IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2022–11–06.
Submitted on 2022–09–28, accepted on 2022–09–28.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.424

Forensic Similarity for Source Camera Model Comparison

Marina Gardella[1], Pablo Musé[2]

[1]Université Paris-Saclay, Université Paris Cité, ENS Paris-Saclay, CNRS, SSA, INSERM, Centre Borelli, Gif-sur-Yvette, France
[2]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay
marina.gardella@ens-paris-saclay.fr, pmuse@fing.edu.uy

Communicated by Jean-Michel Morel *Demo edited by* Marina Gardella

Abstract

In the article "Forensic Similarity for Digital Images", O. Mayer and M. C. Stamm introduce the forensic similarity approach, which aims at determining whether two image patches contain the same forensic traces or not. The proposed method is based on a feed-forward neural network which consists of two modules: a feature extraction module using a pair of CNNs in a siamese configuration, and a three-layer neural network that maps the extracted features into a similarity score. In this article, we explore the use of the forensic similarity score for source camera model comparison, as one of the possible applications of such an approach suggested by Mayer and Stamm.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.txt` file of the archive. The original implementation of the method is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: image forensics; source camera; camera model

1 Introduction

Providing information about the camera with which an image was acquired can be crucial for different forensic applications. Indeed, it can provide clues to track pornographic content, to check for copyright infringement, and to verify the consistency of a database. There are different approaches that aim at describing the source device of a given image. Some of them try to identify the particular device with which the image was taken [2, 6], while others focus on identifying the brand or model of the source camera [10].

[1] https://doi.org/10.5201/ipol.2022.424
[2] http://omayer.gitlab.io/forensicsimilarity/

Classic methods tackle this problem by searching for device traces. These traces include sensor pattern noise [6], lens distortions [2], demosaicing artefacts, white balance traces [3] and compression. Some of these features, such as the PRNU pattern [6] or radial distortion [2], are device-specific and can lead to accurate device identification. Others are shared by different devices of the same model or brand, and can therefore provide information about the device model rather than identifying a particular source camera [10].

In this article we explore how the forensic similarity approach introduced in [8] can be applied for source camera model comparison. This approach aims at determining whether two image patches share the same forensic traces or not. Forensic traces are signals embedded in the image during the image formation process. Indeed, from the moment the light hits the camera sensors until the final digital file is delivered, the image undergoes several operations such as demosaicing, denoising, gamma correction, white balance and compression. Each of these operations leaves specific artifacts in the final image. Images acquired with devices from the same model are expected to exhibit the same similar forensic traces, while devices from different models are expected to produce different traces.

2 Problem Formulation

The problem can be stated as follows: Given two image patches, we want to assign a score of 0 to the pair of patches if they have different forensic traces, and a score of 1 if they share the same forensic traces. That is, we search for a map $C : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ is the space of all image patches, such that

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } X_1, X_2 \text{ have different forensic traces,} \\ 1 & \text{if } X_1, X_2 \text{ have the same forensic traces.} \end{cases}$$

This problem can be tackled in three steps. First, a suitable set of N features capturing the forensic information is extracted from each patch, by means of a feature extractor $f : \mathcal{X} \to \mathbb{R}^N$. The resulting feature vectors are then compared based on a similarity function $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$. Finally, the similarity measure is compared to a threshold τ so as to obtain a binary output. The map C can be then written as

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } S(f(X_1), f(X_2)) \leq \tau, \\ 1 & \text{if } S(f(X_1), f(X_2)) \geq \tau. \end{cases}$$

This way, the problem of finding C amounts to find two functions $f : \mathcal{X} \to \mathbb{R}^N$ and $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$ such that $S(f(X_1), f(X_2))$ is as close to 0 as possible whenever X_1 and X_2 have different forensic traces, and as close as possible to 1 whenever the two image patches share the same traces. Figure 2 shows the system overview.

3 Method

In [8], Mayer and Stamm propose to design both the feature extractor function f and the similarity function S based on a learning strategy. In this section we specify the architecture as well as the training strategies developed in their work.

3.1 Architecture

The feature extractor is based on the MISLnet architecture [1] and is depicted in Figure 1. Namely, it consists of 5 convolutional blocks and 2 fully connected layers. Each of the convolutional layers, except for the first one, is followed by batch normalization, hyperbolic tangent activation and max-pooling. The size of the convolutional filters used at each layer are $5 \times 5 \times 3 \times 6$, $7 \times 7 \times 6 \times 96$, $5 \times 5 \times 96 \times 64$, $5 \times 5 \times 64 \times 64$ and $1 \times 1 \times 64 \times 128$ respectively. A stride equal to 1 is used in all the layers except for the second one, where the stride is set to 2. The last convolutional layer, which uses 1×1 kernels, can be regarded as a learned cross-feature maps association. The max-pooling operation is performed using 3×3 kernels. The two fully connected layers that follow the convolutional blocks consist both of 200 neurons with hyperbolic tangent activation.

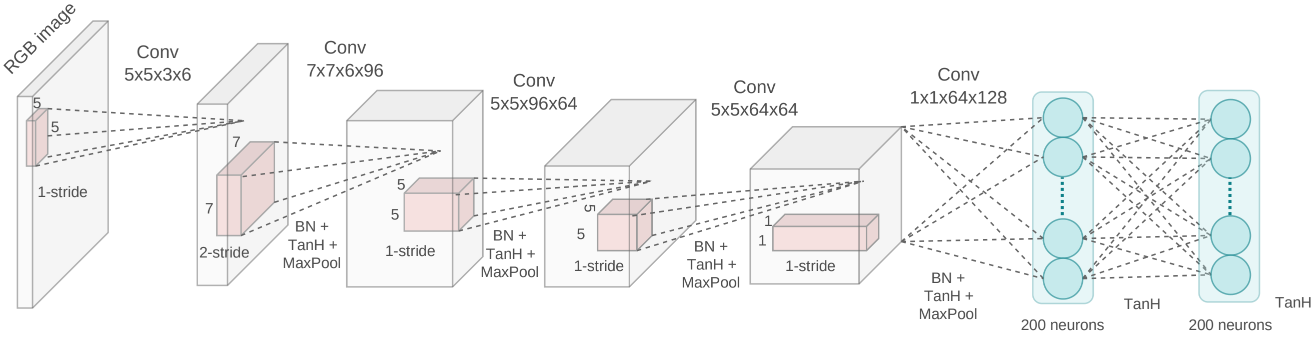


Figure 1: Feature extractor architecture.

Two such feature extractor networks, in siamese configuration with weight-sharing, are used to process in parallel the two patches, producing a feature vector for each patch. Then, a similarity network takes both feature vectors as input and computes their similarity score. Figure 2 shows the complete system overview. The first layer of the similarity network, consisting of 2048 neurons with ReLu activation, maps each feature vector into a new feature space. The authors use a hard-sharing siamese configuration for this first layer. Then, a new feature vector is constructed by concatenating both feature vectors and their element-wise multiplication. This vector feeds another fully-connected layer with 64 neurons. Finally, a single-neuron layer with sigmoid activation takes the resulting 64-dimensional vector and produces the similarity score associated to the pair of input patches[3].

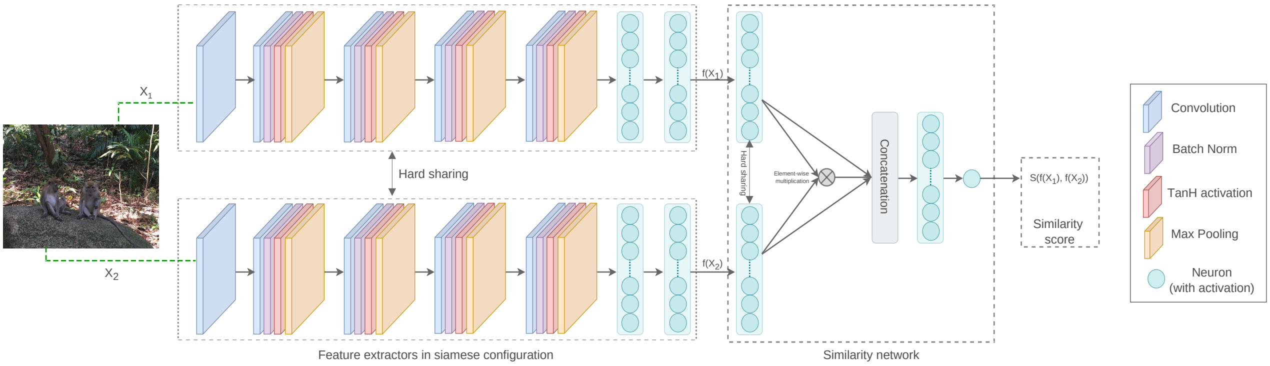


Figure 2: System overview. The first module consists of a pair of feature extractor networks in siamese configuration with weight-sharing. It takes two image patches and computes its corresponding feature vectors. These vectors are then compared by means of the second module (the similarity network), which computes a similarity score associated to the pair of image patches.

[3] In practice, the authors use two output units with softmax activation, one indicating "similar" traces and the other one indicating "different" traces. The observed output for evaluation is the one corresponding to "similar".

3.2 Dataset

Mayer and Stamm collected a dataset of 47,785 images from 95 different camera models. Among them, 26 camera models come from the Dresden dataset [4] while the rest are from the authors' database. This dataset is divided into three disjoint subsets. Subset 1 consists of 50 camera models selected randomly from those having at least 40,000 non overlapping 256 × 256 patches. Subset 2 comprises 30 camera models from the remaining ones having at least 25,000 non-overlapping patches. Subset 3 comprises the remaining 15 camera models. The complete list of camera models is given in Table 1. The camera models in blue were collected from the Dresden dataset [4].

Subsets 1 and 2 are used for training (see Section 3.3) while Subset 3 is used for evaluation, which we will not cover here. The interested reader is referred to the original paper [8].

Subset 1			
Apple iPhone 4	Agfa Sensor530s	Apple iPhone 4s	Canon EOS SL1
Apple iPhone 5	Canon PC1730	Apple iPhone 5s	Canon A580
Apple iPhone 6	Canon ELPH 160	Apple iPhone 6+	Canon S100
Apple iPhone 6s	Canon SX530 HS	Canon SX420 IS	Canon SX610 HS
Casio EX-Z150	Fujifilm S8600	Huawei Honor 5x	LG G2
LG G3	LG Nexus 5x	Motorola Maxx	Motorola Turbo
Motorola X	Motorola XT1060	Nikon S33	Nikon S7000
Nikon S710	Nikon D200	Nikon D3200	Nikon D7100
Panasonic DMC-FZ50	Panasonic FZ200	Pentax K-7	Pentax OptioA40
Praktica DCZ5.9	Ricoh GX100	Rollei RCP-7325XS	Samsung Note4
Samsung S2	Samsung S4	Samsung L74wide	Samsung NV15
Sony DSC-H300	Sony DSC-W800	Sony DSC-WX350	Sony DSC-H50
Sony DSC-T77	Sony NEX-5TL		
Subset 2			
Apple iPad Air 2	Blackberry Leap	Apple iPhone 5c	Canon Ixus70
Agfa DC-733s	Canon PC1234	Agfa DC-830i	Canon G10
Canon SX400 IS	Canon T4i	Fujifilm XP80	Fujifilm J50
HTC One M7	Kodak C813	Kodak M1063	LG Nexus 5
Motorola Nexus 6	Nikon D70	Nikon D7000	Nokia Lumia 920
Olympus TG-860	Panasonic TS30	Pentax OptioW60	Samsung Note3
Samsung Note5	Samsung S3	Samsung S5	Samsung S7
Sony A6000	Sony DSC-W170		
Subset 3			
Agfa DC-504	Canon Ixus55	Agfa Sensor505x	Canon A640
Canon Rebel T3i	LG Optimus L90	LG Realm	Nikon S3700
Nikon D3000	Olympus 1050SW	Samsung Lite	Samsung Nexus
Samsung Note2	Samsung S6	EdgeSony DSC-T70	

Table 1: Camera models used for training and evaluation. Camera models in blue come from the Dresden dataset [4]. Subset 1 is used during the first training phase. Subset 1 and 2 are used during the second training phase. Subset 3 is used for evaluation.

3.3 Training Procedure

The system is trained in two phases. In the first phase, the feature extractor is trained by adding a fully connected layer with softmax activation. The feature extractor is trained as a source camera classifier, using image patches with associated labels corresponding to their source camera model. Research indicates that the deep features associated to camera model classification provide a good starting point for several forensics tasks [7]. The authors use a cross-entropy loss, optimized using stochastic gradient descent for 30 epochs, with batches of 50 images. Initially the learning rate is set to 0.001, and is halved every three epochs. The authors train two versions of the feature extractor: one using 128 × 128 patches and another one using 256 × 256 patches.

During the second training phase, the similarity network is trained to target a specific task. Here, the task is to determine if two image patches come from the same camera model, but it could be to determine a specific editing operation or a specific parameter given an editing operation. The labels in the training dataset are assigned accordingly: 0 indicates that the patches come from the same camera model, and a 1 indicates they come from different ones. During this phase, the weights of the feature extractor are fine-tuned, i.e. they are also updated to better fit the particular task. The similarity network is trained using stochastic gradient descent with cross-entropy loss for 30 epochs. The learning rate is initialized to 0.005, and then is halved every three epochs.

During the first training phase, the feature extractor is trained using 40,000 randomly sampled image patches from each of the camera models in Subset 1, giving a total of 2,000,000 image patches. During the second training phase, the similarity network is trained and the feature extractor weights are updated using a training dataset of pairs of image patches. This dataset is constructed using camera models in Subset 1 and Subset 2.

4 Demo

The goal of the demo is to determine if a pair of images have been captured by the same camera model or not. It takes as input three images: a reference image, and two test images that will be compared to the reference one. To perform image-wise comparison built upon the patch comparison provided by the forensic similarity approach, the user is required to choose the number of randomly chosen patch-to-patch comparisons (ranging from 100 to 700) to be considered. The user can also decide if these patches are taken with overlap (half of the patch size) or not. Finally, the user can decide the patch size, being 128 and 256 the two available options.

The output of the demo is an interactive histogram showing the forensic scores obtained for each patch-to-patch comparison, for both image comparisons. By moving the mouse over the histogram, the user can recover the bins bounds as well as the count that corresponds to each bin. The user can also zoom in different sections of the histogram to better visualize the results.

5 Experiments

In this section we show several experiments conducted using the demo. The images used come from the Vision dataset [9] and the Forchheim dataset [5]. Notice that none of these datasets were used for training: all the images used for these experiments are new to the network.

To assess the performance of the forensic similarity approach, we designed three different experiments with different challenging scenarios. In the first experiment we test the approach using images coming from camera models that were used for training. In the next experiment we compare images coming from known models to images coming from unknown ones. Finally, we test the forensic similarity approach on pairs of camera models that are unknown to the network.

All the experiments were conducted using the default parameters values. Namely, the number of patch-to-patch comparisons is 300, the size of the patches is set to 256 and the patches are taken without overlap.

5.1 Known Camera Models

Figure 3 shows the results obtained when applying the forensic similarity approach for source camera comparison for images taken with camera models that are known to the network. Namely, for this experiment we use camera models that are included in Subset 1 and Subset 2 (see Table 1).

We observe that, when faced to known camera models, the similarity scores between images coming from the same camera model concentrate around 1. Furthermore, the similarity scores obtained when comparing images coming from different camera models concentrate around 0, except for the case in which iPhone 6 is compared to iPhone 6s. This might be mainly due to the fact that these two devices share similar processing pipelines. In this case we observe that the similarity scores wrongly concentrate around 1. However, it can be also noticed that the non-matching histogram exhibits a thicker tail than the matching one.

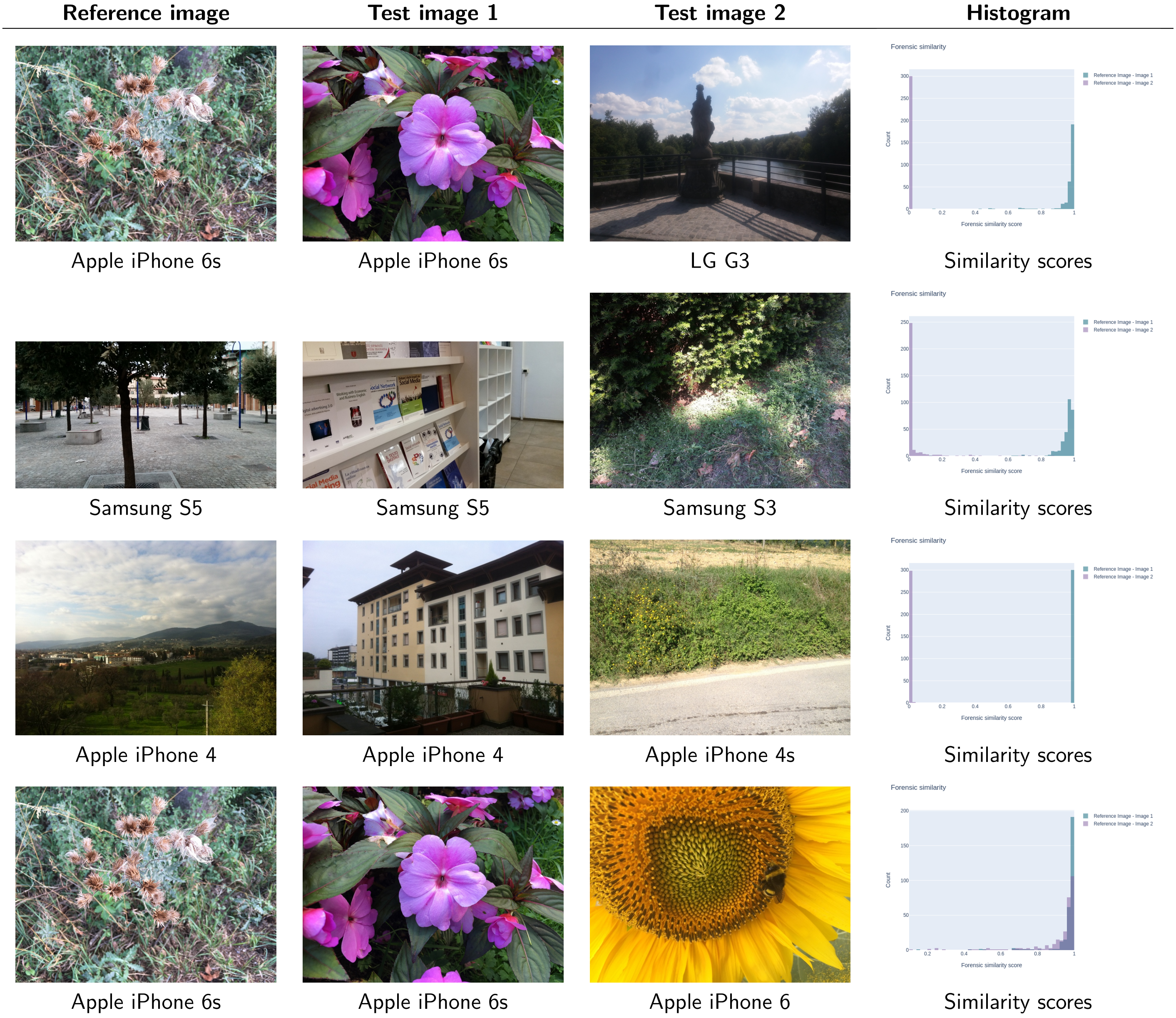


Figure 3: Results of the forensic similarity approach applied to source camera comparison when images under test come from camera models used during training.

5.2 Known and Unknown Camera Models

Figure 5 shows the results obtained when applying the forensic similarity approach for source camera comparison, to test images taken with camera models that are known to the network against images from camera models that were not used for training. Namely, for this experiment we test camera models that are included in Subset 1 and Subset 2 (see Table 1) against camera models that are not part of them.

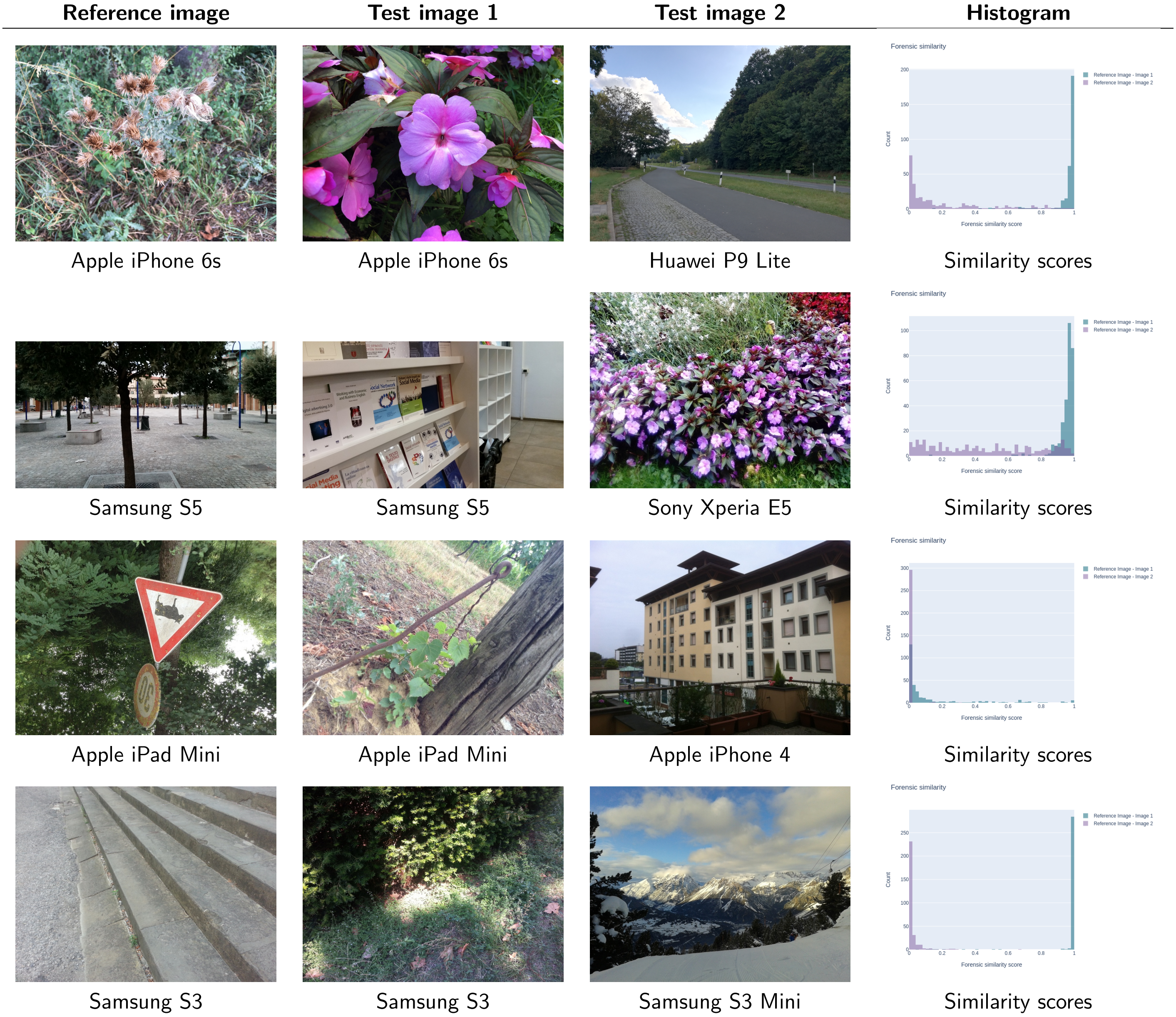


Figure 4: Results of the forensic similarity approach applied to source camera comparison, when comparing images coming from camera models used during training to images from camera models unknown to the network.

Under this setting the results are more heterogeneous. The network is able to distinguish images coming from iPhone 6s from those coming from Huawei P9 Lite as well as those coming from Samsung S3 Mini and Samsung S3. When comparing iPad Mini to iPhone 4, the network also delivers similarity scores close to 0. However, the network is not able to identify two images taken with iPad Mini as having similar forensic traces. On the other hand, the results obtained when comparing Samsung S5 and Sony Xperia E5, the histogram shows that the patch-to-patch similarity scores seem uniformly distributed over the $[0, 1]$ interval, therefore preventing from taking any conclusion about their forensic similarity.

5.3 Unknown Camera Models

Figure 5 shows the results obtained when applying the forensic similarity approach for source camera comparison to test images taken with camera models that are unknown to the network. Namely, in this experiment we consider camera models that are not included in Subset 1 or Subset 2 (see Table 1).

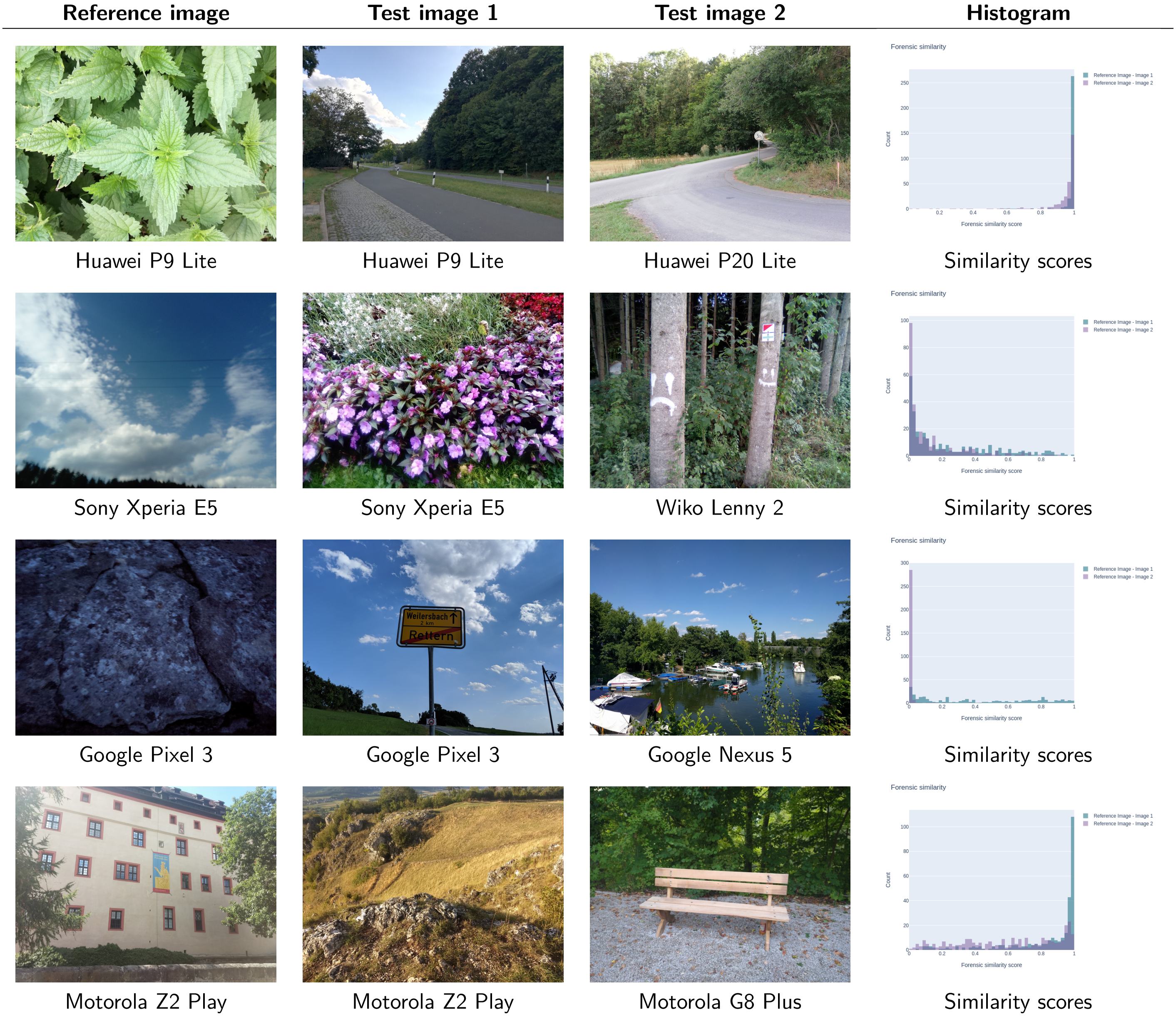


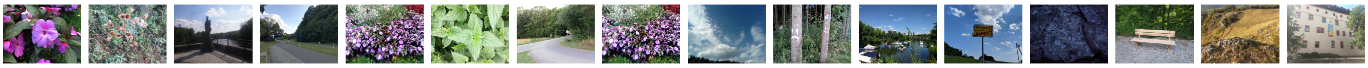
Figure 5: Results of the forensic similarity approach applied to source camera comparison, when comparing images coming from camera models that were not used for training.

In this challenging scenario, we observe degraded results with respect to the previous experiments. Indeed, for camera models showing good results for matching images (Motorola Z2 Play and Huawei P9 Lite), the mismatching results are incorrect. On the contrary, the devices delivering good results when compared to a different camera model (Google Pixel 3 and Sony Xperia E5), fail at identifying matching images.

Acknowledgment

This work has received funding by the Paris Region Ph.D. grant from Région Île-de-France, the ANR-DGA challenge DEFALS (ANR-16-DEFA-0004) and the European Union under the Horizon Europe vera.ai project, Grant Agreement number 101070093.

Image Credits



Forchheim Dataset [5]

Vision Dataset [9]

References

[1] B. Bayar and M.C. Stamm, *Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection*, IEEE Transactions on Information Forensics and Security, 13 (2018), pp. 2691–2706. http://dx.doi.org/10.1109/TIFS.2018.2825953.

[2] K.S. Choi, E.Y. Lam, and K.K.Y. Wong, *Automatic source camera identification using the intrinsic lens radial distortion*, Optics Express, 14 (2006), pp. 11551–11565. https://doi.org/10.1364/OE.14.011551.

[3] Z. Deng, A. Gijsenij, and J. Zhang, *Source camera identification using auto-white balance approximation*, in International Conference on Computer Vision (ICCV), 2011, pp. 57–64. https://doi.org/10.1109/ICCV.2011.6126225.

[4] T. Gloe and R. Böhme, *The 'Dresden Image Database' for Benchmarking Digital Image Forensics*, in ACM Symposium on Applied Computing, Association for Computing Machinery, 2010, p. 1584–1590. https://doi.org/10.1145/1774088.1774427.

[5] B. Hadwiger and C. Riess, *The Forchheim Image Database for Camera Identification in the Wild*, in International Conference on Pattern Recognition Workshops, 2020. https://doi.org/10.1007/978-3-030-68780-9_40.

[6] J. Lukás, J. Fridrich, and M. Goljan, *Digital camera identification from sensor pattern noise*, IEEE Transactions on Information Forensics and Security, 1 (2006), pp. 205 – 214. https://doi.org/10.1109/TIFS.2006.873602.

[7] O. Mayer, B. Bayar, and M.C. Stamm, *Learning unified deep-features for multiple forensic tasks*, in ACM Workshop on Information Hiding and Multimedia Security, Association for Computing Machinery, 2018, p. 79–84. https://doi.org/10.1145/3206004.3206022.

[8] O. Mayer and M. C. Stamm, *Forensic similarity for digital images*, IEEE Transactions on Information Forensics and Security, (2019). https://doi.org/10.1109/TIFS.2019.2924552.

[9] D. Shullani, M. Fontani, M. Iuliani, O. Alshaya, and A. Piva, *VISION: a video and image dataset for source identification*, EURASIP Journal on Information Security, (2017), p. 15. https://doi.org/10.1186/s13635-017-0067-2.

[10] T.H. Thai, F. Retraint, and R. Cogranne, *Camera Model Identification Based on DCT Coefficient Statistics*, Digital Signal Processing, 40 (2015), p. 88–100. https://doi.org/10.1016/j.dsp.2015.01.002.