Source Camera Identification using Noise and Deep Learning

Omar Jarkas
University of Queensland

Abstract

Remote source camera identification is a passive technique to identify and authenticate images. The technique relies on distinguishing cameras based on unique image features. Since cameras produce unique noise, we can exploit such noise to fingerprint them. By fingerprinting cameras, we can provide a practical means of feed authentication. Existing techniques lack scalability and automation capabilities that facilitate live solution. In this work, we present a novel end-to-end source camera identification solution. The technique relies on deep models, Convolutional Neural Networks, and specific unique noise extracted from images for identification. By combining different noise identification techniques and removing biases that hinder performance, we provide a scalable solution that produces a 99% accuracy in a short period on time.

1 Introduction

Remote camera identification is the ability to identify an image or a feed's source [27]. By identifying the source of the camera we can authenticate its sender. It is the process of signing or exploiting the digital characteristics of the camera to distinguish its feed. This can be useful in areas where security properties such as camera authentication are important. Industries that rely upon cameras, especially internet-connected ones, need video feed security assurance. Being able to identify the origin of a feed enables better authentication and integrity verification.

The process of identifying images has been an active research area in the digital forensic community [6]. Since identifying the source can attribute an image to a person. Yet, the same technique can be also used to verify image authenticity. By identifying the camera source we authenticate the image source. There have been various active and passive techniques proposed to distinguish images based on their camera. Active techniques refer to ones that implement active security features on cameras such as secure hardware and digital signature to authenticate the feed. They are mostly consider from IoT security perspective, thus active camera security is part of IoT security research [22]. Passive techniques rely on fingerprinting unique camera features present in the final image to identify the feed [27]. While both techniques have their advantages, active techniques need heavy hardware and software overhead [36]. Moreover, passive techniques have yet to improve in reliability and efficiency [6]. This work will work with passive remote camera identification techniques to build an authentication solution.

Most passive camera identification techniques are neither scalable nor efficient for a reliable solution [6]. Scalability problems stem from inefficient fingerprinting and identification techniques. Moreover, previous research has been focused on the techniques possible for passive identification rather than providing an automated solution for security [6]. In this work, we present a fully automated efficient passive remote camera identification solution. The techniques aim to boost performance and efficiency as well as delivery a fully automated end-to-end solution. To achieve performance and efficiency we combine different effective fingerprinting and identification schemes. Also, we identified and eliminated some biases that hinder the identification [10].

For this work, we use PRNU noise to fingerprint image sources and CNN classification to identify the images. Also, we automate the end-to-end process of image collection, noise extraction, and training using open source SDKs. Following automation and the collection of 4000+ images coming from two different camera sources, the model was able to achieve a high degree of testing accuracy, 98%, in a relatively, short amount of time, 3 epochs.

The following sections make up the publication. Section 2 covers the background of the work. The background discusses the entire section before diving into the implementation and design. Section **??** covers the implementation of the solution before section **??**, evaluation. Section 5 and 6, include the discussion and related works. The paper concludes in section 7.

1.1 Overview

2 Background

Motivation Any chain is as strong as its weakest link. The same holds true in security [2]. With the explosion of IoT and internet-connected-device, more and more businesses are using technologies such as IP cameras [4]. IP cameras are becoming a vital part of many industries due to their convenience in remote event handling. Examples of industries using IP cameras are energy, communication, healthcare, and manufacturing [23]. Yet, due to the nature of IoT devices, they lack security considerations making them notoriously insecure [39]. Off-the-shelf hardware and software capabilities make encryption and secure hardware a challenge [36]. Yet, active techniques such as camera secure hardware and trusted environments are impractical to put in place. For active security, industries must change the hardware and software capabilities or buy special cameras that may not be practical to their implementations. Both options are impractical. With that said, passive fingerprinting of image feed is an attractive alternative to security. By using hardwaremetry to fingerprint images [27], we can prove some security properties on standard cameras. By identify some unique characteristic image produce, we can build a method that verifies identify image source. By fingerprint images, we can authenticate and verify their integrity.

Image Noise In a survey on remote camera identification techniques, Bernacki et. al [6] distinguishes between two types of passive camera identification, Individual Source Camera Identification (ISCI) and Source Camera Model Identification (SCMI). The former aims to identify images from two or more identical cameras and the latter aims for the model or brand distinction. In this survey, most work done on SCMI has been around analyzing fixed Sensor Pattern Noise (SPN) and deep models techniques. SPN is a permanent noise pattern found in the image that is unique to that camera sensor and is introduced at acquisition time. The main part of such noise is the PRNU noise. PRNU noise, first introduced in [24], is the noise caused by manufacturing imperfections of camera sensors silicon wafers [29, 30]. Such imperfections leave a slightly varying level of intensity upon illumination causing a pattern across different camera image acquisition. By extracting such PRNU noise pattern, a camera reference pattern can be established that can identify images by how much they are close or a part of such reference [29, 19, 30]. On the other hand, prominent techniques involve learning image sources using deep models, commonly CNNs. By feeding a sufficiently large enough data set made of different images of different sources and labeling them accordingly, and by minimizing the cost function, the model can learn to identify images based on their respective cameras [11, 37, 7, 9, 16].

Noise extraction Moreover, various denoising filters to extract such PRNU patterns have been introduced. Without affecting the image content, the denoising filter is used to suppress the PRNU noise generated from the image. Then the noise is extracted by subtracting the denoised image from the noisy image. The formula for noise extraction is presented in 1.

$$N = I - F(I) \tag{1}$$

N is the noise extracted by subtracting the original image I from the denoised image $F(I)$. F symbolizes the denoising function.

Camera Pipeline While the filters introduced [19, 38] are successful in denoising PRNU noise, modern cameras are made up of more than one noise. In a modern camera, there exists a pipeline of processing steps, from acquisition to storage that processes, enhances, and stores the image [28]. Each stage in this pipeline introduces its own noise patterns. Such noise can interfere with PRNU noise extraction to produce inconsistent noise patterns. Noise introduced via color correction and tone mapping can degrade the noise extraction pattern [10] thus constrain the identification process [13]. Hence in this work, we aim to extract raw unprocessed images from the camera sensor to improve denoising which leads to better noise extraction and identification. Moreover, images are compressed using a compression algorithm before storage, such compression can also interfere with the noise. Hence the pipeline includes raw image extraction and a special storage format to reduce interfering image noise.

Biases Bias can interfere with the identification accuracy and hinder the performance of the model. With bypassing different processing and storage stages, the aim is to remove many biases. The first step in removing biases is having control over raw image acquisition, storage, and formating. For that, we use two identical Intel® RealSense™ Depth D455 Cameras for ISCI identification for data set collection. We selected

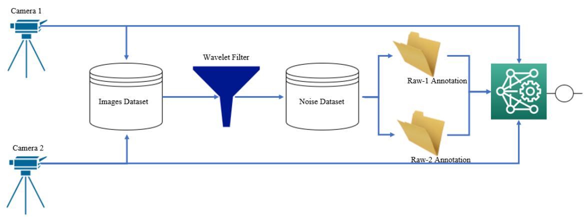

Figure 1: System Pipeline

these cameras due to their open-source SDKs that configure and automate the image acquisition and storage process. The SDK enables capturing raw unprocessed images and storing them as arrays instead of compressing them. Moreover, the automation enables the data collection step to integrate with the system. Besides such optimizations, image content can contribute to a significant amount of bias in a CNN model. For that, a second data set is build based on the uniform noise extracted from the original data set. To remove image content bias, the identification process uses the noise data set.

3 System Design and Implementation

As mentioned, we propose an efficient optimized model for remote camera identification. This section goes over data collection, noise extraction, identification, and automation. The first step in the identification process is image acquisition and storage. Figure 1 the image acquisition process along with the next step in the pipeline. The next step is the image process and PRNU noise extraction. PRNU noise forms the second data set, the data set used for training. Following the collection of the second data set, a necessary preprocessing step occurs. The preprocessing step. The model uses the training set for learning and evaluation. The training includes choosing a simple model and optimizing it. Accuracy and performance optimization includes PSO the convex gradient descent adjustments.

3.1 Data Collection

The first step in the system's pipeline is image acquisition and data collection. For any machine or deep learning model, a sufficient amount of data must be prepared for the learning process. The quantity of the data heavily depends on the quality. The quality of data for machine and deep learning is determined by various factors including but not limited to biases related to image content and noise [17]. This is since such biases can be considered as learning features by the model, thus the result is a reduced accuracy and performance.

A significant bias, that can effect quality, happens at the image acquisition level in the camera pipeline. The advancement in modern cameras has been towards outputting a final image accurate to the real world. For that, beyond the image acquisition pipeline, that is necessary to any image taken, there exists an image processing pipeline that transforms the image into its final pleasant form [10]. While traces left at the acquisition steps by the camera sensor, which include PRNU, and CFA interpolation, and demosaicing, are unavoidable, other stages can be removed. Stages such as color correction, tone mapping, and gamma compression can be bypassed by outputting raw images. Since each of such processing steps leaves a noise trace that can interfere with noise extraction in future stages, using raw unprocessed images can significantly improve the PRNU noise pattern extracted. Such improvements in the data set collection can have a significant improvement in the data set required for identification. Moreover, it can boost identification performance and accuracy.

For raw image acquisition and data collection, we utilize Intel® RealSense™ cross-platform SDK [1] to configure and automate the image acquisition process. The SDK is an open-source framework for Intel® RealSense™ [1] cameras that allows manipulation and automation of the image acquisition and storage pipeline. With that said, a script is developed for the image acquisition process that connects to each camera, configures the image stream to 16-bit raw unprocessed each 30 seconds frame, and stores them as raw data. Raw images are single-channel intensity images produce directly by the image sensor and passed through and CFA to produce a single dimension image with 16-bits of data [34]. In this case, since we are using 16-bit raw images, each image is stored as an array.

Using the camera and the framework, we captured over 4,000 1280 x 800 16-bit minimally processed raw image streams from each camera and stored them as arrays. The process of acquisition and storage is automated and integrated in the system as the first stage of the pipeline.

3.2 PRNU Extraction

The second stage of the pipeline is concerned with utilizing a PRNU noise extraction mechanism to denoise the image data set and form a new noise data set. Nothing is perfect, like everything in the world, images are not perfect. Due to many factors, events contaminate images with different kinds of acquisition, transmission, compression, and storage noise. Yet, while noises are inevitable, over the years, there have been many successful denoising techniques. Techniques of which can extract much noise [15]. As mention, for this work, PRNU is the main noise of interest. For PRNU noise extraction, the experiment relies on the same techniques used in prior well-known work [29, 19, 38]. In this section, we examine the noise extraction technique and build the noise data set.

The goal is to extract noise that can serve as a feature pattern for identification. PRNU noise, as mentioned, is a unique noise pattern that can distinguish different cameras. By extracting PRNU noise and training a CNN model, we can perform camera identification.

For building the new noise data set the main problem is being able to extract the specific noise pattern required. If we are able to find a filter that is able to suppress PRNU noise from images, we can use formula (1) to extract PRNU noise. The formula works by subtracting the denoised image from the original noisy image to end up with the noise.

3.2.1 Wavelet Transformation

As mentioned, the goal is to extract the specific unique noise pattern, PRNU. Image denoise is still an active research area and a fundamental problem in digital signal processing (DSP). The extraction of single noise is impossible. Yet, since PRNU noise appears in the final image as a smoothness inconsistency, smooth noising removal methods can work [1]. Denoising images by smoothing methods [1] extracts PRNU noise along with other noise. But, since raw image avoids many other processing noises, PRNU is the dominant noise that remains.

Spatial-Frequency Filtering is one of the popular filters used for smoothness denoising. The filtering method transforms the digital image into a frequency signal. Following the transformation into the frequency domain, a function extracts the noise frequency. For stationary and periodic signals, the Fast Fourier Transform (FFT) enables effective extraction. Given a stationary signal and a cut-off frequency, FFT smoothens the signal by removing frequency size above the threshold.

$$f(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-iwt} dt, \tag{2}$$

FT is illustrated in 2, were an image in transfered into a non stationary image signal. Following transformations are built on-top of FT.

While FFT does a good job for periodic signals, images transformed into the frequency domain are nonperiodic. Images are complex frequencies with varying magnitudes and need to be model as a function of time. Since FFT cannot achieve good time resolution, the Short-time Fourier transform (STFT).

$$f(\tau, \omega) = \int_{-\infty}^{+\infty} f(t) w(t-\tau) e^{-iwt} dt, \tag{3}$$

The above formula 3 illustrates STFT. The formula is based on FT with the addition of a moving window function w. τ presents the time localizing translation parameter, the translation of the function across the signal.

STFT came as a solution to handle the non-periodic signals, especially images signals. STFT attempts to divide the non-stationary signals into discrete stationary ones. Given a window function of fixed length w, w divides the signal into the small stationary signal. After obtaining a set of stationary signals that make up the image signal, an FFT transformation on each signal can occur. The shortening of the time resolution in small windows W results in a much better frequency resolution. This flow from the inescapable law of physics which states:

> We cannot know what frequency exists at what time instance, but we can know what frequency bands exist at what time intervals.

The law states the higher the time resolution, the greater the frequency uncertainty. For high-frequency resolution, short time windows yield less uncertainty and vice versa. The law is presented in the following equation.

$$\Delta t \Delta f \geq \frac{1}{4\pi} \tag{4}$$

Δt and Δf are reciprocal and bounded by 4π.

While STFT outperforms FFT, it still has some major limitations that influence certainty and noise extraction. One major drawback is the fixed length of the window function. A fixed window cannot distinguish between high-frequency areas and low ones. High-frequency areas need shortened time frames and low-frequency areas need a widened time frame according to formula 4. Thus, a fixed windows function results in degraded noise suppression and smoothness.

We cannot property 4, but, we can improve on it. If we can have a scalable window function, optimal noise suppression is possible. We can guarantee optimal noise suppression if we have an adjustable window function that can shorten and widen depending on the frequency-time resolution. With that said, the Wavelet Transform introduced the concept of wavelets. Wavelets are adjustable time frames that yield an optimal frequency-time resolution.

$$W\{f(\tau, s)\} = \int_{-\infty}^{+\infty} f(t) \frac{1}{s} \psi(\frac{t-\tau}{s}) dt, \tag{5}$$

The above equation 5 represents the Wavelet Transform. Instead of the window function of equation 3, we have a complex conjugate known as the wavelet ψ. The wavelet represent a modified window function that is scalable depending on the frequency region on the signal represented by s. s is the inverse of frequency and determine the window in time.

Amongst the mentioned techniques, the Wavelet Transform is the most effective at PRNU extraction. Since WT can ensure a high degree of frequency-time resolution, better PRNU denoising is possible.

3.2.2 Extraction

Given the data set made up of raw unprocessed data, we can extract the PRNU image noise to form a new noise data set. The first step in the noise data set process is denoising each image. Following that, noise extraction occurs by applying equation 1. For each image in the original data set, noise extract is the difference between its pixel values and its denoised version. WT denoises the image, and noise is then extracted.

3.2.3 Noise Data set

We use the noise data set for training for the following reason. Plotting the new noise data set for visual representation results in a homogeneous field color. Such consistent color across all the different images taken removes image content bias. Since there is no image content in the noise image arrays, there is no bias introduced by image content features. Performance and accuracy improve with the reduction of biases.

3.3 Training

Following the denoising and the noise extraction process, we need an efficient CNN model for learning against the noise data set. For that, this section covers and justifies the model and its architecture. Yet, before any training can take place, preparing the noise data set for training is necessary. This section covers all necessary step required for identification. The section starts by covering the preprocessing step before describing the model and the training process. Following that, the section covers model optimization before evaluating and testing the results.

3.3.1 Preprocessing

The preprocessing stage is an essential step in the identification process. The first step is data annotation. We annotated and shuffle the data for binary CNN classification. Before training the model the noise data set must for formated and spit in training and testing sets. The training set if necessary for learning PRNU patterns found in the noise data set. The training data set forms 70% of the whole noise data set. While the evaluation of the model uses the testing data set. After the data annotation, shuffling, and segregation, the model is ready for training.

3.3.2 Training

For training, we train a simple CNN model derived from the documentation on the noise data. We derive the CNN model from simple binary cats and dogs classification built by [12] which is able to produce a 90%+ testing accuracy. We chose this model due to its simplicity that can aid identification performance.

The model is a linear stack of layers made of three components, an input layer, hidden layers, and an output layer. The input layer accepts the 1280 x 800 16-bit noise pixels as input for each noise image array. The hidden layers are 3 layers with 2 convolutional layers and 1 fully connected dense layer. The final layer, the output layer, displays the probability results of the classification.

Further dive into each layer, the input layer consists of 1280 x 800 input nodes, each presenting a pixel value. The input layer depends on the dimension of the image, a cropped image yield a reduced number of nodes. The input layer connects to the first layer in the hidden layers. The first layer is, as mentioned, a convolutional layer that receives input from prior-layer neurons. A fixed number of neurons make up the layer, for the first model tested, we chose an arbitrary number of neurons, 256. This number is common and used in the simple dogs and cat classification [12].

Going more in-depth, 3 components make up a convolutional layer. The first component is the weight and biases associated with each neuron or node. For this, as mentioned 256 neurons make up the convolutional layer in the model. Neurons represent the different weights and biases of a convolutional layer required to yield a feature map. The second component is the activation function necessary to formalize the weights and biases of the layer to yield a feature map. The convolutional layers in the model use Rectified Linear Unit Function (ReLU) as the activation function. ReLU reduces the learning time by efficient gradient decent adjustments. The last component in the convolutional layer is an operation used to extract features, smooth and sharp. Pooling is the operation of downsampling the features to increase performance and extracting edges and distinct features. 2 adjacently connected convolutional layers make up 2 out of 3 layers in the hidden one both using 2-dimensional max pooling.

The final layer in the hidden layer is 64 nodes fully connected dense layer. Unlike convolutional layers, a dense layer is one in which each neuron connects to each one in the previous layer. While dense layer increase model complexity, it enhances feature extraction and classification. The 2 components that make up the layer are the Sigmoid activation function and the 64 nodes. The dense layer is then finally connected to the output layer, the output layer is then given by two nodes that resulting in the last feature map. The output is a probability, the higher probability is the classification decision.

3.3.3 Particle Swarm Optimization

Following the simple architecture of section 3.3.2, we optimize the model using Particle Swarm Optimization (PSO) [25]. PSO relies on the iterative tuning and model evaluation of the different hyper-parameters to obtain optimal results. By tuning and evaluation the model on different hyper-parameter we can select the optimal accuracy model.

CNN relies on loss function for training, the change of the loss function following each input decides classification. Minimizing the loss or cost function summarizes CNN classification. Gradient descent is loss function reduction by the architectural and hyper-parameter tuning [33].

For this model we tuning the following parameters:

- Number of convolutional layers;
- Number of neurons in the convolutional layers;
- Number of dense layers;

Following the optimization, we generated 27 models presented in 3.3.3.

Following optimization, the optimized model is integrated and automated into the system's pipeline. With that, we have an end-to-end solution for camera identification. Starting from image extraction to identification, the model achieves high accuracy and performance. We automate the model by developing an algorithm that can run in the background. The algorithm starts by extracting images from each camera at random intervals. After reaching a specific threshold, the algorithm starts PRNU noise extraction. Following building the second data set, automatic PSO optimization and model selection occurs. After selecting the optimal model, identification is the classification of new images. This is also automated by taking and evaluating a random image every specified time.

Conv Layers	Nodes	Dense	Training Accuracy
1	128	0	98.22%
1	128	1	97.83%
1	128	2	97.52%
1	32	0	99.42%
1	32	1	97.27%
1	32	2	98.5%
1	64	0	97.32%
1	64	1	97.64%
1	64	2	97.64%
2	128	0	99.05%
2	128	1	98.28%
2	32	0	96.88%
2	32	1	98.92%
2	32	2	98.2%
2	64	0	98.08%
2	64	1	98.81%
2	64	2	99.1%
2	128	0	96.69%
3	32	0	97.89%
3	32	1	98.71%
3	32	2	97.61%
3	64	0	98.91%
3	64	1	97.92%
3	64	2	98.37%
3	128	0	96.69%
3	128	1	98.04%

4 Evaluation

For this experiment we collected a data set of 4000+ images coming from both cameras. However, only a small fraction of the entire data set was required for a 99% accuracy. As mentioned, with the increase in dimension, the time require to denoise an image increase. Due to denoising constraints imposed on image dimensional, images are cropped into 256 x 256 before denoising. Moreover, during experimentation we realize that a data set of 500+ from each camera is enough to achieve a 99% accuracy after 3 epoch of 32 batch size.

Following PSO and the generation of 27 different models, we selected the model with high accuracy after 2 epochs. Model with 1 convolutional layer, 32 nodes, and 0 dense layers achieved the highest accuracy after 3 epochs.

5 Discussion

With the ever-increasing reliance on internet-connect-camera in industry, our solution can prove invaluable. This is true especially in industries where camera security is crucial. Yet, while this solution is accurate and

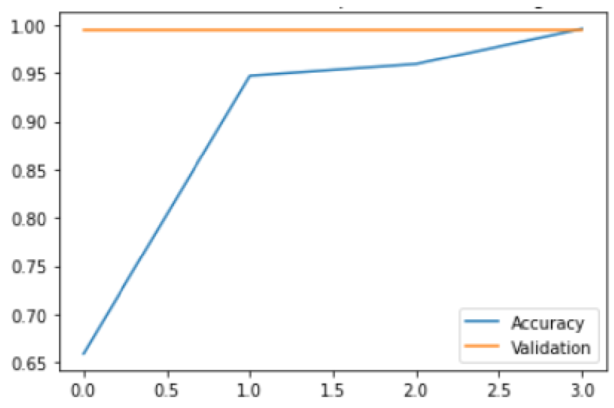

Figure 2: Optimized model (1 convolutional layer, 32 nodes, and 0 dense layers) testing and validation accuracy

practical, it is vulnerable to some attacks. We identify one possible attack that can render the identification useless. But, such an attack is technical and relies on physical camera accessibility a deep understanding of the technology.

5.1 Attack Scenario

Assume the following scenario, attacker Bob wants to bypass the identification process to send false images. Assuming secure storage of image and noise data sets, yet Bob has access to the camera, Bob can capture an image. After Bob has an image, he can extract the PRNU noise of the image, using formulas 1 and 5, and inject it into a false image. The new image might pass the identification process since it contains a similar PRNU pattern learned by the model. Bob can also enhance classification by denoising the false image before adding the PRNU noise. Thus making the PRNU the only dominant noise pattern in the image. Other attack scenarios against deep learning techniques are identified in [31, 21]. Such attacks relies on the adversary knowing different properties of the model and the system to bypass identification.

Unfortunately, there is no effective solution to such an attack. Yet, due to the complexity and technicality of such an attack, it is unlikely and involves access to images or cameras.

5.2 Scalability

Furthermore, one scalability issue that the model needs to contend with the time required for PRNU extraction. PRNU extraction can take a relatively long time with respect to other processes in the model. It can take up to 20 seconds to denoise 1280 x 800 images [8]. But, in this solution, noise extraction happens once at the beginning, moreover, it is automated in the background. After, noise extraction and training the identification process happens instantaneously. Moreover, we tested the model on cropped images of 256 x 256; the model still achieves a high degree of testing accuracy.

6 Related Work

In his seminal work, Lukas et. al [29, 30] demonstrated the possibility of passive remote camera identification and verification. The identification focuses on extracting and establishing a noise reference pattern. The paper at to extract PRNU noise using a wavelet-based denoising filter algorithm like 1. Following PRNU extraction from a different image source, the identification process relies on a correlation function. The Correlation function establishes a reference pattern for each noise source. Noise reference then evaluates any new image; identification occurs on the source noise pattern. The further the noise extract from the source reference the less likely the image is from the camera.

Yet, while the seminal work facilitated much research in PRNU and passive identification, it is only proof of work. Moreover, the work doesn't consider other variables and biases that affect identification. Also, it doesn't go into the nature and storage of the images in place. The model is not designed to provide a live solution for security and authentication. For PRNU noise extraction works using equation [1].

Another work the attempts to check wavelet-PRNU identification against compressed images is [2]. The work relies on using a similar identification technique to [1] while presenting a solution for compressed video feed identification. The results show that the experiment was successful in getting a high degree of accuracy for classification, around 90%

Following this work, Goljan et. al [20], attempted similar pattern noise extraction. Instead of using correlation, they use cross-correlation analysis and peak-to-correlation energy ratio (PCE). The experiments also proved successful with slight accuracy improvement. But, a considerable improvement in the accuracy came in 2012 with the use of the circular correlation norm to identify cameras.

Moreover, beyond correlation and PCE, PRNU was also examined in other work that relies on different identification mechanisms.

Baar et. at [3] introduces an image identification database grouping. The intuition behind it is the identification and grouping of images coming from the same camera. The work examined different approximation methods for grouping images based on origin and used the K-means algorithm to achieve such grouping. The technique used the K-means algorithm to automate the process, replacing correlation and using MATLAB for more efficient denoising and clustering.

6.0.1 Deep Models

Camera identification has taken a great leap with the improvements of deep methods [11]. This is since with the advancement of machine learning, data collection, and computation, identification can be model as a classification problem.

In 2016, Baroffio et.al proposed a CNN deep model for camera identification for both individual device-attribution and model-attribution [5]. In the work, the author postulates that deep models are able to extract such noise features and learn to distinguish cameras. By feeding a deep model source labels image, the hope is the model is able to identify the source. The paper goes through the architecture of the convolutional models and explains each step and layer. The model uses a well-known Dresden data set [18] to carry out two experiments. The first being on individual camera identification and the second is for model identification. The model used training and testing accuracy metrics for evaluation. The results produce a high degree of accuracy for such identification in both model and device verification.

After Baroffio et.al, the same concept was applied using a different deep model. In 2017, Chen et. al [11] investigated the use of residual Neural Networks (ResNet) in three concepts for camera identification, brand-attribution, model-attribution, and device-attribution. This is an important distinction since it allows the identification of a mobile's brand, model, and sensor. The work implements ResNet deep models for identification. The data set, composed of more the 15,000 images captured from 13 different phones. The Dresden Database was also used for this work [18]. The Deep model was implemented on the data set to train, classified, and identify 13 different phones and brands of phones. The author claims that ResNet, for such classification, produced a much more effective approach to image forensics than [5]. This is due to its enhanced accuracy results. Moreover, the author claim his model outperformed well-known CNN models such as AlexNet [26] and GoogLeNet [35]. Following this work, [16] proposed another technique for mobile source camera identification. Both methodologies are similar, yet, the latter work relied on a different dataset, MICHE-I Dataset [14]. Moreover, it implemented its own custom CNN architecture model for the learning and testing. The work tuned its own hyperparameters. The result also claims a higher accuracy than [5].

Based on the power of CNN to automatically learn to classify image sources, Tuama et. al, [37] added a high pass filter to the input image for optimization. But, while this work uses noise extraction layers, the filter only reduces the image content[32]. The high-pass filter was not used to extract noise that is in turn used for training.

While this work includes feature extraction filters and deep models, it is different than the work done in the thesis. The papers mentioned don't for biases and don't provide an end-to-end automated solution. By relying on automatic identification with noise extraction or noise data collection, image content affect the results. The CNN model with that can grow image content bias and produce an unacceptable amount of false positives.

7 Conclusion

Secure camera identification techniques have gained interest with the rise of IP cameras. Yet, while active techniques secure such devices, they often come with heavy overhead. Since cameras rely on camera sensors, we can use the field of hardwaremetry to exploit it and prove some security properties. Industries reliance on camera and other IoT devices have increase interest in their security.

In this work, we present a practical security technique for remote camera identification. The solution presented a new approach for camera identification based on noise data set. The paper also differs from other existing work by providing an end-to-end scalable solution with controlled biases.

References

[1] Intel® realsense™ cross-platform sdk.

[2] Arce, I. The weakest link revisited [information security]. *IEEE Security Privacy 1*, 2 (2003), 72–76.

[3] Baar, T., van Houten, W., and Geradts, Z. Camera identification by grouping images from database, based on shared noise patterns. *arXiv preprint arXiv:1207.2641* (2012).

[4] Banafa, A. Iot standardization and implementation challenges. *IEEE internet of things newsletter* (2016), 1–10.

[5] Baroffio, L., Bondi, L., Bestagini, P., and Tubaro, S. Camera identification with deep convolutional networks. *arXiv preprint arXiv:1603.01068* (2016).

[6] Bernacki, J. A survey on digital camera identification methods. *Forensic Science International: Digital Investigation 34* (2020), 300983.

[7] Bernacki, J. Robustness of digital camera identification with convolutional neural networks. *Multimedia Tools and Applications* (2021), 1–17.

[8] Bernacki, J., Klonowski, M., and Syga, P. Some remarks about tracing digital cameras-faster method and usable countermeasure. In *SECRYPT* (2017), pp. 343–350.

[9] Bondi, L., Baroffio, L., Güera, D., Bestagini, P., Delp, E. J., and Tubaro, S. First steps toward camera model identification with convolutional neural networks. *IEEE Signal Processing Letters 24*, 3 (2016), 259–263.

[10] Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., and Barron, J. T. Unprocessing images for learned raw denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 11036–11045.

[11] Chen, Y., Huang, Y., and Ding, X. Camera model identification with residual neural network. In *2017 IEEE International Conference on Image Processing (ICIP)* (2017), IEEE, pp. 4337–4341.

[12] Chollet, F. Building powerful image classification models using very little data. *Keras Blog 5* (2016).

[13] Dang-Nguyen, D.-T., Pasquini, C., Conotter, V., and Boato, G. *RAISE: A Raw Images Dataset for Digital Image Forensics*. Association for Computing Machinery, New York, NY, USA, 2015, p. 219–224.

[14] De Marsico, M., Nappi, M., Riccio, D., and Wechsler, H. Mobile iris challenge evaluation (miche)-i, biometric iris dataset and protocols. *Pattern Recognition Letters 57* (2015), 17–23.

[15] Fan, L., Zhang, F., Fan, H., and Zhang, C. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art 2*, 1 (2019), 1–12.

[16] Freire-Obregón, D., Narducci, F., Barra, S., and Castrillón-Santana, M. Deep learning for source camera identification on mobile devices. *Pattern Recognition Letters 126* (2019), 86–91.

[17] Gerhard, T. Bias: considerations for research practice. *American Journal of Health-System Pharmacy 65*, 22 (2008), 2159–2168.

[18] Gloe, T., and Böhme, R. The'dresden image database'for benchmarking digital image forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (2010), pp. 1584–1590.

[19] Goljan, M. Digital camera identification from images–estimating false acceptance probability. In *International workshop on digital watermarking* (2008), Springer, pp. 454–468.

[20] Goljan, M., Fridrich, J., and Filler, T. Large scale test of sensor fingerprint camera identification. In *Media forensics and security* (2009), vol. 7254, International Society for Optics and Photonics, p. 72540I.

[21] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[22] Hassan, W. H., et al. Current research on internet of things (iot) security: A survey. *Computer networks 148* (2019), 283–294.

[23] Hurtoi, V., and Avadanei, D. Iot project management. *Informatica Economica 24*, 3 (2020), 75–80.

[24] Janesick, J. R. *Scientific charge-coupled devices*, vol. 83. SPIE press, 2001.

[25] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (1995), vol. 4, pp. 1942–1948 vol.4.

[26] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems 25* (2012), 1097–1105.

[27] Lin, X., Li, J.-H., Wang, S.-L., Cheng, F., Huang, X.-S., et al. Recent advances in passive digital image security forensics: A brief review. *Engineering 4*, 1 (2018), 29–39.

[28] Lukac, R., Martin, K., and Plataniotis, K. N. Demosaicked image postprocessing using local color ratios. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 6 (2004), 914–920.

[29] Lukas, J., Fridrich, J., and Goljan, M. Determining digital image origin using sensor imperfections. In *Image and Video Communications and Processing 2005* (2005), vol. 5685, International Society for Optics and Photonics, pp. 249–260.

[30] Lukas, J., Fridrich, J., and Goljan, M. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security 1*, 2 (2006), 205–214.

[31] Marra, F., Gragnaniello, D., and Verdoliva, L. On the vulnerability of deep learning to adversarial attacks for camera model identification. *Signal Processing: Image Communication 65* (2018), 240–248.

[32] Qian, Y., Dong, J., Wang, W., and Tan, T. Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics 2015* (2015), vol. 9409, International Society for Optics and Photonics, p. 94090J.

[33] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[34] Sumner, R. Processing raw images in matlab. *Department of Electrical Engineering, University of California Sata Cruz* (2014).

[35] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[36] Tawalbeh, L., Muheidat, F., Tawalbeh, M., and Quwaider, M. Iot privacy and security: Challenges and solutions. *Applied Sciences 10* (06 2020), 4102.

[37] Tuama, A., Comby, F., and Chaumont, M. Camera model identification with the use of deep convolutional neural networks. In *2016 IEEE International workshop on information forensics and security (WIFS)* (2016), IEEE, pp. 1–6.

[38] Van Houten, W., and Geradts, Z. Source video camera identification for multiply compressed videos originating from youtube. *Digital Investigation 6*, 1-2 (2009), 48–60.

[39] Wurm, J., Hoang, K., Arias, O., Sadeghi, A.-R., and Jin, Y. Security analysis on consumer and industrial iot devices. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)* (2016), IEEE, pp. 519–524.

Notes