# CAMPrints: Leveraging the "Fingerprints" of Digital Cameras to Combat Image Theft

Bangjie Sun
National University of Singapore
bangjie@comp.nus.edu.sg

Mun Choon Chan
National University of Singapore
chanmc@comp.nus.edu.sg

Jun Han
KAIST
junhan@cyphy.kaist.ac.kr

## Abstract

Photo sharing is increasingly popular, driven by social media platforms like Instagram and services such as Flickr and Google Photos. However, this growth has been accompanied by significant issues, particularly image theft. To address this issue, we introduce *CAMPrints*, a robust system for detecting image theft. *CAMPrints* verifies whether edited images found online contain camera fingerprints matching those of user-provided reference images. The system overcomes the challenges of identifying images altered by diverse image processing operations. We select a small yet representative set of operations by categorizing them based on their impact on pixel values and locations. A deep-learning model is trained to recognize and compare camera noise patterns pre- and post-editing. We conduct real-world evaluations involving 36 cameras across eight make-and-model combinations, along with over 40 image processing operations applied to more than 4,000 images. *CAMPrints* achieves an average AUC of 0.92, significantly outperforming the state-of-the-art methods by up to 1.8 times.

## CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Computer vision**.

## Keywords

Camera Fingerprint, PRNU, Image Theft, Copyright Monitoring

**ACM Reference Format:**
Bangjie Sun, Mun Choon Chan, and Jun Han. 2025. CAMPrints: Leveraging the "Fingerprints" of Digital Cameras to Combat Image Theft. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3711875.3729158

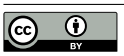
## 1 Introduction

Photo sharing has become increasingly popular and now accounts for over 30% of the global digital content market [63]. Social media platforms have played significant roles in making photography more accessible, driving its widespread appeal. Instagram is a favored platform among photographers while photo-sharing sites like Flickr and Google Photos provide convenient ways for users to store and share their images with friends and family. The proliferation of digital content has introduced significant challenges, particularly financial losses due to *image theft*, the unauthorized use of copyrighted photographs. It is estimated that approximately 2.5 billion images are stolen each day, leading to annual losses of around $600 billion in potential licensing fees and damages [16, 20, 57]. Exacerbating the problem, copyright violators often take deliberate steps to evade measures like watermarking and embedding metadata. It is reported that 72% of stolen images are altered and 68% of stolen images have watermarks removed, signaling intentional copyright violations [7, 20, 42].

To address the growing issue of image theft, *copyright monitoring services* like TinEye [73] and Pixsy [59] have gained prominence. Pixsy, for instance, has assisted over 200,000 photographers, illustrators, agencies, and artists globally in detecting and managing copyright infringement cases [59]. Additionally, many photo-sharing platforms, such as Flickr, partner with these services to protect their users' intellectual property and ensure adherence to copyright regulations [18, 44, 61]. Such collaborations empower creators to safeguard their work from unauthorized use while protecting the integrity of the platforms.

For image theft detection, copyright monitoring services primarily rely on *Reverse Image Search*, a technique that identifies similar or identical images on the internet using an image as input [19, 40, 66]. A well-known example is Google Image Search [41]. However, this process typically analyzes visual patterns such as colors, shapes, and textures, which often result in **false positives**, alongside metadata like EXIF data, which can be easily **altered or removed** [22, 23, 26, 77]. Recent advancements, both commercial and academic, focus on leveraging the unique hardware fingerprints of digital cameras, specifically the photo response non-uniformity (PRNU) – a distinctive noise pattern caused by imperfections in imaging sensors [27, 37, 38, 55]. However, these approaches are highly sensitive to various image processing operations (e.g., cropping, rotation, and sharpening), making them less effective against deliberately edited images.

In light of this, we pose the following question: ***can we develop a robust solution capable of accurately detecting image theft, even when the images have undergone extensive processing operations?*** To address this, we propose *CAMPrints*, a robust system designed to verify whether an edited *test image* suspected of image theft contains camera fingerprints matching those of *reference images* provided by the user. The key novelty of *CAMPrints* lies in the robust matching algorithm that extracts discernible features from a camera's *noise pattern* (i.e., its unique hardware fingerprint) even after extensive image processing operations. Figure 1 illustrates a typical use case for *CAMPrints*. (1) The user uploads photos to social media or photo-sharing platforms and simultaneously saves them in *CAMPrints* as *reference images*. (2) A copyright violator steals the photos, applies intentional edits, and republishes them. (3) Social




*MobiSys '25, June 23–27, 2025, Anaheim, CA, USA*

ACM ISBN 979-8-4007-1453-5/2025/06
https://doi.org/10.1145/3711875.3729158

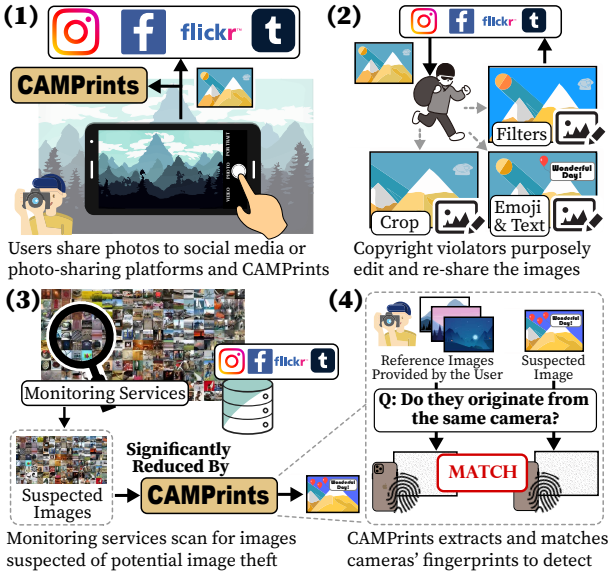



**Figure 1: Figure illustrates a typical use case for *CAMPrints*. (1) The user uploads photos to social media or photo-sharing platforms and saves them in *CAMPrints* as *reference images*. (2) A copyright violator steals the photos, applies intentional edits, and republishes them. (3) Social media and photo-sharing platforms collaborate with monitoring services to scan their databases for images suspected of potential image theft and input these images into *CAMPrints* for further filtering and verification. (4) For each suspected image, *CAMPrints* extracts its camera's fingerprint, comparing it with those of the *reference images* provided by the user. *CAMPrints* determines whether these images originate from the same camera. If a match is detected, *CAMPrints* notifies the user of potential image theft.**

media and photo-sharing platforms collaborate with monitoring services to scan their databases for images suspected of potential image theft and input these images into *CAMPrints* for further filtering and verification. (4) For each suspected image, *CAMPrints* extracts its camera's fingerprint, comparing it with those of the *reference images* provided by the user. *CAMPrints* determines whether these images originate from the same camera. If a match is detected, *CAMPrints* notifies the user of potential image theft.

Designing *CAMPrints*, however, comes with the following challenges. First, due to the virtually infinite range of possible image processing operations and their combinations, it is impractical to account for all of them explicitly. To address this complexity, the core idea is to identify **a small yet representative set** of image processing operations that can achieve high coverage across a broad spectrum of potential transformations. Since image processing operations primarily affect pixel values, spatial locations, or both (e.g., color adjustments affect only pixel values, while rotation affects only spatial locations), we categorize the coverage of these operations by focusing on the possible combinations of pixel value and location changes. Leveraging this observation, we propose a mathematical model to quantify the effects of image processing operations on pixel values and spatial locations of an image and its noise patterns (e.g., camera fingerprints). From this, we select representative image processing operations that introduce a broad spectrum of value and location changes. By training *CAMPrints* using only these representative operations, we significantly reduce problem complexity while preserving robustness.

Second, training a deep-learning model to recognize and compare noise patterns before and after representative image processing operations poses an additional challenge. Since image processing operations modify features of noise patterns in varied ways, directly inputting all data into the model can hinder convergence. To overcome this, we employ the *Curriculum Learning* [69] strategy, which improves convergence and performance by presenting data samples in order of increasing difficulty, similar to how humans learn. We use our proposed mathematical model to quantify learning difficulty based on the extent of changes in pixel values and spatial locations (e.g., operations like cropping followed by resizing are more challenging as they affect both).

We conduct an extensive evaluation of *CAMPrints* to demonstrate its effectiveness through real-world experiments. Specifically, we test *CAMPrints* using 36 smartphone cameras across eight different make-and-model combinations, with at least three instances per make-and-model to ensure diversity. The evaluation includes more than 40 types of image processing operations, covering a wide range of transformations applied through popular image editing tools like Photoshop and smartphone apps such as iPhone Photos. With experiments on over 4,000 images, our results show that *CAMPrints* achieves an average area under the ROC curve (AUC) of 0.92 across all tested image processing operations, significantly surpassing state-of-the-art methods by up to 1.8 times in terms of AUC. Furthermore, we conduct experiments using commercial products, including social media platforms like Instagram, Facebook, and Tumblr, as well as photo editing tools such as iPhone Photos, Android Gallery, Snapseed, and Adobe Photoshop. *CAMPrints* demonstrates robustness and effectiveness in detecting image theft across these scenarios, achieving an AUC of up to 0.93. Additionally, we integrate *CAMPrints* with Reverse Image Search, a key component of existing copyright monitoring systems. Our results reveal that *CAMPrints* reduces false positives by over 80%, averaging only 3.9 false positives per search while maintaining high detection accuracy.

## 2 Background

We first introduce copyright monitoring services and their limitations. We then present camera "fingerprints" and the physics primer of these "fingerprint".

### 2.1 Copyright Monitoring Services

*Copyright monitoring services*, such as TinEye and Pixsy [59, 73], are increasingly popular among photographers to detect unauthorized use of their images online. Many social media and photo-sharing platforms partner with these services to protect their users' intellectual property and ensure adherence to copyright regulations [18, 44, 61].

**Core Functionality.** These tools employ *Reverse Image Search* to scan billions of web pages and databases, comparing uploaded

images with indexed content [19, 40, 66]. TinEye, for example, employs image recognition algorithms to identify matches based on *visual similarities*, such as colors, shapes, and textures. By automating the discovery process, these services save time and provide photographers with actionable insights to effectively protect their copyrights.

**Limitations.** Although these tools are optimized for a high detection rate of stolen images, they suffer from a significant drawback: a high number of false positives, where *benign images* are mistakenly identified as matches. This often occurs when images share common elements, such as landmarks or logos. For example, Pixsy has acknowledged that search results may incorrectly link to images with similar street scenes or landmarks taken by different photographers [59]. These false positives increase the burden on users, who must review numerous results to determine whether they are actual infringements. This process is time-consuming and resource-draining, hindering efficient detection of image theft. To address this, we develop *CAMPrints*, a system that reduces false positives by comparing camera fingerprints while maintaining accurate detection of image theft.

## 2.2 "Fingerprints" of Digital Cameras

The "fingerprints" of a digital camera refer to unique traces left on the images it captures, which can identify the specific camera that took a particular photo. These fingerprints are typically based on physical and digital characteristics of the camera, as depicted in Figure 2(a). These unique traces include hardware-based identifiers like sensor pattern noise [38] and lens distortion patterns [14, 64] and software-based ones such as color filter array (CFA) interpolation [5, 31], demosaicing artifacts [6, 68] and EXIF metadata [10, 62].

**Sensor Pattern Noise.** Every image sensor has small manufacturing imperfections, as depicted in Figure 2(b), which can produce a unique pattern of noise, known as the Photo-Response Non-Uniformity (PRNU). These imperfections result from two main factors. (1) Variations in the quality of the raw materials used to manufacture the sensor can lead to non-uniform sensitivity across pixels (i.e., *material impurities*). (2) Small differences in the physical size of photodiodes and other microscopic components cause inconsistencies in light capture (i.e., *proximity effects*). Since PRNU is inherent to the sensor's physical properties, it cannot be entirely eliminated and remains a **persistent** characteristic, appearing in every image captured by the camera. The PRNU pattern also remains **consistent** across all images taken by the same camera [3]. Therefore, PRNU can be extracted from an image and compared with others to determine if they originate from the same device.

**Other Types of "Fingerprints".** Unlike PRNU, other types of "fingerprints", such as lens distortion patterns, image processing and compression artifacts, and EXIF metadata, can be easily modified or removed using image processing techniques. For example, as the camera's software, specifically the Image Signal Processing (ISP) module, is designed to enhance image quality by adjusting factors like sharpness, color, and noise reduction, it is highly sensitive to image processing operations that alter these same characteristics.

***Takeaway.*** While various unique camera characteristics can help detect image theft, *CAMPrints* prioritizes the PRNU pattern because it is consistent, resilient to software modifications, and remains persistent over time. This approach ensures reliable identification and traceability, even with post-processing or aging of the images.

***In the remainder of this paper, we refer to camera fingerprints as the PRNU or the noise pattern found in images.***

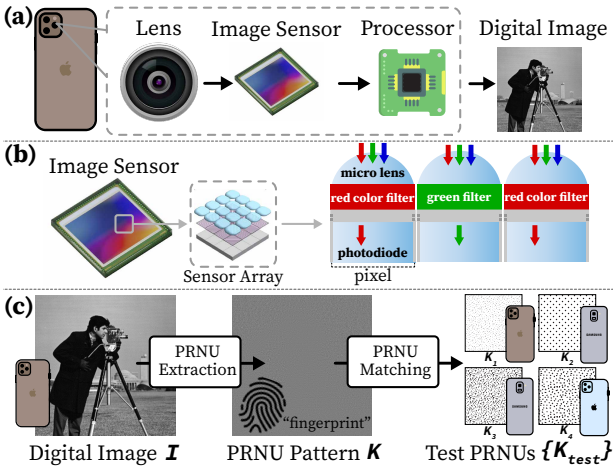



**Figure 2: (a) depicts physical and digital components of a camera that may leave unique traces on digital images. (b) depicts sources of manufacturing imperfections in an image sensor. (c) depicts the pipeline to extract and match camera fingerprints. Note that the Test PRNUs are not actual PRNU patterns and are provided for illustrative purposes to highlight distinct patterns.**

## 2.3 Physics Primer of PRNU

Various physical and operational factors, such as light intensity, resolution, compression, shutter speed, motion, color sensitivity, and white balance, can influence PRNU patterns. However, many previous works [25, 38, 50] have demonstrated that PRNU patterns can still be reliably extracted and matched from the RGB images. This demonstrates the effectiveness of the PRNU-based approach despite these potential variations. We now present the physics behind the PRNU-based approach.

**PRNU Modeling.** A camera's PRNU pattern, $K$, left in a digital image, $I$, can be modeled as a multiplicative noise – i.e., the amount of the noise depends on the pixel intensity of the image sensor – illustrated by the equation: $I = I_0(1+K)+\epsilon$. $I_0$ denotes the theoretical noise-free image (i.e., the ground truth scene to be captured) and $\epsilon$ denotes random noises from other sources, such as read noise, dark current, and photon shot noise [67]. Figure 2(c) depicts the pipeline to extract the PRNU pattern, $K$, from a digital image, $I$, and match a set of test PRNU patterns, $\{K_{test}\}$, with $K$ to identify which images are captured by the same camera.

**PRNU Extraction.** To estimate $K$ from an image, we first denoise the image using a function, $Denoise(\cdot)$, to approximate the noise-free image, $I_0 = Denoise(I)$. Many techniques have been proposed to denoise an image, including signal processing techniques such as wavelet-based methods [37, 38], and deep-learning approaches [17, 45, 60, 79]. Then, we compute the noise residual, $W = I - Denoise(I) = K \times Denoise(I) + \epsilon$. As $\epsilon$ usually follows

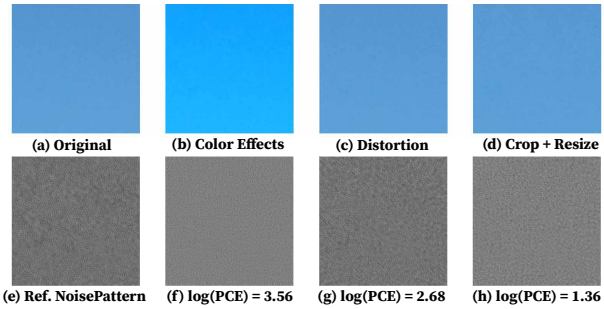



**Figure 3: (a) shows a sample image of a clear sky to effectively visualize PRNU patterns. (b)–(d) display images processed using *Color Effects*, *Distortion*, and *Crop+Resize*, respectively. (e)–(h) present the corresponding extracted noise patterns along with their matching scores (i.e., $log(PCE)$) compared to the reference noise pattern. We advise the readers to view this figure in color.**

a Gaussian distribution, we can use maximum likelihood estimation [56] to approximate $K$ using the equation: $\hat{K} = \frac{\sum_I W_I I_I}{\sum_I I_I I_I}$.

**PRNU Matching.** To compare a reference PRNU, $K_{ref}$, and a test PRNU, $K_{test}$, Peak-to-Correlation Energy (PCE) is a commonly used metric for assessing their similarity [38, 58]. PCE is derived from the Normalized Cross-Correlation (NCC) [80] between the two PRNU patterns. While NCC quantifies the alignment and overlap of the patterns based on pixel intensities, PCE refines this measurement by normalizing the peak correlation value against the energy of the surrounding correlation map. This makes PCE more resilient to variations in image content and size than NCC. A PCE value above a predefined threshold indicates that the PRNUs likely originate from the same camera. Therefore, selecting an appropriate threshold is critical for balancing the trade-off between false positives and false negatives in PRNU matching.

**Limitations of PCE.** Using PCE as a similarity metric necessitates proper spatial alignment of PRNU patterns, but geometric transformations, such as rotation and perspective transformation, can lead to misalignment of PRNU patterns by altering pixel locations. Figure 3 illustrates the impact of these transformations. To effectively visualize the noise patterns, we start with a sample image of a clear sky (i.e., Original). The image is then processed using three representative operations: *Color Effects*, *Distortion*, and *Crop+Resize*, with the resulting images shown in Figure 3(b)–(d), respectively. Subsequently, Figure 3(e)–(h) displays the corresponding extracted noise patterns along with their matching scores (i.e., $log(PCE)$) compared to the reference noise pattern. We observe a significant drop in the matching score from 3.56 to 1.36 as the extent of image modifications increases. Note that $log(PCE) < 3$ usually indicates that two images are taken by different devices. In addition, other image processing operations, like noise addition and reduction, also affect PRNU patterns significantly by altering the embedded noise signal. As a result, PCE demonstrates limited robustness when images undergo image processing operations, highlighting the necessity for *CAMPrints* to address these challenges.

## 3 Core Idea and Feasibility Study

The core idea of *CAMPrints* lies in the assumption that the PRNU patterns of images taken by the same sensor are similar and those by different sensors are distinguishable, even after extensive image processing operations. To mitigate the impact of these operations, *CAMPrints* utilizes deep learning to learn an encoding function, $Enc(\cdot)$, to transform a PRNU pattern to a *lower-dimensional embedding*. The *embedding distance* will be small if the images are taken by the same sensor and will be large otherwise. Data augmentation of **representative** image processing operations ensures that the deep learning model can adapt to images processed by various types of operations.

**Extended PRNU Modeling.** Since the PRNU modeling presented in §2.3 does not account for image processing operations, we propose an extended PRNU model given by $I = T(I_0(1+K)+\epsilon)$, where $T$ is the product of transformation matrices representing image processing operations, defined as $T = \prod T_i$ (note that $T_i$ can be either pre-multiplied or post-multiplied). After denoising the image, $T$ remains in the PRNU pattern, as derived from the equation: $W = TK \times Denoise(I) + T\epsilon$. This is demonstrated by the example patterns shown in Figure 3, where image operations such as *Distortion* and *Crop+Resize* are applied to the original image. By applying the same transformation matrix (i.e., $T_{distortion}$ or $T_{crop+resize}$) to the reference noise pattern, we can obtain noise patterns similar to those in Figure 3(g) and (h). Therefore, with this extended PRNU modeling, calculating $\hat{K}$ using the equation $T\hat{K} = \frac{\sum_I W_I I_I}{\sum_I I_I I_I}$ becomes extremely challenging, as $T$ is unknown and may alter the distribution of $T\epsilon$.

**Rationale Behind Core Idea.** With the extended PRNU modeling, directly estimating $K$ is extremely challenging. Instead, we adopt a different approach by assuming that the noise residue of images taken by the same sensor is similar and those by different sensors are distinguishable. We propose an encoding function, $Enc(\cdot)$, designed to suppress irrelevant details of $W$ that are susceptible to $T$, ensuring that $Enc(W_1)$ and $Enc(W_2)$ are similar when $W_1$ and $W_2$ are obtained from the same camera. Since $Enc(\cdot)$ essentially serves as the "universal" inverse function, $T^{-1}$, to every possible $T$, the key to its effectiveness lies in ensuring that $Enc(\cdot)$ captures as many types of image processing operations as possible.

**Challenge and Solution.** The primary challenge, however, is the virtually infinite range of possible image processing operations and their combinations, making it impractical to account for all of them explicitly. To address this complexity, *CAMPrints* identifies ***a small, yet representative set*** of image processing operations that can achieve ***high coverage across the broad spectrum of potential transformations***. Since image processing operations primarily affect pixel values, spatial locations, or both, we can streamline the coverage of these operations by focusing on the possible combinations of pixel value and location changes.

**Feasibility Study.** We conduct a feasibility study to evaluate our hypothesis that a small but representative set of operations exists. We find this set by quantifying the *coverage* of each operation. We analyze the impact of each operation on pixel values and spatial locations in both the image and noise pattern across various parameters while preserving overall similarity to the original. Specifically, we measure pixel value changes using *Kullback-Leibler Divergence*

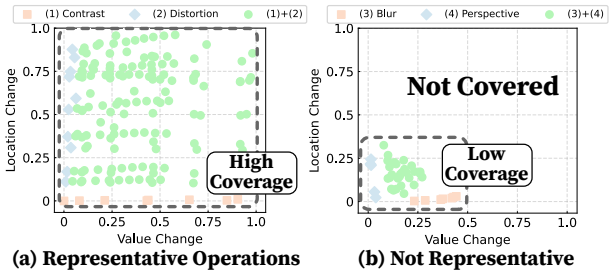



**Figure 4: (a) highlights representative operations that produce a broad range of pixel value and location changes, demonstrating high coverage. (b) exemplifies operations with limited coverage.**

*(KLD)* [21, 48] by computing the pixel intensity distributions (i.e., normalized pixel histograms) of the original and processed images. *KLD* quantifies how much the processed image's pixel distribution deviates from the original's. We measure pixel location changes using *optical flow* [4] by estimating motion vectors of pixels between the original and processed images. The magnitude of these vectors reflects how much the pixel locations have shifted due to the operation.

**Results and Insights.** Figure 4 illustrates the impact of example image processing operations. Figure 4(a) highlights representative operations that produce a broad range of pixel value and location changes, demonstrating high coverage. In contrast, Figure 4(b) exemplifies operations with limited coverage. Since the effects of certain operations can be additive (e.g., combining operations (1) and (2) to achieve a broader range of changes), it is possible to optimize coverage along each dimension (value or location). Additionally, achieving high coverage for both the image and the PRNU pattern is essential, as image processing operations may affect them differently, resulting in four distinct dimensions. Consequently, we optimize four dimensions by identifying a core set of four representative operations: (1) Contrast, (2) Distortion, (3) Add Noise, and (4) Crop+Resize. *CAMPrints* leverages this set to train its deep learning models (see §4.5), significantly simplifying the problem while maintaining robustness. However, it is important to note that our approach focuses on quantifying suitable representative operations in terms of improving the *coverage*, while alternative sets of operations may also be viable.

# 4 System Design

In this section, we present *CAMPrints*'s system design.

## 4.1 System Model

**System Goal and Requirements.** *CAMPrints* aims to determine whether an image suspected of image theft is captured by the same camera as the reference images provided by the copyright owner, utilizing the unique camera fingerprints embedded in the images. Specifically, images taken by different camera instances but the same make-and-model should be flagged as *benign images* even when they capture similar scenes (e.g., same landmarks). On the contrary, reusing the same image with editing should be flagged as *image theft*. We design *CAMPrints* to satisfy the following requirements: (1) accurately determine whether the camera fingerprints of the test image match those of the reference images (*device-level accuracy*); (2) process each image within a few seconds (*speed*); (3) adjust the system settings flexibly to prioritize minimizing either the false positive rate or the false negative rate (*usability*).

**Threat Model.** The attackers (i.e., copyright violators) aim to reuse or alter the image in a way that avoids detection of image theft while preserving its artistic style, structure, and core content. They can use image editing software and test against detection methods but will not employ transformations that completely alter or regenerate the image. This excludes diffusion models and extreme processing, such as severe down-sampling. The modified image must retain similarity within a defined threshold.

## 4.2 System Overview

*CAMPrints* determines whether the camera fingerprints of a test image, intentionally modified by the copyright violator, match those of the reference images. *CAMPrints* utilizes the Photo-Response Non-Uniformity (PRNU), a distinct noise pattern caused by minute manufacturing imperfections in image sensors (see §2.2). As depicted in Figure 5, *CAMPrints* is divided into the *Training* and *Verification* phases.

① **Training Phase.** The *Training Phase* is a one-time process where social media and photo-sharing platforms together with copyright monitoring service providers train and deploy *CAMPrints*. They collect online public datasets containing images and their corresponding camera labels (i.e., device models and IDs) to train the deep learning models for extracting and matching features from PRNU patterns. During this phase, *CAMPrints* first processes the dataset in the *Data Processing* module (§4.3) to generate image triplets to improve data efficiency and simulate potential image modifications by applying a predefined set of image processing operations. In the *Noise Pattern Extractor* module (§4.4), *CAMPrints* fine-tunes a deep learning model, pre-trained for image denoising, to extract noise patterns (i.e., PRNU patterns) from the images. Finally, in the *Embedding Extractor* module (§4.5), *CAMPrints* learns to extract distinctive features from noise patterns before and after image processing operations, while filtering out irrelevant details susceptible to alteration by these operations.

② **Verification Phase.** In the *Verification Phase*, social media and photo-sharing platforms, and copyright monitoring services first scan their databases for images suspected of theft and then input these images to *CAMPrints* for detailed filtering and verification. *CAMPrints* requires the user to provide a set of images captured by the same camera to serve as references for camera fingerprint matching. For each suspected image, *CAMPrints* begins by using the trained models in the *Noise Pattern Extractor* module (§4.4) to extract the noise patterns from both the reference and the suspected image. It then generates the corresponding embeddings in the *Embedding Extractor* module (§4.5). Finally, in the *Camera Fingerprint Matching* module (§4.6), *CAMPrints* calculates the similarity score, *Sim*, between the mean of the reference embeddings and the test embedding, comparing it to a predefined threshold. If *Sim* is above the threshold, *CAMPrints* outputs a match, indicating

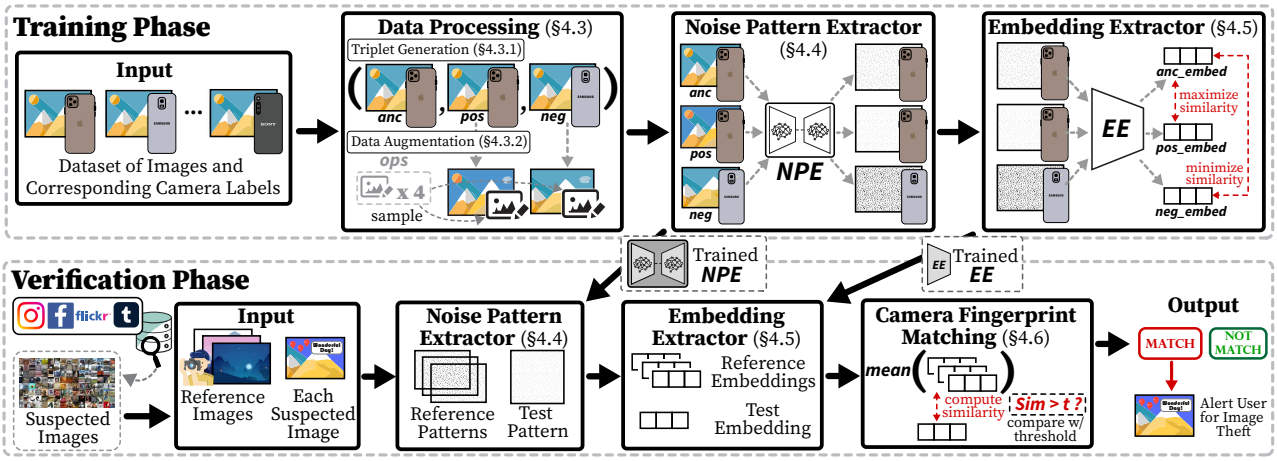



**Figure 5: Figure depicts an overview of the system design. *CAMPrints* is divided into *Training* and *Verification* phases. The *Training Phase* is a one-time process where social media and photo-sharing platforms together with copyright monitoring service providers train and deploy *CAMPrints* using online public datasets. In the *Verification Phase*, these platforms and services first scan their databases for images suspected of theft and then input these images to *CAMPrints*. It extracts and matches the embeddings of user-provided reference images and the suspected image using the trained models. *CAMPrints* alerts the user when a match occurs, indicating potential image theft.**

that the suspected image likely originates from the same camera as the reference images, suggesting a potential image theft.

## 4.3 Data Processing

*Data Processing* module processes the online public dataset containing images and their corresponding camera labels (i.e., device models and IDs). It consists of two sub-modules. First, the *Triplet Generation* sub-module (§4.3.1) generates image triplets, $(anc, pos, neg)$, consisting of an anchor, a positive, and a negative image. We ensure that the anchor and positive images are captured by the same camera, while the negative image is taken by a different camera than the anchor. Second, the *Data Augmentation* sub-module (§4.3.2) simulates potential image modifications by applying an image processing operation, selected from a predefined set of image processing operations (see §3), to positive and negative images.

*4.3.1 Triplet Generation.* Although images are widely available online, their device information, such as camera specifications, is often missing or difficult to obtain, and collecting large datasets for every camera is impractical. As *CAMPrints* does not require device information from users, we need to efficiently utilize the limited *labeled data* publicly available online. Hence, we create image triplets, $(anc, pos, neg)$, which allow for training on smaller datasets through different permutations of relative comparisons. Training with triplets optimizes a similarity metric by maximizing the similarity between *anc* and *pos* while minimizing it with *neg*. Specifically, we form triplets by selecting all images in the dataset as *anc* and carefully choosing *pos* and *neg*. We first identify all images taken by the same camera as *anc* and compute the content similarity using a pre-trained ResNet-50 [43]. The image with the lowest content similarity is chosen as *pos* to prevent the model from overfitting on image content, allowing it to focus on the camera's noise pattern. For *neg*, we randomly select $M$ cameras different from the camera of *anc* (with $M = 5$ empirically set to balance data variety and training time) and choose the image with the highest content similarity from each camera as *neg*.

*4.3.2 Data Augmentation.* In real-world scenarios, images may undergo various image processing operations such as compression, resizing, cropping, noise addition, and color adjustments. By incorporating these transformations during training, data augmentation exposes the model to a wide range of image variations, helping it learn to identify noise patterns that remain consistent despite such modifications. Given the virtually infinite number of possible image processing operations and their combinations, it is impractical to account for all of them explicitly. Instead, we identify a small yet representative set of image processing operations that provide broad coverage of potential transformations (see §3). This set includes four image processing operations: *color effects*, *barrel distortion*, *Gaussian noise addition*, and *crop and resize*. We apply these image processing operations to *pos* and *neg* images.

## 4.4 Noise Pattern Extractor

In this module, *CAMPrints* aims to extract the camera's noise pattern (i.e., PRNU pattern) from an input image.

**Training Phase.** In this phase, *CAMPrints* first learns how to denoise the input image, $I$ by the denoising function, $Denoise(\cdot)$, and then obtains the noise pattern by subtracting the denoised image, $W = I - Denoise(I)$. Specifically, *CAMPrints* fine-tunes a deep learning model [71] initialized with pre-trained weights for the image denoising task[1], which removes noise from corrupted images while preserving their essential details and structures. We name this model as *Noise Pattern Extractor* ($NPE$). $NPE$ is initially trained using pairs of noisy and clean images, where it learns to

[1]The model is pre-trained using datasets that contain high-resolution images along with their corresponding noisy versions (i.e., adding synthetic Gaussian noise).

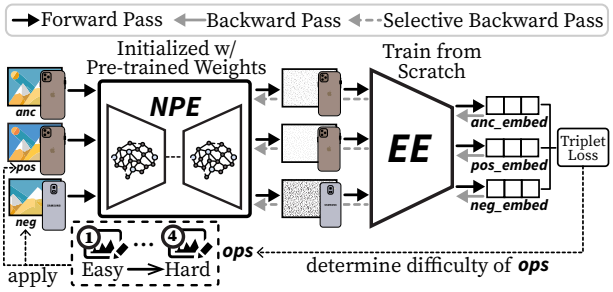



**Figure 6: Figure depicts the *Curriculum Learning* strategy where operations are applied to *pos* and *neg* images based on their learning difficulty. We dynamically determine the difficulty using average *Triplet Loss* and pixel value and location measures (see §3).**

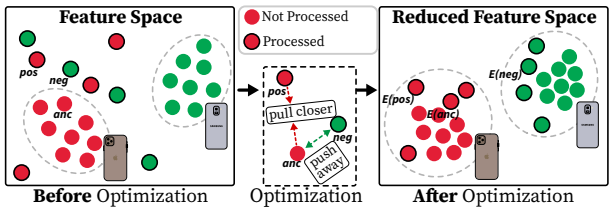



**Figure 7: Figure depicts the effect of *Embedding Extractor* (*EE*; see §4.5) in the feature space.**

predict the clean image from the noisy input by minimizing the Mean Squared Error (MSE). As depicted in Figure 6, we utilize the pre-trained weights and continue fine-tuning the model with a *selective backward pass*. During this process, we only backpropagate the loss to update the weights of the *NPE* in epochs when the image processing operations are particularly challenging or when the performance of the *Embedding Extractor* (*EE*; see §4.5) has nearly saturated (i.e., the loss plateaus). This approach ensures that the model focuses on refining the *NPE* when the difficulty of the task increases or when further improvements to *EE* are minimal.

**Verification Phase.** *CAMPrints* processes the reference images and the suspected image to obtain corresponding noise patterns, $\{W_{ref}\}$ and $W_{test}$.

## 4.5 Embedding Extractor

In this module, *CAMPrints* extracts distinctive features from noise patterns, $W$, before and after various image processing operations, filtering out irrelevant details susceptible to alteration by these operations. As depicted in Figure 7, image processing operations can shift noise patterns away from their original clusters, with unprocessed patterns represented by filled circles and processed ones by outlined circles. This shift makes matching challenging, as distances in the feature space can become misleading (e.g., *anc* - *pos* may appear farther than *anc* - *neg* due to image processing operations). To address this issue, *CAMPrints* employs an embedding network, $E(\cdot)$, to optimize the feature space, "pulling" *anc* and *pos* closer while "pushing" *anc* and *neg* apart. After training, patterns from the same camera form tighter clusters (e.g., $E(anc)$ - $E(pos)$ is closer than $E(anc)$ - $E(neg)$), improving camera matching accuracy.

**Training Phase.** In this phase, we train an embedding network designed to filter out irrelevant details in noise patterns that are easily altered by image processing operations while preserving consistent features before and after these operations. The primary challenge lies in the fact that different image processing operations introduce shifts in the feature space with varying directions and magnitudes. Training the model with all types of image processing operations simultaneously can confuse the network and hinder convergence. To address this, we employ the ***Curriculum Learning*** strategy [69], where the model learns from data samples in a progression of increasing difficulty, similar to how humans learn, to enhance both convergence and performance. This approach is illustrated in Figure 6. Specifically, we first rank image processing operations based on their impact on pixel values and spatial locations (see §3). Operations that affect both pixel values and spatial locations, such as cropping and resizing, are considered more challenging. In contrast, operations like *geometric distortion* primarily affect spatial locations, making them less complex (see Figure 4). During training, we dynamically update the difficulty metric by computing the average triplet loss for each epoch. Every four epochs, we re-arrange the order of applying image processing operations based on this updated difficulty metric to ensure an optimal learning progression.

**Verification Phase.** *CAMPrints* processes the noise patterns, $\{W_{ref}\}$ and $W_{test}$ to obtain embeddings, $\{embed_{ref}\}$ and $embed_{test}$.

## 4.6 Camera Fingerprint Matching

*CAMPrints* processes reference embeddings, $\{embed_{ref}\}$, and a test embedding, $embed_{test}$, as input. It begins by aggregating the reference embeddings through their mean, $embed_{ref} = mean(\{embed_{ref}\})$. Next, *CAMPrints* computes a similarity score, *Sim*, between the aggregated reference embeddings and the test embedding. The similarity score is calculated using a combination of cosine similarity, *cosine_sim*, and Euclidean distance, *dist*, defined as $Sim = \alpha \cdot cosine\_sim + \beta \cdot dist$. Finally, *CAMPrints* compares *Sim* to a predefined threshold, $t$. If $Sim > t$, the model outputs a match, indicating that the test image likely originates from the same camera as the reference images, suggesting potential image theft.

# 5 Evaluation

This section evaluates *CAMPrints* through comprehensive real-world experiments.

## 5.1 Experiment Setup

*5.1.1 Dataset and Devices.* We evaluate *CAMPrints* with *SOCRatES* dataset [30], a benchmark dataset for source camera identification – i.e., identifying which source camera captures the input image. We ensure ***at least three different instances*** per make-and-model to evaluate *CAMPrints*'s effectiveness in distinguishing ***device-level*** camera fingerprints. We have more than 4,000 images from 36 different devices from eight make-and-model combinations across five popular smartphone brands, namely Apple, Samsung, Motorola, LG, and Sony, as depicted in Figure 8(a). We conduct this study upon the approval of our institution's Institutional Review Board.

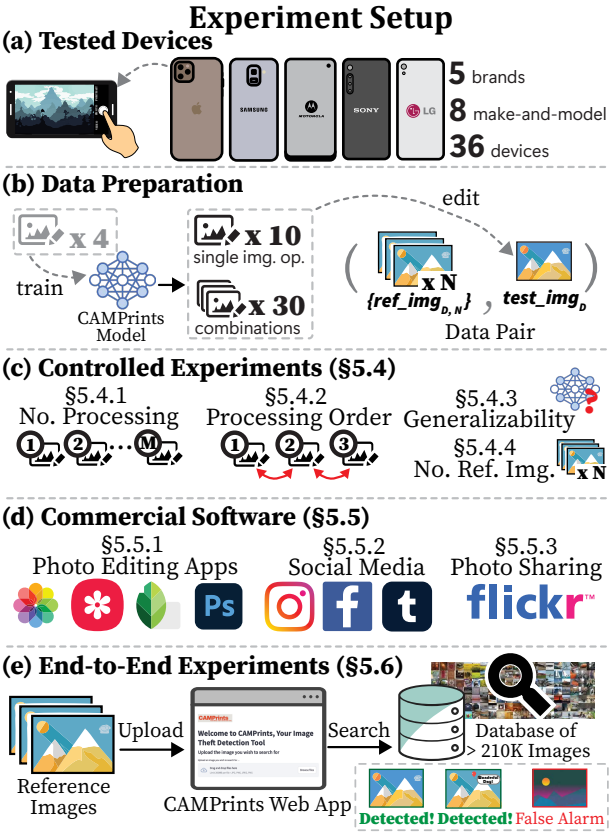



**Figure 8: Figure depicts *CAMPrints*'s evaluation setup and experiments. (a) summary of tested devices. (b) data preparation procedure. (c) controlled experiments. (d) experiments using commercial software. (e) setup and procedure of the end-to-end experiments.**

*5.1.2 Data Preparation.* Figure 8(b) depicts a flowchart of our data preparation. For each device, we randomly split the images into 80% for training and 20% for testing. We train the model using ***only four*** types of image processing operations (see §4.3), namely *color effects*, *barrel distortion*, *Gaussian noise addition*, and *crop and resize*. For testing, we pair different sets of $N$ reference images, $\{ref\_img\}$, and test images, $test\_img$, where reference and test images can be taken from the same or different devices. We apply image processing operations to all test images with *random parameters* (e.g., cropping ratio can be a random value between 0.5 and 1). In total, we evaluate *CAMPrints* with ten single operations and 30 combinations of operations on over 4,000 images.

*5.1.3 Baseline.* We compare *CAMPrints* with three baseline methods, namely DRUNET [13], MWDCNN [72], and Wiener [9]. DRUNET and MWDCNN are state-of-the-art academic proposals, while the Wiener method is commonly adopted in commercial image forensic products. These baseline methods extract the noise patterns from both reference images and test images and then compare them using the Peak-to-Correlation Energy (PCE) score (see §2.3). A large score (generally $> 10^2$) indicates that the images belong to the same camera. For multiple reference images, they adopt the maximum likelihood estimation (MLE) approach to average the noise patterns.

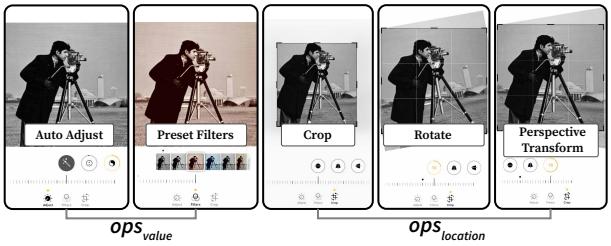



**Figure 9: Figure depicts *examples* of image processing operations on iOS categorized into those affecting pixel values ($ops_{value}$) and locations ($ops_{location}$). Combinations affect *both* values and locations ($ops_{both}$). Note that we evaluate *CAMPrints* with *more* operations.**

*5.1.4 Evaluation Metrics.* We use the following metrics to evaluate *CAMPrints*. **True Positives (*TP*)** refers to the number of actual *image theft cases* accurately detected (i.e., reference images and test images captured by the same camera). **True Negatives (*TN*)** is the number of *benign images* accurately rejected (i.e., reference and test images captured by different cameras). **False Negatives (*FN*)** refers to the number of actual *image theft cases* incorrectly rejected. **False Positives (*FP*)** is the number of *benign images* incorrectly detected as *image theft cases*, indicating *false alarms*. Furthermore, as *CAMPrints* and baseline methods are threshold-based, we use the Receiver Operating Characteristic (ROC) curve to measure the true positive rate (***TPR***) and false positive rate (***FPR***) at every possible threshold. ***TPR*** measures the proportion of actual *image theft cases* correctly detected. $TPR = \frac{TP}{TP+FN}$. On the contrary, ***FPR*** measures the proportion of *benign images* mistakenly identified as *image theft cases*. $FPR = \frac{FP}{FP+TN}$. We use the area under the curve (***AUC***) to measure the overall performance and effectiveness of *CAMPrints* and baseline methods in detecting image theft with high TPR and low FPR.

## 5.2 Overall Performance

We evaluate *CAMPrints* with a total of 40 image processing operations[2] across three categories – i.e., those affecting pixel values ($ops_{value}$) and pixel locations ($ops_{location}$), and combinations of these image processing operations, which affect *both* pixel values and locations ($ops_{both}$). Figure 9 depicts examples of image processing operations. We compare *CAMPrints* with three baseline methods, namely DRUNET, Wiener, and MWDCNN. We prepare a balanced dataset consisting of 50% *image theft cases* and 50% *benign images* (i.e., images with similar scenes but taken from different devices) to ensure unbiased evaluation metrics.

**Results.** Figure 10 depicts the ROC curve of *CAMPrints* alongside baseline methods for all 40 image processing operations. The analysis highlights that *CAMPrints* consistently outperforms baseline methods in detecting *image theft cases*, as evidenced by its high TPR and relatively low FPR. For instance, when the acceptable FPR is set to 0.2, *CAMPrints* achieves a TPR of 0.87, whereas DRUNET,

[2]Comparable to iPhone Photos app offering 15 - 20 distinct operations

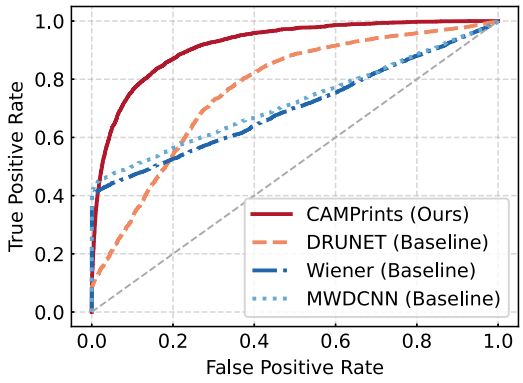



**Figure 10: Figure depicts the ROC Curve of *CAMPrints* and baseline methods.**

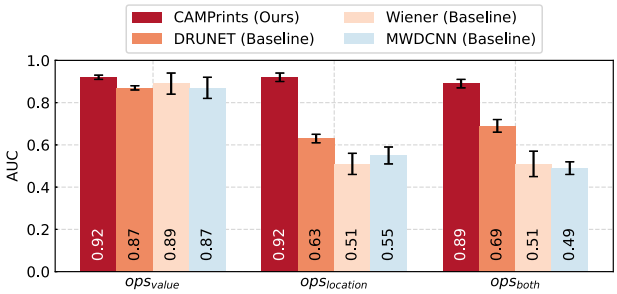



**Figure 11: Figure depicts *CAMPrints*'s performance compared to baseline methods.**

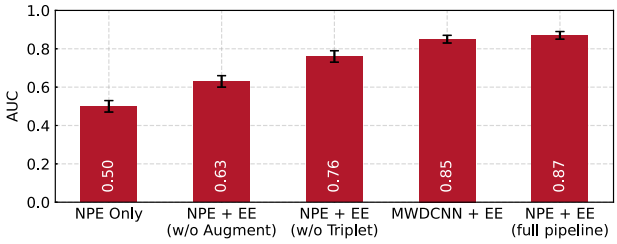



**Figure 12: Figure depicts performance with different model architectures and training strategies.**

MWDCNN, and Wiener only reach TPRs of 0.55, 0.56, and 0.54, respectively. Conversely, to achieve a TPR above 0.95, *CAMPrints* maintains an FPR of approximately 0.5, while the baselines result in an FPR of 1.0. While an FPR of 0.5 may appear high, it is acceptable in an *end-to-end setting* (see §5.6), where we integrate *CAMPrints* with Reverse Image Search. In this case, *CAMPrints* generates an average of only 3.9 false positives per test image when searching through an image database of over 210K images. This implies that for every accurate detection of an image theft case, the user typically reviews only five images, significantly alleviating user burden.

Figure 11 depicts the performance of *CAMPrints* when different categories of operations are applied. Specifically, *CAMPrints* achieves average AUC of 0.92, 0.92, and 0.89 for $ops_{value}$, $ops_{location}$, and $ops_{both}$, respectively. While baseline methods may produce AUC similar to *CAMPrints* for $ops_{value}$, they fall notably behind in the cases of $ops_{location}$ and $ops_{both}$. We observe substantial margins of 0.36 and 0.33 on average for $ops_{location}$ and $ops_{both}$, respectively. We attribute *CAMPrints*'s performance to its *Embedding Extractor* (see §4.5). It captures global geometric patterns of camera fingerprints even after extensive image manipulations – such as rotation, cropping, and perspective transformations – which significantly changes the appearance of objects and scenes in the image. In contrast, baseline methods, which rely on a cross-correlation-based approach, focus primarily on local patterns – such as relative distances between noise pixels – which can be easily altered by $ops_{location}$. Therefore, *CAMPrints* is robust against different image processing operations.

## 5.3 Ablation Study

We evaluate the effectiveness of model design and training strategy. We compare the following alternative design choices: (1) **NPE Only:** We use *Noise Pattern Extractor* to extract the noise patterns and compute the PCE score. (2) **NPE + EE (w/o Augment):** We use both *Noise Pattern Extractor* and *Embedding Extractor* to obtain the embeddings of noise patterns and calculate the embedding distance. The *Embedding Extractor* is trained using non-processed images only. (3) **NPE + EE (w/o Triplet):** The *Embedding Extractor* is trained using processed images without image triplets. We replace the *Triplet Loss* with a *Cross Entropy Loss* during the training. (4) **MWDCNN + EE:** We combine the baseline method, MWDCNN, and our trained *Embedding Extractor*. (5) **NPE + EE (full pipeline):** The full pipeline of *CAMPrints*.

**Results.** Figure 12 illustrates *CAMPrints*'s performance under various design configurations. The results reveal a consistent improvement in AUC as additional training strategies are incorporated. Notably, *CAMPrints*'s full pipeline, NPE + EE (full pipeline), achieves an AUC of 0.87, significantly surpassing NPE + EE (w/o Augment) and NPE + EE (w/o Triplet), which attain AUC of 0.63 and 0.76, respectively – representing substantial gains of 0.24 and 0.11. This highlights the effectiveness of *CAMPrints*'s *Image Generator* module (see §4.3) in helping the model recognize and compare camera fingerprints before and after image processing operations. Furthermore, MWDCNN + EE achieves an AUC of 0.85, coming remarkably close to the performance of the full *CAMPrints* pipeline, with a marginal difference of only 0.02. This highlights the strength of *CAMPrints*'s *Embedding Extractor* (see §4.5) in effectively capturing the global geometric patterns inherent in camera fingerprints. Additionally, our *Embedding Extractor* demonstrates compatibility with other approaches, including baseline methods, further showcasing its versatility and integration potential.

## 5.4 Controlled Experiments

We evaluate the robustness of *CAMPrints* against different usage conditions, as depicted in Figure 8(c). We adopt the same balanced dataset in §5.2.

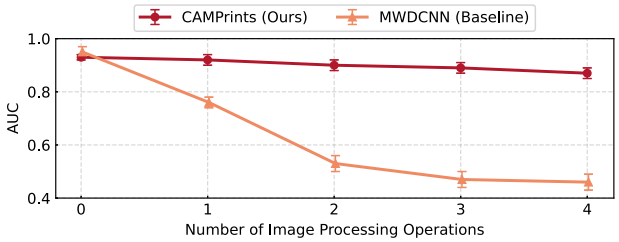



**Figure 13: Figure depicts the *CAMPrints*'s performance for different numbers of image processing operations.**

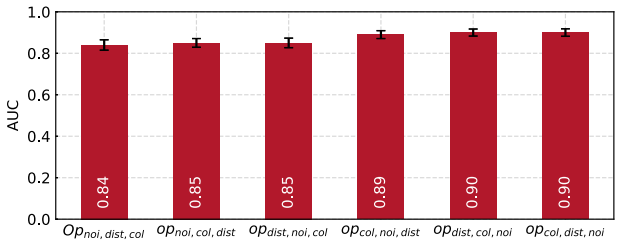



**Figure 14: Figure depicts the *CAMPrints*'s performance for different orders of image processing operations.**

*5.4.1 Number of Image Operations.* We analyze how increasing the ***number*** of distinct image processing operations affects *CAMPrints*'s performance. The combinations are selected from a representative set of operations (see §3).

**Results.** Figure 13 illustrates the performance of both *CAMPrints* and MWDCNN (Baseline). The results reveal a declining trend in performance for both as the number increases. This is expected as applying more operations alters and diminishes the camera fingerprints. However, *CAMPrints* exhibits only a slight performance decline from an average AUC of 0.93 to 0.87, whereas the baseline undergoes a steep drop. This highlights the robustness of *CAMPrints* even under extensive image modifications.

*5.4.2 Order of Image Operations.* We analyze the effect of the ***order*** of distinct operations. Using *color effects* ($op_{col}$), *barrel distortion* ($op_{dist}$), *Gaussian noise addition* ($op_{noi}$), we create six permutations (e.g., $op_{col,dist,noi}$ and $op_{noi,dist,col}$).

**Results.** As depicted in Figure 14, the order of image processing operations has minimal effect, with performance varying within a small standard deviation of 0.03. However, we note that *CAMPrints*'s performance tends to decline when $op_{noi}$ is applied before $op_{col}$ as $op_{col}$ tends to amplify the *Gaussian noises* introduced to the image.

*5.4.3 Generalizability to Unseen Operations.* Since *CAMPrints* employs a deep-learning approach, assessing its generalizability to unseen operations is crucial, particularly when training involves only four representative ones. We test *CAMPrints* on a diverse set of operations excluded during training: (1) $ops_{value}$ (unseen) includes iPhone preset filters, Gaussian blurring, median blurring, sharpening, and vignette addition; (2) $ops_{location}$ (unseen) includes flipping, rotation, resizing, and perspective transformations; (3) $ops_{both}$ (unseen) combines operations from $ops_{value}$ and $ops_{location}$.

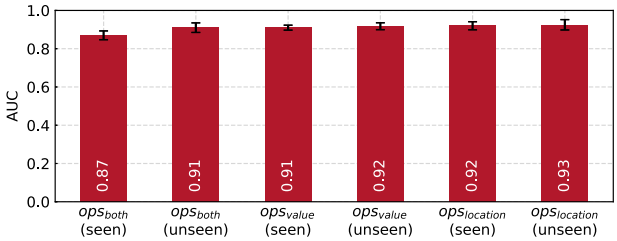



**Figure 15: Figure depicts *CAMPrints*'s performance for unseen image processing operations.**

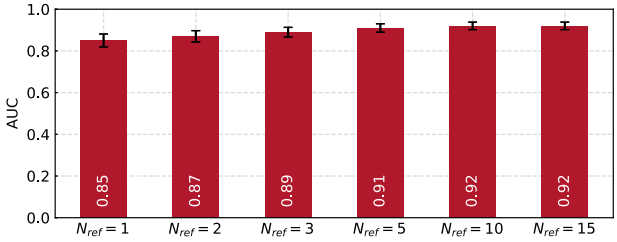



**Figure 16: Figure depicts *CAMPrints*'s performance when increasing the number of reference images.**

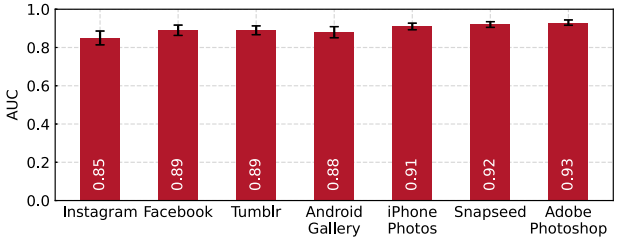



**Figure 17: Figure depicts *CAMPrints*'s performance when images are edited by social media platforms and commercial applications.**

**Results.** As depicted in Figure 15, *CAMPrints* exhibits comparable performance for both unseen and seen operations, with an average AUC of 0.92 and 0.90, respectively. This suggests that the representative operations employed during the training (see §3) effectively simulate the effects of a wide range of unseen ones, improving the model's generalizability.

*5.4.4 Varying Number of Reference Images.* In real-world scenarios, photographers often own multiple images taken with the same camera, which can serve as reference images. To evaluate *CAMPrints*'s performance under these conditions, we vary the number of reference images, $N_{ref}$.

**Results.** As illustrated in Figure 16, *CAMPrints*'s performance improves from an AUC of 0.85 with a single reference image to 0.92 when 15 images are used. Notably, *CAMPrints*'s performance plateaus at 0.92 when more than ten reference images are provided. This indicates that *CAMPrints* effectively aggregates camera fingerprint information from multiple reference images. Additional images beyond ten offer little to no further benefit.

## 5.5 Commercial Software

To evaluate the robustness of *CAMPrints* with real-world commercial products, we perform image editing using smartphone applications, social media platforms, and photo-sharing websites, as depicted in Figure 8(d).

*5.5.1 Smartphone Applications.* Smartphone applications are increasingly popular for automatically editing images. We evaluate *CAMPrints* using auto-adjust tools from various smartphone applications, including iPhone Photos, Android Gallery, Snapseed, and Adobe Photoshop. These tools automatically adjust parameters such as brightness, contrast, saturation, and sharpness to enhance the overall image quality with minimal user input. Android Gallery also applies appropriate cropping, rotation, or perspective transformations to improve the image's visual appeal.

**Results.** As depicted in Figure 17, *CAMPrints* achieves average AUC of 0.93, 0.92, 0.91, and 0.88 for Adobe Photoshop, Snapseed, iPhone Photos, and Android Gallery, respectively. The lower performance on Android Gallery is expected due to the combination of image processing operations affecting both pixel values and locations, whereas the other applications primarily focus on pixel value adjustments.

*5.5.2 Social Media Platforms.* We evaluate *CAMPrints* using images uploaded to social media platforms, including Instagram, Facebook, and Tumblr, simulating the scenario where a screenshot of a copyright-protected image is taken and shared on social media. We randomly select 50 test images from our dataset and upload screenshots of these images. We then download the images and compare them with their corresponding reference images.

**Results.** Figure 17 depicts *CAMPrints*'s performance, with average AUC of 0.89, 0.89, and 0.85 for Tumblr, Facebook, and Instagram, respectively. We attribute the performance drop on Instagram to its compression method, which reduces image details in the high-frequency domain, particularly the noise pixels critical for identifying camera fingerprints.

*5.5.3 Photo-sharing Websites.* We evaluate *CAMPrints* using images collected from Flickr, one of the most popular photo-sharing platforms for professional and amateur photographers. The images on Flickr are often heavily edited to reflect photographers' aesthetic preferences. We obtain the ground truth device information from the EXIF metadata. In total, we gather over 200 images from 25 different devices, including the latest models like the iPhone 15 Pro Max and Samsung Galaxy S23 Ultra. We create a balanced dataset consisting of 50% *image theft cases* and 50% *benign images*, similar to the methodology in §5.2. Notably, these devices are not part of the training of *CAMPrints*.

**Results.** *CAMPrints* achieves an overall AUC of 0.91, which is comparable to the performance on devices seen during training (see §5.2). This demonstrates that *CAMPrints* can generalize effectively to unseen devices.

## 5.6 End-to-End Experiments

We explore the potential of integrating *CAMPrints* into existing *copyright monitoring services*, such as TinEye and Pixsy, which primarily utilize Reverse Image Search to identify similar images online using an image as input. Since Reverse Image Search APIs, like Google Image API, only query their own image databases, we resort to an open-source tool [47], which has demonstrated performance comparable to those of commercial APIs, to test our image dataset.

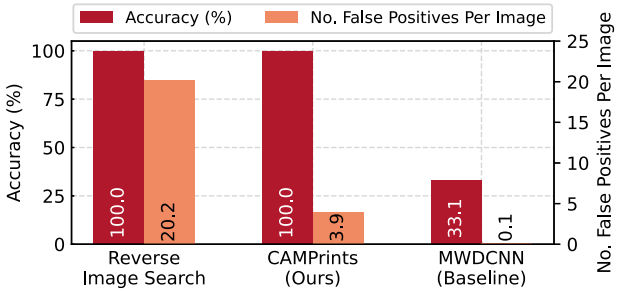



**Figure 18: Figure depicts the performance of Reverse Image Search, *CAMPrints*, and MWDCNN (Baseline) when searching for the edited images from an image database with above 210K images.**

***CAMPrints*'s Web Application.** We develop a web application to integrate *CAMPrints* with Reverse Image Search and compare its performance with MWDCNN (Baseline). Figure 8(e) provides an overview of *CAMPrints*'s web application. Specifically, the user first uploads a set of reference images to *CAMPrints*. Then, *CAMPrints* uses Reverse Image Search to scan a database of over 210K images. *CAMPrints* selects only those images with similar content. For each image, *CAMPrints* extracts and compares the camera fingerprint with those of the reference images. Any images identified as image theft are displayed to the user. For this experiment, we configure the web application ***not to tolerate any false negative***, prioritizing the detection of as many image theft cases as possible, even at the cost of potential false positives.

**Data Preparation.** We randomly select 16 devices from our test dataset and 10 images per device. The unprocessed versions of these images are used as reference images. To create the test images, we apply random combinations of image processing operations to the selected images.

**Evaluation Metrics.** *Accuracy* refers to the percentage of edited images correctly identified in the search results. *Number of False Positives Per Image* represents the number of benign images incorrectly included in the search results, divided by the total number of reference images.

**Results.** Figure 18 illustrates the performance of Reverse Image Search, *CAMPrints*, and MWDCNN (Baseline). As we configure the web application to prioritize accuracy (i.e., no false negative), both Reverse Image Search and *CAMPrints* achieve 100% accuracy, while the baseline only reaches 33.1%. In contrast, *CAMPrints* averages 3.9 false positives per image, significantly fewer than 20.2 for Reverse Image Search. This makes *CAMPrints* particularly effective in reducing the user's burden of reviewing search results. Additionally, the average time taken per image is 1.38s for Reverse Image Search, 2.36s for *CAMPrints*, and 5.14s for the baseline. Thus, with only a slight increase in processing time, incorporating *CAMPrints* into existing *copyright monitoring services* can significantly enhance usability while maintaining accuracy.

## 6 Discussion

We now present important discussion points of *CAMPrints*.

**Deployment Considerations.** We envision that social media and photo-sharing platforms could train and deploy *CAMPrints* using public datasets to detect *image theft* by comparing a suspected image with user-provided reference images. Furthermore, *CAMPrints* could be integrated into existing copyright monitoring services to enhance the detection accuracy of *image theft* and significantly reduce false positives.

**Complementary to Other Protection Methods.** *CAMPrints* could serve as a complementary solution to other copyright protection methods, such as watermarking, digital signatures, and blockchain technologies, because PRNU is inherently embedded in every photo and requires no additional effort from the photographer.

**Generalization to Other Transformations.** As *CAMPrints* utilizes data augmentation with a small representative set of image processing operations to learn transformation-equivariant representations of PRNU patterns, it generalizes well to a broad range of geometric transformations (see §5). For instance, although flipping and rotating an image alters the PRNU pattern to varying degrees, *CAMPrints* can still match the PRNU patterns after these modifications. *CAMPrints* could be further enhanced by incorporating additional or targeted transformations into the training set, though this would come at the cost of increased training time. Additionally, while adversarial attacks are not the primary focus of *CAMPrints*, they could impact its performance. Such attacks introduce subtle perturbations that, though imperceptible to humans, can significantly alter machine learning model outputs by exploiting architectural vulnerabilities. Future work could investigate integrating adversarial training techniques into *CAMPrints* to enhance its resilience against such attacks.

**Limitations.** Since *CAMPrints* is designed to enhance the robustness of PRNU extraction and matching, it is not intended for images that lack PRNU, such as AI-generated or computer-rendered images. Additionally, *CAMPrints* is less effective on heavily modified or regenerated images that deviate significantly from the original, such as those created with generative AI to depict similar scenes. However, this limitation is less critical as its primary use case revolves around scenarios where copyright violators often retain distinctive features of the original to preserve its economic and artistic value [15, 51, 70].

**Extension and Impact of *CAMPrints*.** We envision that *CAMPrints* could be further developed to assist forensic investigators in identifying potential suspects involved in the distribution of illegal or inappropriate online content [25]. By leveraging robust PRNU-based image analysis like *CAMPrints*, investigators could trace images back to their source devices, helping to establish a chain of custody and providing critical evidence in forensic examinations. This capability could be valuable in cases involving cyber crimes and deepfake investigations. Additionally, *CAMPrints* could contribute to the development and adoption of open standards for verifying the authenticity and provenance of digital content. Standards such as the Coalition for Content Provenance and Authenticity (C2PA) [12] aim to provide a transparent framework for tracking the origin and modifications of digital media. By integrating *CAMPrints* with such initiatives, platforms and law enforcement agencies could enhance digital content verification processes, mitigate misinformation, and improve trust in online media. This extension could support broader efforts to combat digital manipulation and strengthen content authentication in an increasingly AI-driven media landscape.

## 7 Related Work

We present related works of *CAMPrints*.

**Sensor Pattern Noise-Based Approaches.** A line of research explores the unique sensor pattern noise generated due to minor manufacturing imperfections in image sensors. These works rely on signal processing techniques like wavelet-based methods to analyze the noise patterns [1, 25, 32, 35–39], or deep learning-based approaches [11, 17, 45, 52, 60, 79] (e.g., convolutional neural networks and other architectures) to denoise images. Furthermore, another family of works [53, 74–76] focuses on efficient fingerprint matching and image retrieval in large databases. Some studies investigate the robustness of PRNU against image operations like cropping, resizing, and compression [8, 33, 34, 46, 50, 54, 78], but lack a practical solution, like *CAMPrints*, capable of handling a wide variety of image operations.

**Multi-Fingerprint Approaches.** Beyond sensor noise patterns, another line of research [24] explores using a combination of hardware and software-based camera fingerprints, such as Color Filter Array (CFA) patterns, Image Signal Processing (ISP) artifacts, and other distinctive camera-specific characteristics to enhance identification accuracy.

**Commercial Solutions.** Commercial solutions involve Reverse Image Search techniques [2, 28, 29, 65, 66], which uses deep learning to find visually similar images. Forensic tools like Amped Authenticate [27] and MOBILedit Forensic [55] implement various algorithms to detect and analyze camera-specific fingerprints. Other image forensic tools [49] analyze metadata embedded in digital images to uncover information about their origin, authenticity, and potential manipulations.

## 8 Conclusion

we present *CAMPrints*, a robust system for detecting image theft. *CAMPrints* verifies whether edited images found online contain camera fingerprints matching those of user-provided reference images. The system overcomes the challenges of identifying images altered by diverse image processing operations. We select a small yet representative set of operations by categorizing them based on their impact on pixel values and locations. A deep-learning model is trained to recognize and compare camera noise patterns pre- and post-editing. We conduct real-world evaluations involving 36 cameras across eight make-and-model combinations, along with over 40 operations applied to more than 4,000 images. *CAMPrints* achieves an average AUC of 0.92, significantly outperforming the state-of-the-art methods by up to 1.8 times.

## Acknowledgments

This research is partially supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) under grants RS-2024-00464269 and RS-2023-00277848.

## References


[1] KR Akshatha, A Kotegar Karunakar, H Anitha, U Raghavendra, and Dinesh Shetty. 2016. Digital camera identification using PRNU: A feature based approach. *Digital investigation* 19 (2016), 69–77.

[2] Hanaa Al-Lohibi, Tahani Alkhamisi, Maha Assagran, Amal Aljohani, and Asia Othaman Aljahdali. 2020. Awjedni: a reverse-image-search application. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 9, 3 (2020), 49.

[3] Chiara Albisani, Massimo Iuliani, and Alessandro Piva. 2021. Checking PRNU Usability on modern devices. , 2535–2539 pages.

[4] Andrea Alfarano, Luca Maiano, Lorenzo Papa, and Irene Amerini. 2024. Estimating optical flow: A comprehensive review of the state of the art. , 104160 pages.

[5] Sevinc Bayram, Husrev Sencar, Nasir Memon, and Ismail Avcibas. 2005. Source camera identification based on CFA interpolation. , III–69 pages.

[6] Sevinc Bayram, Husrev T Sencar, and Nasir Memon. 2008. Classification of digital camera-models based on demosaicing artifacts. *digital investigation* 5, 1-2 (2008), 49–59.

[7] berify. 2018. [Infographic] A Snapshot of Online Image Theft. https://berify.com/blog/image-theft/

[8] Nabeel Nisar Bhat and Tiziano Bianchi. 2022. Investigating inconsistencies in prnu-based camera identification. , 851–855 pages.

[9] Luca Bondi, Paolo Bestagini, and Nicolò Bonettini. 2021. Python porting of PRNU extractor and helper functions. https://github.com/polimi-ispl/prnu-python.

[10] Matthew Boutell and Jiebo Luo. 2004. Photo classification by integrating image content and camera metadata. , 901–904 pages.

[11] Andrea Bruno, Paola Capasso, Giuseppe Cattaneo, Umberto Ferraro Petrillo, and Riccardo Improta. 2023. A novel image dataset for source camera identification and image based recognition systems. *Multimedia Tools and Applications* 82, 8 (2023), 11221–11237.

[12] C2PA. 2024. About C2PA. https://c2pa.org/about/

[13] Mirco Ceccarelli, Francesco Argentieri, Alessandro Piva, Dasara Shullani, and Daniele Baracchi. 2021. PRNU extraction via denoiser based on convolutional neural network (DRUNet). https://github.com/ocrim1996/prnu-python.

[14] Kai San Choi, Edmund Y Lam, and Kenneth KY Wong. 2006. Automatic source camera identification using the intrinsic lens radial distortion. *Optics express* 14, 24 (2006), 11551–11565.

[15] Cassidy Cole. 2024. How Artists Can Use Copyright Law to Protect Their Work and Build Their Legacy. https://www.artworkarchive.com/blog/how-artists-can-use-copyright-law-to-protect-their-work-and-build-their-legacy

[16] Copytrack. 2019. How Big a Problem Is Copyright Infringement? https://nanpa.org/2019/04/24/how-big-a-problem-is-copyright-infringement/

[17] D. Cozzolino and L. Verdoliva. 2020. Noiseprint: A CNN-Based Camera Model Fingerprint. *IEEE Transactions on Information Forensics and Security* 15 (2020), 144–159. https://doi.org/10.1109/TIFS.2019.2916364

[18] DATAINTELO. 2024. Copyright Licensing Market Outlook. https://dataintelo.com/report/global-copyright-licensing-market

[19] Dealavo. 2024. Reverse image search and its role in e-commerce. https://dealavo.com/en/reverse-image-search/

[20] Dunja Djudjic. 2019. Over 2.5 billion online images are stolen every day, Copytrack reports. https://www.diyphotography.net/over-2-5-billion-online-images-are-stolen-every-day-copytrack-reports/

[21] Minh N Do and Martin Vetterli. 2000. Texture similarity measurement using Kullback-Leibler distance on wavelet subbands. , 730–733 pages.

[22] DomainTools. 2024. A Brief Comparison of Reverse Image Searching Platforms. https://www.domaintools.com/resources/blog/a-brief-comparison-of-reverse-image-searching-platforms/

[23] Stanley Dunthorne. 2023. Who's stealing my images? How to use Reverse Image Search. https://www.hallaminternet.com/who-is-stealing-my-images/

[24] John Entrieri and Matthias Kirchner. 2016. Patch-based desynchronization of digital camera sensor fingerprints. *Electronic Imaging* 28 (2016), 1–9.

[25] Eitan Flor, Ramazan Aygun, Suat Mercan, and Kemal Akkaya. 2021. PRNU-based source camera identification for multimedia forensics. , 168–175 pages.

[26] Vegard Flovik. 2020. Deep learning based reverse image search for industrial applications. https://towardsdatascience.com/deep-learning-based-reverse-image-search-for-industrial-applications-33ba4b0d32c4

[27] Marco Fontani. 2017. PRNU-based Camera Identification in Amped Authenticate. https://blog.ampedsoftware.com/2017/10/04/prnu-based-camera-identification-in-amped-authenticate

[28] Mathieu Gaillard and Előd Egyed-Zsigmond. 2017. Large scale reverse image search.

[29] Mathieu Gaillard, Elöd Egyed-Zsigmond, and Michael Granitzer. 2018. CNN features for reverse image search. *Document numérique* 21, 1 (2018), 63–90.

[30] Chiara Galdi, Frank Hartung, and Jean-Luc Dugelay. 2019. SOCRatES: A Database of Realistic Data for SOurce Camera REcognition on Smartphones. , 648–655 pages.

[31] Shang Gao, Guanshuo Xu, and Rui-Min Hu. 2012. Camera model identification based on the characteristic of CFA and interpolation. , 268–280 pages.

[32] Miroslav Goljan. 2008. Digital camera identification from images–estimating false acceptance probability. , 454–468 pages.

[33] Miroslav Goljan, Mo Chen, Pedro Comesaña, and Jessica Fridrich. 2016. Effect of compression on sensor-fingerprint based camera identification. *Electronic Imaging* 28 (2016), 1–10.

[34] Miroslav Goljan and Jessica Fridrich. 2008. Camera identification from cropped and scaled images. , 154–166 pages.

[35] Miroslav Goljan and Jessica Fridrich. 2012. Sensor-fingerprint based identification of images corrected for lens distortion. , 132–144 pages.

[36] Miroslav Goljan, Jessica Fridrich, and Mo Chen. 2010. Defending against fingerprint-copy attack in sensor-based camera identification. *IEEE Transactions on Information Forensics and Security* 6, 1 (2010), 227–236.

[37] Miroslav Goljan, Jessica Fridrich, and Mo Chen. 2010. Sensor noise camera identification: Countering counter-forensics. , 283–294 pages.

[38] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. 2009. Large scale test of sensor fingerprint camera identification. , 170–181 pages.

[39] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. 2010. Managing a large database of camera fingerprints. , 75–86 pages.

[40] Matt Golowczynski. 2020. Google reverse image search: Everything you need to know. https://smartframe.io/blog/google-reverse-image-search-everything-you-need-to-know/

[41] Google. 2024. Google Images. https://images.google.com/

[42] Geoff Harris. 2020. A lot of photographers find out about image theft when the culprits tag them in social media. https://amateurphotographer.com/technique/interviews/a-lot-of-photographers-find-out-about-image-theft-when-companies-tag-them-in-social-media/

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. , 770–778 pages.

[44] Aaron Hockley. 2019. Flickr Addresses Copyright Challenges in Partnership with Pixsy. https://techphotoguy.com/flickr-copyright-infringement-pixsy/

[45] Chen Hui, Feng Jiang, Shaohui Liu, and Debin Zhao. 2022. Source camera identification with multi-scale feature fusion network. , 6 pages.

[46] Massimo Iuliani, Marco Fontani, and Alessandro Piva. 2021. A leak in PRNU based source identification—questioning fingerprint uniqueness. *IEEE Access* 9 (2021), 52455–52463.

[47] Rahul Kad. 2023. Reverse Image Search Engine. https://github.com/RAHUL-KAD/Reverse-Image-Search-Engine.

[48] Roland Kwitt and Andreas Uhl. 2008. Image similarity measurement by Kullback-Leibler divergences between complex wavelet subband statistics for texture retrieval. , 933–936 pages.

[49] Laaposto. 2024. Image verification assistant. https://mever.iti.gr/forensics/results/396d47c2d379d8dbcd9aa3a8311e0bdbde5cf433/

[50] Liu Liu, Xinwen Fu, Xiaodong Chen, Jianpeng Wang, Zhongjie Ba, Feng Lin, Li Lu, and Kui Ren. 2023. Fits: Matching camera fingerprints subject to software noise pollution. , 1660–1674 pages.

[51] Rae Marie Manar. 2024. 9 Common Reasons for Copyright Infringement and Its Effects. https://www.artworkarchive.com/blog/how-artists-can-use-copyright-law-to-protect-their-work-and-build-their-legacy

[52] Manisha, Chang-Tsun Li, Xufeng Lin, and Karunakar A Kotegar. 2022. Beyond prnu: Learning robust device-specific fingerprint for source camera identification. *Sensors* 22, 20 (2022), 7871.

[53] Francesco Marra, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. 2017. Blind PRNU-based image clustering for source identification. *IEEE Transactions on Information Forensics and Security* 12, 9 (2017), 2197–2211.

[54] Fernando Martín-Rodríguez, Fernando Isasi-de Vicente, and Mónica Fernández-Barciela. 2023. A stress test for robustness of photo response nonuniformity (camera sensor fingerprint) identification on smartphones. *Sensors* 23, 7 (2023), 3462.

[55] MOBILedit. 2024. Answers the question if a photo was taken by a suspected camera. https://www.mobiledit.com/camera-ballistics

[56] In Jae Myung. 2003. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology* 47, 1 (2003), 90–100.

[57] NANPA. 2019. Global Infringement Report 2019. https://www.copytrack.com/blog/global-infringement-report-2019

[58] Bing Pan, Huimin Xie, and Zhaoyang Wang. 2010. Equivalence of digital image correlation criteria for pattern matching. *Applied optics* 49, 28 (2010), 5501–5509.

[59] Pixsy. 2024. About Pixsy. https://www.pixsy.com/about

[60] Feng Qian, Sifeng He, Honghao Huang, et al. 2023. Web Photo Source Identification based on Neural Enhanced Camera Fingerprint.

[61] QYResearch. 2024. Global Copyright Licensing Market Insights, Forecast to 2029. https://www.qyresearch.com/reports/1114678/copyright-licensing

[62] Caleb Riggs, Tanner Douglas, and Kanwalinderjit Gagneja. 2018. Image mapping through metadata. , 8 pages.

[63] Sudip Saha. 2022. Photo sharing market Outlook (2023 to 2033). https://www.futuremarketinsights.com/reports/photo-sharing-market

[64] Kai San Choi, Edmund Y Lam, and Kenneth KY Wong. 2006. Source camera identification using footprints from lens aberration. , 172–179 pages.

[65] Divya Singh, Jimson Mathew, Mayank Agarwal, and Mahesh Govind. 2023. DLIRIR: Deep learning based improved reverse image retrieval. *Engineering*

*Applications of Artificial Intelligence* 126 (2023), 106833.
[66] Paras Nath Singh and Tara P Gowdar. 2021. Reverse image search improved by deep learning. , 596–600 pages.
[67] TELEDYNE Vision Solutions. 2024. Scientific Camera Noise Sources. https://www.teledynevisionsolutions.com/learn/learning-center/scientific-imaging/scientific-camera-noise-sources/
[68] Matthew James Sorrell. 2009. Digital camera source identification through JPEG quantisation. , 291–313 pages.
[69] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *International Journal of Computer Vision* 130, 6 (2022), 1526–1565.
[70] Benjamin Sutton. 2018. How Artists Can Use Copyright Law to Safeguard Their Work. https://www.artsy.net/article/artsy-editorial-artists-copyright-law-safeguard-work
[71] Chunwei Tian, Menghua Zheng, Wangmeng Zuo, Bob Zhang, Yanning Zhang, and David Zhang. 2023. Multi-stage image denoising with the wavelet transform. *Pattern Recognition* 134 (2023), 109050.
[72] Chunwei Tian, Menghua Zheng, Wangmeng Zuo, Bob Zhang, Yanning Zhang, and David Zhang. 2023. Multi-stage image denoising with the wavelet transform. https://github.com/hellloxiaotian/MWDCNN.
[73] TinEye. 2024. About TinEye. https://tineye.com/about
[74] Diego Valsesia. 2016. Imaging using random projections: compression, communication, camera identification.
[75] Diego Valsesia, Giulio Coluccia, Tiziano Bianchi, and Enrico Magli. 2015. Large-scale image retrieval based on compressed camera identification. *IEEE Transactions on Multimedia* 17, 9 (2015), 1439–1449.
[76] Diego Valsesia, Giulio Coluccia, Tiziano Bianchi, Enrico Magli, et al. 2015. CAMERA IDENTIFICATION AND IMAGE RETRIEVAL VIA COMPRESSED PRNU.
[77] Angela Villaruz. 2024. Pros and Cons of Reverse Image Search. https://www.softlist.io/pros-and-cons-of-reverse-image-search/
[78] Waheeb Yaqub, Manoranjan Mohanty, and Nasir Memon. 2018. Towards camera identification from cropped query images. , 3798–3802 pages.
[79] Wen-Na Zhang, Yun-Xia Liu, Ze-Yu Zou, Yun-Li Zang, Yang Yang, and Bonnie Ngai-Fong Law. 2019. Effective source camera identification based on MSEPLL denoising applied to small image patches. , 1740–1744 pages.
[80] Feng Zhao, Qingming Huang, and Wen Gao. 2006. Image matching by normalized cross-correlation. , II–II pages.