion of the image (i.e., a 64×64 color image patch). This has lead to the development of forgery localization techniques [21].

In this paper we propose a CNN-based method for estimating patch reliability for camera model attribution. As explained in [22], not all image patches contain enough discriminative information to estimate the camera model (e.g., saturated areas and too dark regions). Leveraging the network proposed in [20], we show how it is possible to determine whether an image patch contains reliable camera model traces for camera model attribution. Using this technique, we build a reliability map, which indicates the likelihood of each image region to be possibly used for camera model attribution, as shown in Figure 1. This map can be used to select only reliable patches for camera model attribution. Additionally, it can also be used to drive tampering localization methods [21] by providing valuable information on which patches should be considered to be unreliable.

The proposed method leverages CNN feature learning capabilities and transfer learning training strategies. Specifically, we make use of a CNN composed by the architecture proposed in [20] as feature extractor, followed by a series of fully connected layers for patch reliability estimation. Transfer learning enables to preserve part of the CNN weights of [20], and train the whole architecture end-to-end with a reduced number of image patches. Our strategy is validated on the Dresden Image Database [23]. We first validate the proposed architecture and training strategy. Then, we compare the proposed solution against a set of baseline methodologies based on classic supervised machine-learning techniques. Finally, we show how it is possible to increase camera model attribution accuracy by more than 8% with respect to [20] using the proposed method.

2. Problem Statement and Related Work

In this section we introduce the problem formulation with the notation used throughout the paper. We then provide the reader a brief overview about CNNs and their use in multimedia forensics.

2.1. Problem Formulation

Let us consider a color image $\mathbf{I}$ acquired with camera model l belonging to a set of known camera models $\mathcal{L}$. In this paper, we consider the patch-based closed-set camera model attribution problem as presented in [20]. Given an image $\mathbf{I}$, this means

- Select a subset of K color patches $\mathbf{P}_k,\ k \in [1, K]$.
- Obtain an estimate $\hat{l}_k = \mathcal{C}(\mathbf{P}_k)$ of the camera model associated with each patch through a camera attribution function $\mathcal{C}$.
- Optionally obtain final camera model estimate $\hat{l}$ through majority voting over $\hat{l}_k,\ k \in [1, K]$.

Our goal is to detect whether a patch $\mathbf{P}_k$ is a good candidate for camera model attribution estimation. To this purpose, we propose a CNN architecture that learns a function $\mathcal{G}$ expressing the likelihood of a patch $\mathbf{P}_k$ to provide correct camera model identification, i.e., $g_k = \mathcal{G}(\mathbf{P}_k)$. High values of g_k indicate high probability of patch $\mathbf{P}_k$ to provide correct camera information. Conversely, low g_k values are attributed to patches $\mathbf{P}_k$ that cannot be correctly classified. Pixel-wise likelihood is then represented by means of a reliability map $\mathbf{M}$, showing which portion of an image is a good candidate to estimate image camera model, as shown in Figure 1.

2.2. Convolutional Neural Networks in Multimedia Forensics

In this section, we present a brief overview of the foundations of convolutional neural networks (CNNs) that are needed to follow the paper. For a thorough review on CNNs, we refer the readers of this paper to Chapter 9 of [24].

Deep learning and in particular CNNs have shown very good performance in several computer vision applications such as visual object recognition, object detection and many other domains such as drug discovery and genomics [25]. Inspired by how the human vision works, the layers of a convolutional network have neurons arranged in three dimensions, so each layer has a width height, and depth. The neurons in a convolutional layer are only connected to a small, local region of the preceding layer, so we avoid wasting resources as it is common in fully-connected neurons. 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. CNNs enable learning data-driven, highly representative, layered hierarchical image features from sufficient training data

To better understand the role of each layer, we describe the most common building blocks in a CNN:

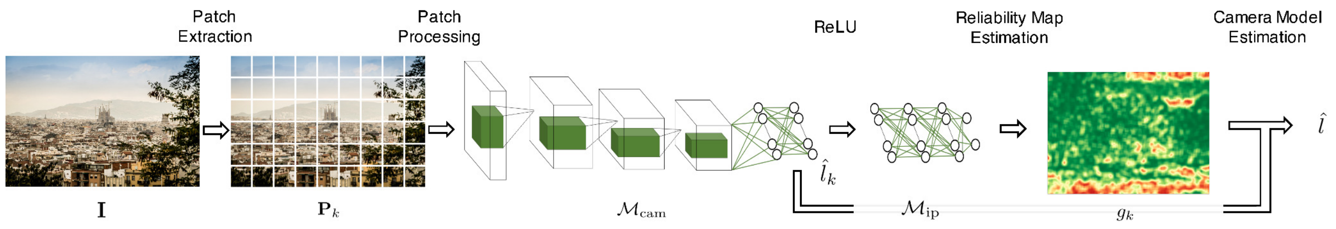


Figure 2. Block diagram of the proposed approach. Image $\mathbf{I}$ is split into patches. Each patch $\mathbf{P}_k,\ k \in [0, K]$ is processed by the proposed CNN (composed by $\mathcal{M}_{\text{cam}}$ and $\mathcal{M}_{\text{ip}}$) to obtain a reliability score g_k and a camera model estimate $\hat{l}_k$. The reliability map is determined from all $g_k,\ k \in [0, K]$ values, and the overall picture camera model estimate $\hat{l}$ can be computed.

- *Convolution*: each convolution layer is a filterbank, whose filters impulse response h are learned through training. Given an input signal x, the output of each filter is $y = x * h$, i.e., the valid part of the linear convolution. Convolution is typically done on 3D representations consisting of the spatial coordinates (x, y) and the number of feature maps p (e.g., $p = 3$ for an RGB input).

- *Max pooling*: returns the maximum value of the input x evaluated across small windows (typically of 3x3 pixels).

- *ReLU*: Rectified Linear Units use the rectification function $y = \max(0, x)$ to the input x, thus clipping negative values to zero.

- *Inner Product*: indicates that the input of each neuron of the next layer is a linear combination of all the outputs of the previous layer. Combination weights are estimated during training.

- *SoftMax*: maps the input into compositional data (i.e., each value is in the range $[0, 1]$, and they all sum up to one). This is particularly useful at the end of the network in order to interpret its outputs as probability values.

There has been a growing interest in using convolutional neural networks in the fields of image forensics and steganalysis [26, 27]. These papers mainly focus on architectural design of CNNs where a single CNN model is trained and then tested in experiments. Data-driven models have recently proved valuable for other multimedia forensic applications as well [28, 29]. Moreover, initial exploratory solutions targeting camera model identification [30, 20, 21] 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. As a matter of fact, the use of CNNs also makes it possible to capture characteristic traces left by non-linear and hard to model operations present in the image acquisition pipeline of capturing devices.

In this paper, we employ CNNs as base learners and test several different training strategies and network topologies. In our study, at first, a recently proposed CNN architecture is adopted as a feature extractor, trained on a random subsample of the training dataset. An intermediate feature representation is then extracted from the original data and pooled together to form new features ready for the second level of classification. Results have indicated that learning from intermediate representation in CNNs instead of output probabilities, and then jointly retraining the final architecture, leads to performance improvement.

3. Patch Reliability Estimation Method

In this section we provide details of our method for patch reliability estimation and camera attribution. The proposed pipeline is composed by the following steps (see Figure 2):

1. The image under analysis is split into patches
2. A CNN is used to estimate patch reliability likelihood
3. From the same CNN we estimate a camera model for each patch
4. A reliability mask is constructed and camera attribution of the whole image is performed

Below is a detailed explanation of each step.

3.1. Patch Extraction

The proposed method works by analyzing image patches. The first step is to split the color image $\mathbf{I}$ into a set of K patches $\mathbf{P}_k,\ k \in [0, K]$. Each patch has 64×64 pixel resolution. The patch extraction stride can range from 1 to 64 per dimension, depending on the amount of desired overlap. This can be chosen to balance the trade-off between mask resolution reliability and computational burden.

3.2. Patch Camera Reliability

Each patch $\mathbf{P}_k$ is input into the CNN $\mathcal{M}$ shown in Figure 3, which can be logically split into two parts ($\mathcal{M}_{\text{cam}}$ and

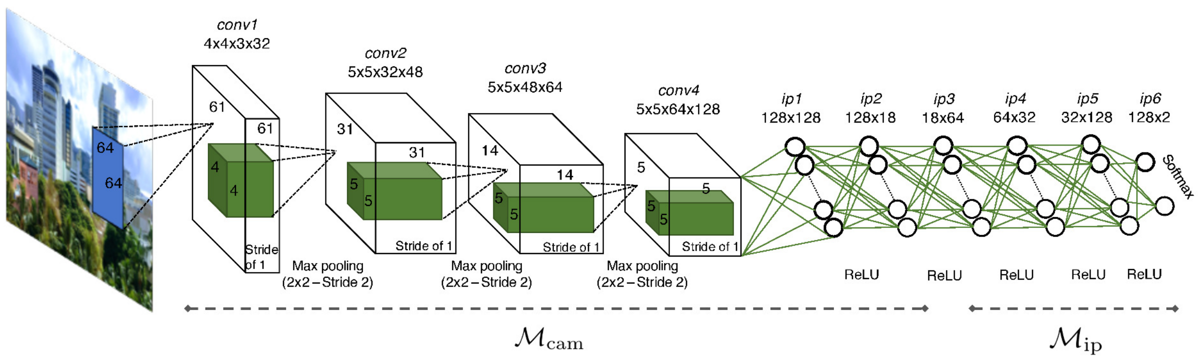


Figure 3. Representation of the proposed CNN architecture $\mathcal{M}$ working on image patches. The first part ($\mathcal{M}_{\text{cam}}$) computes a $|\mathcal{L}|$-element feature vector used for camera attribution. The second part ($\mathcal{M}_{\text{ip}}$) is used to derive the camera model attribution reliability.

$\mathcal{M}_{\text{ip}}$) connected through a ReLU activation layer. Our proposed CNN learns a patch reliability function $\mathcal{G}$ and returns the patch reliability $g_k = \mathcal{G}(\mathbf{P}_k)$.

The first part (i.e., $\mathcal{M}_{\text{cam}}$) is the CNN presented in [20] without last layer's activation. The rationale behind this choice is that this network is already known to be able to extract characteristic camera information. Therefore, we can mainly think of this portion of the proposed CNN as the feature extractor, turning a patch $\mathbf{P}_k$ into a feature vector in $\mathbb{R}^{|\mathcal{L}|}$, where $|\mathcal{L}|$ is the number of considered camera models. Formally, $\mathcal{M}_{\text{cam}}$ is composed by:

- *conv1*: convolutional layer with 32 filters of size $4 \times 4 \times 3$ and stride 1.
- *conv2*: convolutional layer with 48 filters of size $5 \times 5 \times 32$ and stride 1.
- *conv3*: convolutional layer with 64 filters of size $5 \times 5 \times 48$ and stride 1.
- *conv4*: convolutional layer with 128 filters of size $5 \times 5 \times 64$ and stride 1, which outputs a vector with 128 elements.
- *ip1*: inner product layer with 128 output neurons followed by a ReLU layer to produce a 128 dimensional feature vector.
- *ip2*: final $128 \times |\mathcal{L}|$ inner product layer.

The first three convolutional layers are followed by max-pooling layers with 2×2 kernels and 2×2 stride. This network contains 360 462 trainable parameters.

The second part of our architecture (i.e., $\mathcal{M}_{\text{ip}}$) is composed by a series of inner product layers followed by ReLU activations. This part of the proposed CNN can be considered as a two-class classifier trying to distinguish between patches that can be correctly classified, and patches that cannot correctly be attributed to their camera model. As shall be clear in Section 4, we tested different possible $\mathcal{M}_{\text{ip}}$ architecture candidates, to decide upon the following structure (denoted later on as $\mathcal{M}_{\text{ip}}^4$ due to the 4 layers that characterize it):

- *ip3*: inner product layer with 64 output neurons followed by ReLU
- *ip4*: inner product layer with 32 output neurons followed by ReLU
- *ip5*: inner product layer with 128 output neurons followed by ReLU
- *ip6* inner product layer with 2 output neurons followed by softmax normalization

The first element of the softmax output vector can be considered the likelihood of a patch to be correctly classified. Therefore, we consider this value as g_k, and the transfer function learned by the whole $\mathcal{M}$ network as $\mathcal{G}$.

3.3. Patch Camera Attribution

In order to detect the camera model from each patch $\mathbf{P}_k$, we exploit the architecture $\mathcal{M}_{\text{cam}}$. As explained in [20], this CNN output is a $|\mathcal{L}|$-element vector, whose argmax indicates the used camera model $\hat{l}_k \in \mathcal{L}$. Note that the softmax normalization at the end of $\mathcal{M}_{\text{cam}}$ (as proposed in [20]) is not needed in this situation, as it only impacts the training strategy and not the argmax we are interested in.

3.4. Reliability-Map and Camera Attribution

In order to compute the reliability map $\mathbf{M}$, we aggregate all g_k values estimated from patches $\mathbf{P}_k,\ k \in [1, K]$. This is done by generating a bidimensional matrix $\mathbf{M}$ with the same size of image $\mathbf{I}$, and fill in the positions covered by the patch $\mathbf{P}_k$ with the corresponding g_k values. In case of overlapping patches, g_k values are averaged. This map provides pixel-wise information about image regions from which reliable patches can be extracted. A few examples of estimated maps $\mathbf{M}$ are reported in Figure 4.

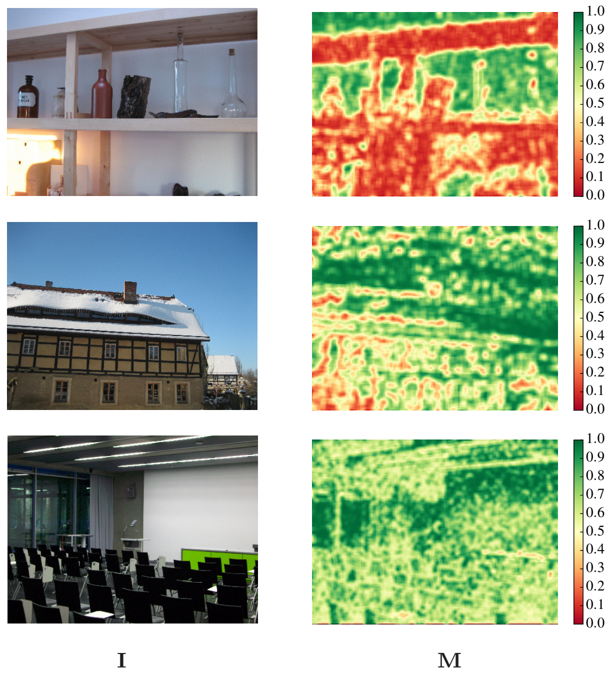


Figure 4. Examples of images $\mathbf{I}$ (left) and the estimated reliability maps $\mathbf{M}$ (right). Patch reliability is not always strictly linked to the image semantic content. Green areas represent high g_k values, thus reliable regions.

Finally, to attribute image $\mathbf{I}$ to a camera model, it is possible to select l_k only for the most reliable patch (i.e., highest g_k), or perform majority voting on estimated l_k values belonging only to reliable regions, i.e., $\{l_k | g_k > \gamma\}$, where $\gamma \in [0, 1]$ is the reliability threshold (set to 0.5 in this paper).

4. Experimental Results

In this section we report the details about our experiments. First, we describe the dataset. Then, we provide an insight on the used training strategies.

4.1. Dataset

In this paper we evaluate our solution adapting the dataset splitting strategy proposed in [4, 20, 22] to our problem. This strategy is tailored to the Dresden Image Dataset [23], which consists of 73 devices belonging to 25 different camera models. A variable number of shots are taken with each device. Different motives are shot from each position. We refer to a scene as combination of geographical position with a specific motive. With this definition in mind, the dataset consists of a total of 83 scenes. Since we are trying to classify image patches at the level of camera model rather than instance level, we only consider camera models with more than one instance available. This leads us to a total of 18 camera models (as Nikon D70 and D70s basically differ only in their on-device screen) and nearly $15\,000$ shots.

In order to evaluate our method we divide the dataset consisting of 18 camera models into 3 sets, namely training $\mathcal{D}_\text{T}$, validation $\mathcal{D}_\text{V}$ and evaluation $\mathcal{D}_\text{E}$. $\mathcal{D}_\text{T}$ is again split into two equal sets $\mathcal{D}_\text{T}^\text{cam}$ and $\mathcal{D}_\text{T}^\text{ip}$. $\mathcal{D}_\text{T}^\text{cam}$ is used for training the parameters of $\mathcal{M}_\text{cam}$, whereas $\mathcal{D}_\text{T}^\text{ip}$ is used for training $\mathcal{M}_\text{ip}$. $\mathcal{D}_\text{V}$ is used to decide how many epochs to use during training to avoid overfitting. Finally, $\mathcal{D}_\text{E}$ is used for evaluating the trained network on a disjoint set of images in a fair way.

While training a CNN, it is very important to avoid overfitting the data. In our dataset we have images of different scenes taken by different cameras, and our goal is to learn information about camera model from an image. As $\mathcal{D}_\text{V}$ is used to avoid overfitting, it is important that $\mathcal{D}_\text{T}$ and $\mathcal{D}_\text{V}$ are sufficiently different from each other. It is also important that we test on data that has variation with respect to the training data. In order to achieve these goals, we do the following:

- Images for $\mathcal{D}_\text{E}$ are selected from a single instance per camera and a set of 11 scenes.
- Images for $\mathcal{D}_\text{T}$ are selected from the additional camera instances and 63 different scenes.
- Images in $\mathcal{D}_\text{V}$ are selected from the same camera instances used for $\mathcal{D}_\text{T}$, but from the remaining 10 scenes. This makes sets $\mathcal{D}_\text{V}$ and $\mathcal{D}_\text{T}$ disjoint with respect to scenes, leading to robust training.

For training, validation and testing $K = 300$ non overlapping color patches of size 64×64 are extracted from each image. the resulting dataset $\mathcal{D}_\text{T}^\text{cam}$ contains more than $500\,000$ patches split into 18 classes. $\mathcal{D}_\text{T}^\text{ip}$ is reduced to $90\,000$ patches to balance reliable and non-reliable image patches according to $\mathcal{M}_\text{cam}$ classification results. Finally, $\mathcal{D}_\text{V}$ and $\mathcal{D}_\text{E}$ are composed by more than $700\,000$ and $800\,000$ patches, respectively.

4.2. Training Strategies

Given that the proposed approach builds upon a pre-trained network (i.e., $\mathcal{M}_\text{cam}$), we propose a two-tiered transfer learning-based approach denoted as *Transfer*. For the sake of comparison, we also test two additional strategies, namely *Scratch* and *Pre-Trained*. In the following we report details about each strategy.

Scratch. This training strategy is the most simple one. It consists in training the whole two-class architecture $\mathcal{M}$ using only $\mathcal{D}_\text{T}^\text{ip}$ for training and $\mathcal{D}_\text{V}$ for validation. This can be considered as a baseline training strategy. We use Adam optimizer with default parameters as suggested in [31] and batch size 128. Loss is set to binary-crossentropy.

Pre-Trained. This strategy takes advantage of the possibility of using a pre-trained $\mathcal{M}_\text{cam}$. In this case, we train

$\mathcal{M}_{\text{cam}}$ for camera model attribution using softmax normalization on its output. Training is carried out on $\mathcal{D}_{\text{T}}^{\text{cam}}$ and validation on $\mathcal{D}_{\text{V}}$. Once $\mathcal{M}_{\text{cam}}$ has been trained, we freeze its weights, and train the rest of the architecture $\mathcal{M}_{\text{ip}}$ as a two-class classifier (i.e., reliable vs. non-reliable patches) using $\mathcal{D}_{\text{T}}^{\text{ip}}$ and $\mathcal{D}_{\text{V}}$. Optimization during both training steps is carried out using Adam optimizer with default values and batches of 128 patches. We select categorical-crossentropy as our loss function.

Transfer. This two-tiered training strategy is meant to fully exploit the transfer learning capability of the proposed architecture. The first step consists in training $\mathcal{M}_{\text{cam}}$ for camera model attribution using softmax normalization on its output and $\mathcal{D}_{\text{T}}^{\text{cam}}$ and $\mathcal{D}_{\text{V}}$ as datasets. This training step is optimized using Adam with default parameters, 128 patches per batch, and categorical-crossentropy as loss function.

The second step of $\mathcal{M}$ training consists in freezing all the convolutional layers of $\mathcal{M}_{\text{cam}}$, and continue training all the inner product layers of both $\mathcal{M}_{\text{cam}}$ and $\mathcal{M}_{\text{ip}}$ using the datasets $\mathcal{D}_{\text{T}}^{\text{ip}}$ and $\mathcal{D}_{\text{V}}$. This enables to jointly learn the weights of the classifier $\mathcal{M}_{\text{ip}}$, and tailor feature extraction procedure in the last layers of $\mathcal{M}_{\text{cam}}$ (i.e., *ip1* and *ip2*) to the classification task. For this step we use binary-crossentropy as loss, and stochastic gradient descent (SGD) with oscillating learning rate between $5 \cdot 10^{-5}$ and $15 \cdot 10^{-5}$ as optimizer. This choice is motivated by preliminary studies carried out in [32], and experimentally confirmed in our analysis.

5. Discussion

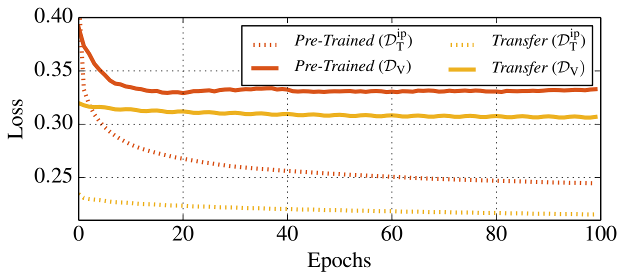
In this section we discuss the experimental results. First we show the capability of the proposed approach to distinguish between patches that contain camera model information and patches that are not suitable for this task. Then, we show how it is possible to improve camera model identification using the proposed approach.

5.1. Patch Reliability

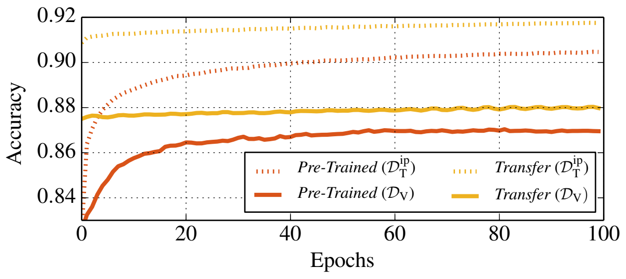
In order to validate the patch reliability estimation we perform a set of tests.

CNN Architecture. The first set of experiments has been devoted to the choice of a network architecture for $\mathcal{M}_{\text{ip}}$. To this purpose, we trained a set of different architectures for 15 epochs using the *Pre-Trained* strategy. As architectures we selected all possible combinations of up to six inner product layers (with ReLU activation) composed by 32, 64 or 128 neurons each. The last layer is always set to two neurons followed by softmax. From this experiment, we selected the model $\mathcal{M}_{\text{ip}}$ with the highest validation accuracy for each tested amount of layers, which are:

- $\mathcal{M}_{\text{ip}}^2$ composed by two inner product layers with 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^3$ composed by three inner product layers with 64, 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^4$ composed by four inner product layers with 64, 32, 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^5$ composed by five inner product layers with 64, 32, 64, 128 and 2 neurons, respectively.
- $\mathcal{M}_{\text{ip}}^6$ composed by six inner product layers with 64, 32, 32, 64, 64 and 2 neurons, respectively.



(a) Loss



(b) Accuracy

Figure 5. Loss and accuracy curves on training ($\mathcal{D}_{\text{T}}^{\text{ip}}$) and validation ($\mathcal{D}_{\text{V}}$) datasets using *Pre-Trained* and *Transfer* strategies on $\mathcal{M}_{\text{ip}}^4$.

Training Strategy. In order to validate the proposed two-tiered *Transfer* training strategy, we trained the five selected models with the *Scratch*, *Pre-Trained* and *Transfer* strategies. Examples of training (on $\mathcal{D}_{\text{T}}^{\text{ip}}$) and validation (on $\mathcal{D}_{\text{V}}$) loss curves for the *Pre-Trained* and *Transfer* strategies on $\mathcal{M}_{\text{ip}}^4$ are shown in Figure 5(a). It is possible to notice that the chosen optimizers enable a smooth loss decrease over several epochs. Moreover, the *Transfer* strategy provides a lower loss on both training and validation data, thus yielding better results compared to *Pre-Trained*. Similar conclusions can be drawn from the accuracy curves presented in Figure 5(b). We do not display curves for the *Scratch* strategy, as it is always worse than both the *Pre-Trained* and *Transfer* strategies. This was expected, as the amount of used training data in $\mathcal{D}_{\text{T}}^{\text{ip}}$ is probably not enough to learn all parameters of $\mathcal{M}$. Therefore, starting from a pre-trained $\mathcal{M}_{\text{cam}}$ becomes necessary.

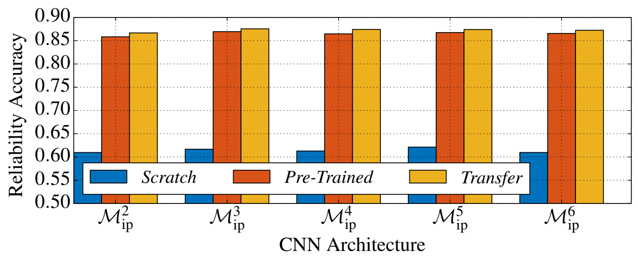


Figure 6. Patch reliability estimation accuracy. The *Transfer* strategy yields the most accurate results for every architecture.

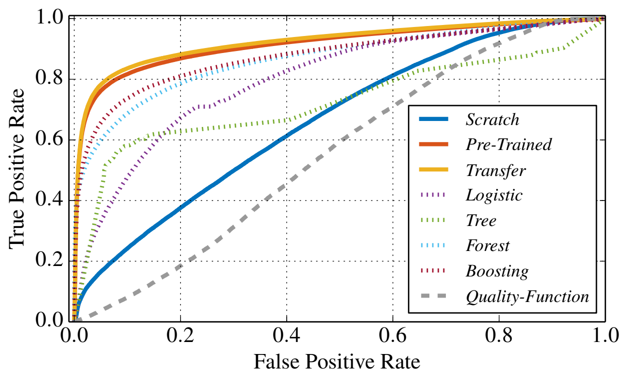


Figure 7. ROC curves on reliable patch detection. The proposed methods are represented by means of solid lines. Other baselines are dotted or dashed. Our *Transfer* strategy achieves the best overall performance.

Figure 6 shows the reliability patch estimation accuracy for all models from $\mathcal{M}_{\text{ip}}^2$ to $\mathcal{M}_{\text{ip}}^6$, trained with all three training strategies and tested on the evaluation dataset $\mathcal{D}_{\text{E}}$. For each architecture, we selected the model with highest validation accuracy achieved over 100 epochs. These results further confirm that the *Scratch* strategy is not a viable solution for this problem. The *Transfer* strategy is the best choice for each network, achieving around 86% accuracy in detecting reliable patches. In other words, 86% of the selected patches will be correctly attributed to their camera, whereas only 14% of them will be wrongly classified. In the ideal scenario of errors uniformly spread across all models and images, this means we could use a majority voting strategy to further increase accuracy at the image level.

From Figure 6, it is also possible to notice that increasing $\mathcal{M}_{\text{ip}}$ depth does not increase accuracy. Therefore, from this point on, we only consider architecture $\mathcal{M}_{\text{ip}}^4$ as a good trade-off.

Baselines Comparison. In order to further validate the proposed approach, we also considered two possible baseline solutions.

The first one consists in using other kinds of supervised classifiers exploiting the 18-element vector returned by $\mathcal{M}_{\text{cam}}$ as feature. To this purpose, we trained a logistic regressor (*Logistic*), a decision tree (*Tree*), a random forest (*Forest*) and a gradient boosting classifier (*Boosting*). For each method, we applied z-score feature normalization and we selected the model achieving highest validation accuracy on $\mathcal{D}_{\text{V}}$ after a parameter grid-search training on $\mathcal{D}_{\text{T}}^{\text{ip}}$. Accuracy results on patch reliability on evaluation set $\mathcal{D}_{\text{E}}$ were 70.7%, 73.9%, 78.6% and 81.8%, respectively. None of them approaches the 86% of the proposed solution.

The second baseline solution we tested is the quality-function presented in [22] (*Quality-Function*). This function is computed for each patch and returns a value between zero and one indicating whether the patch is suitable for training $\mathcal{M}_{\text{cam}}$. Although Quality-Function was not intended to work as test reliability indicator, we decided that a comparison was necessary for completeness. To this purpose, Figure 7 shows receiver operating characteristic (ROC) curves obtained thresholding our reliability likelihood estimation g_k, the soft output of the other classifiers (i.e., logistic regressor, decision tree, etc.), and the quality-function returned value [22]. As expected, the use of the quality-function presented in [22] provides less accurate results. Conversely, the proposed method achieves better performance than all other classifiers when trained according to *Transfer* strategy.

Table 1. Camera model attribution accuracy using selected reliable patches from test dataset only. Using *Transfer* strategy (bold), the amount of selected patches in $\mathcal{D}_{\text{E}}$ is always greater than 77% of $|\mathcal{D}_{\text{E}}|$. Accuracy improvement over random patch selection is greater than 8%.

$\mathcal{M}_{\text{ip}}$	Strategy	Patches	Accuracy	Acc. Delta
$\mathcal{M}_{\text{ip}}^2$	*Scratch*	553 475	0.9009	0.0342
	Pre-Trained	618 958	0.9478	0.0811
	Transfer	637 135	**0.9513**	**0.0845**
$\mathcal{M}_{\text{ip}}^3$	*Scratch*	518 228	0.9041	0.0374
	Pre-Trained	626 767	0.9520	0.0853
	Transfer	641 808	**0.9556**	**0.0888**
$\mathcal{M}_{\text{ip}}^4$	*Scratch*	562 897	0.8963	0.0296
	Pre-Trained	649 515	0.9499	0.0832
	Transfer	647 998	**0.9532**	**0.0865**
$\mathcal{M}_{\text{ip}}^5$	*Scratch*	511 425	0.9045	0.0378
	Pre-Trained	648 665	0.9529	0.0862
	Transfer	651 508	**0.9530**	**0.0863**
$\mathcal{M}_{\text{ip}}^6$	*Scratch*	517 386	0.9035	0.0367
	Pre-Trained	651 405	0.9501	0.0834
	Transfer	652 308	**0.9531**	**0.0864**

5.2. Camera Model Attribution

After validating the possibility of selecting reliable patches with the proposed method, we tested the effect of this solution on camera model attribution. To this purpose, we report in Table 1 the evaluation set results for the five investigated $\mathcal{M}_{\text{ip}}$ models and the three training strategies

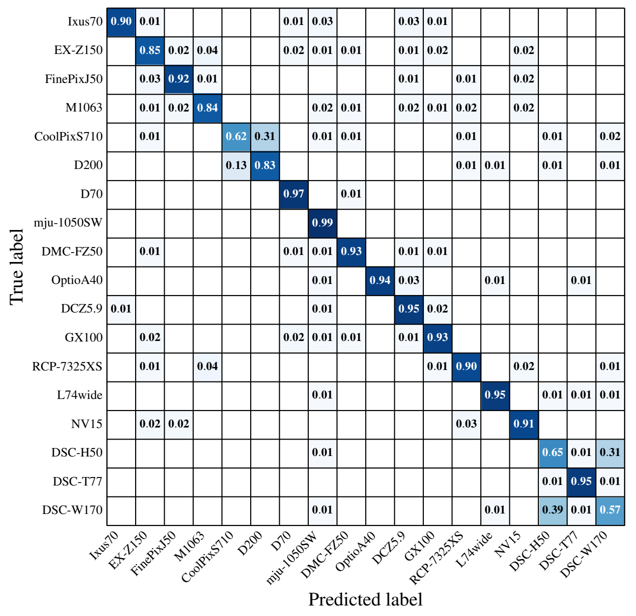


Figure 8. Camera model attribution confusion matrix obtained with $\mathcal{M}_{\text{cam}}$ on $\mathcal{D}_{\text{E}}$ without patch selection.

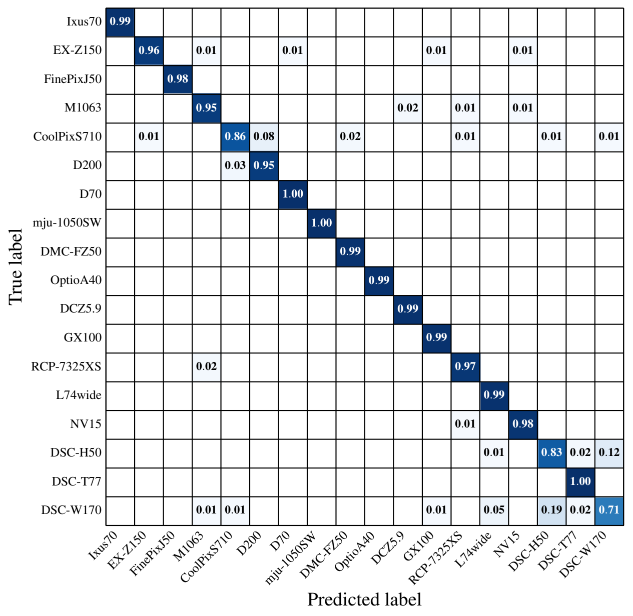


Figure 9. Camera model attribution confusion matrix obtained with $\mathcal{M}_{\text{cam}}$ on $\mathcal{D}_{\text{E}}$ using patches selected with $\mathcal{M}_{\text{ip}}^{4}$.

when a single patch is used for camera model attribution. We do so in terms of:

1. *Patches*, i.e., the number of estimated reliable patches.
2. *Accuracy*, i.e., the average achieved camera model attribution result.
3. *Accuracy Delta*, i.e., the accuracy increment with respect to not using patch selection but randomly picking them (i.e., [20]).

These results highlight that it is possible to improve camera model attribution by more than 8%.

Figure 8 shows confusion matrix results using $\mathcal{M}_{\text{cam}}$ (i.e., the output of the network proposed in [20]) on evaluation data $\mathcal{D}_{\text{E}}$ randomly selecting patches. The average accuracy per patch is 87%. Figure 9 shows the same results, evaluating only patches considered reliable using $\mathcal{M}_{\text{ip}}^{4}$. In this scenario, accuracy increases to more than 95%. By comparing the two figures, it is possible to notice that many spurious classifications outside of the confusion matrix diagonal are corrected by the use of reliable patches only.

6. Conclusions

In this paper we presented a method for reliability patch estimation for camera model attribution. This means being able to estimate the likelihood that an image patch will be correctly attributed to the camera model used to acquire the image from which the patch comes from.

The proposed solution is based on concatenating a pretrained CNN for patch-wise camera model attribution with a dense network that acts as a binary classifier. Exploiting transfer learning techniques and a two-tiered training strategy, it is possible to achieve 86% of accuracy in patch reliability estimation. Moreover, by running camera model attribution on single selected patches, camera attribution accuracy increases by more than 8%.

In addition to the impact on camera model attribution, the proposed method returns a reliability mask that highlights which image regions are considered reliable in terms of camera attribution. This could be useful in the future to better understand which image details are more important to camera model attribution CNNs. Moreover, it could be paired with splicing localization algorithms based on camera model traces to possibly opt-out unreliable regions from the analysis.

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] 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. 26, pp. 1–42, 2011.

[2] A. Piva, “An overview on image forensics,” *ISRN Signal Processing*, vol. 2013, 2013.

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

[4] 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.

[5] 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.

[6] 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.

[7] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, “Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 5302–5306, October 2014, Paris, France.

[8] J. Lukáš, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, June 2006.

[9] M. Goljan and J. Fridrich, “Camera identification from cropped and scaled images,” *Proceedings of the SPIE Electronic Imaging*, vol. 6819, January 2008, San Jose, CA.

[10] 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. 69–72, September 2005, Genova, Italy.

[11] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian, “Forensic detection of median filtering in digital images,” *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 89–94, July 2010, Singapore.

[12] X. Zhao and M. C. Stamm, “Computationally efficient demosaicing filter estimation for forensic camera model identification,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 151–155, September 2016, Phoenix, AZ.

[13] 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.

[14] S.-H. Chen and C.-T. Hsu, “Source camera identification based on camera gain histogram,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 429–432, September 2007, San Antonio, TX.

[15] 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, January 2014.

[16] C. Chen and M. C. Stamm, “Camera model identification framework using an ensemble of demosaicing features,” *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pp. 1–6, November 2015, Rome, Italy.

[17] A. Tuama, F. Comby, and M. Chaumont, “Source camera model identification using features from contaminated sensor noise,” *Proceedings of the International Workshop on Digital-Forensics and Watermarking*, pp. 83–93, October 2016, Tokyo, Japan.

[18] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “Evaluation of residual-based local features for camera model identification,” *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Eds. Cham, Switzerland: Springer International Publishing, 2015, pp. 11–18.

[19] A. Tuama, F. Comby, and M. Chaumont, “Camera model identification based on machine learning approach with high order statistics features,” *Proceedings of the IEEE European Signal Processing Conference*, pp. 1183–1187, August 2016, Budapest, Hungary.

[20] 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.

[21] 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,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1855–1864, July 2017, Honolulu, HI.

[22] 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.

[23] 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, December 2010.

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

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

[26] 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.

[27] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich, “Jpeg-phase-aware convolutional neural network for steganalysis

of jpeg images," *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pp. 75–84, June 2017, Philadelphia, PA.

[28] J. Chen, X. Kang, Y. Liu, 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.

[29] 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.

[30] 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.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference on Learning Representations*, vol. arXiv, no. 1412.6980, May 2015, San Diego, CA.

[32] L. N. Smith, "Cyclical learning rates for training neural networks," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 464–472, March 2017, Santa Rosa, CA.