Computational Sensor Fingerprints

Paweł Korus *Member, IEEE* and Nasir Memon, *Fellow, IEEE*

***Abstract*—Analysis of imaging sensors is one of the most reliable photo forensic techniques, but it is increasingly challenged by complex image processing in modern cameras. The underlying photo response non-uniformity (PRNU) is distilled into a static sensor fingerprint unique for each device. This makes it easy to estimate and spoof and limits its reliability in face of sophisticated attackers. We propose to exploit computational capabilities of emerging intelligent vision sensors to design next-generation *computational sensor fingerprints*. Such sensors allow for running neural network inference directly on raw pixels, which enables end-to-end optimization of the entire photo acquisition and distribution pipeline. Control over fingerprint generation allows for adaptation to various requirements and threat models. In this study we provide a detailed assessment of security properties and evaluate two approaches to prevent spoofing: fingerprint generation based on local image content and adversarial training. We found that adversarial training is currently impractical, but content fingerprints deliver good performance in the considered cross-domain (RAW-RGB) setting and could provide robust best-effort protection against photo manipulation. Moreover, computational fingerprints can alleviate other limitations of PRNU, e.g., its limited reliability for dark/texture content and expensive fingerprint storage that hinders scalability. To enable this line of work, we developed a novel open-source and high-fidelity simulation environment for modeling photo acquisition and distribution pipelines (https://github.com/pkorus/neural-imaging).**

I. Introduction

Sensor fingerprints proved to be one of the most reliable and versatile image forensic traces with applications in source attribution, manipulation detection [1–3] and several other downstream problems (e.g., clustering [4], photo carving [5], user authentication [6]). Its analysis exploits intrinsic photo-response non-uniformity (PRNU) of imaging sensors and involves estimation of a unique fingerprint for each camera. Subsequent statistical analysis verifies consistency of image residuals against the expected fingerprint, either globally (for attribution) or locally (for manipulation detection).

Despite their impressive track record, PRNU fingerprints have several limitations. Analysis requires pixel-perfect synchronization between the analyzed image and the fingerprint which makes the process brittle and may require brute-force search for synchronization. While efficient algorithms exist for simple transformations (e.g., cropping) many post-processing steps need complex heuristics (e.g., motion-stabilized video [7] or high dynamic range imaging [8]) or have no solutions at all. Increasing adoption of computational photography and learned image processing operators will be another challenge. Problems with fingerprint uniqueness have already been reported for modern smart-phones [9] and neural image signal processors (ISPs), even trained to faithfully reproduce the standard camera pipeline, can yield incompatible fingerprints [10, 11]; more complex pipelines (e.g., for low-light imaging [12]) can invalidate the approach entirely [11]. As ISPs evolve, intrinsic sensor fingerprinting will likely become obsolete.

We believe computational photography brings not only challenges but also opportunities. We show that it is possible to learn novel sensor fingerprints using neural networks (NNs). Such *computational sensor fingerprints* (CSF) could be deployed on emerging intelligent sensors with integrated NN inference capabilities [13–16]). Modern machine learning frameworks bring powerful automatic differentiation capabilities, which can be leveraged to optimize complex models that would otherwise be untractable. We can learn to embed in the raw domain and detect in the RGB domain, even with a complex channel in between (camera ISP, lossy compression, or even recapture [17, 18]). This enables media protection at the very beginning of their digital life cycle.

A next-generation sensor fingerprinting system could also address other limitations of PRNU, such as low reliability in dark/textured images, and poor security & privacy properties. Static nature the artifact makes it easy to estimate and spoof. Unsuspecting users can be deanonymized by linking photographs to public social media profiles [19], which may be undesirable, e.g., when documenting civil rights abuses. These reasons alone make a compelling case for integrating a *secure capture* (SC) mode with control over various *authentication signals* with different security & privacy trade-offs. An exhaustive discussion of secure capture requirements can be found in a recent report from Witness, an organization that helps defend human rights using photo & video technology [20].

SC is currently provided by several smart-phone apps [21–23]) and is gaining traction in emerging standardization efforts, e.g., via CAI and C2PA initiatives [24, 25]. These solutions are based on meta-data extensions with cryptographic digital signatures, and predominantly use software implementations, which may be susceptible to OS-level spoofing e.g., via device virtualization. One of the main limitations is their need for compatible tools at every editing and recompression step and susceptibility to meta-data removal (e.g., via stripping or recapture). Integration with the camera is still an open problem with ongoing R&D efforts[1]. Deploying protection at the sensor

This material is based upon work supported by the National Science Foundation under Grant No. 1909488.

P. Korus is with Tandon School of Engineering, New York University, USA, and also with the Department of Telecommunications, AGH University of Science and Technology, Poland (e-mail: pkorus@agh.edu.pl).

N. Memon is with Tandon School of Engineering, New York University, USA (e-mail: memon@nyu.edu).

[1] Truepic has recently announced a prototype implementation based on the trusted execution environment (TEE) in Qualcomm's Snapdragon chips. The details of their system are not public, but the system aims to move signature computation to a trusted environment and to integrate with the emerging meta-data standards [24].

level could be attractive since it is the very first step in photo acquisition. Subsequent tampering with user-level firmware allows for seamless synthesis of counterfeit photos with correct forensic traces and meta-data [26]. A next-generation sensor fingerprinting system could complement the emerging media security ecosystem and authentication infrastructure.

In this work, we consider a generalization of sensor fingerprints that naturally lends itself to implementation using neural networks and could be deployed on the emerging intelligent vision sensors and optimized end-to-end for complex distribution channels. Our system generates fingerprints dynamically based on a secret key and an additional authentication context (e.g., timestamp, location, or local image content) which gives it flexibility to adapt to various applications and threat models. Introducing content dependency increases resistance to spoofing and raises the bar for crafting undetectable photo manipulations. In addition to improved security, our learned sensor fingerprinting system delivers better detection performance at a lower image distortion level compared with standard PRNU. This shows that degradation of image quality is limited and should not deter vendors from deployment.

We believe that our work could be the first step towards a next-generation sensor fingerprinting system that takes full advantage of the emerging in-camera computational capabilities. We build upon a solid foundation of PRNU fingerprints and aim to improve them with end-to-end optimization. We summarize the main contributions of our work as follows:

1) We propose *computational sensor fingerprints*, a novel concept that takes advantage of emerging vision sensors with integrated computational capabilities.
2) We perform a detailed comparison with intrinsic sensor fingerprints (and spread spectrum watermarks) in a simulated environment and show novel results on synthetic PRNU simulation.
3) We provide a systematic study of security properties and evaluate two strategies to prevent spoofing: training-time attack modeling and content fingerprints; we demonstrate that content fingerprints can deliver good detection performance in cross domain matching (RAW-RGB) and significantly raise the bar for a successful attack.
4) We extended our neural imaging toolbox with sensor fingerprint modeling and optimization features.

To enable this line of work, we built a high-fidelity simulation environment for modeling and end-to-end optimization of complex photo acquisition and distribution pipelines. Our *neural imaging toolbox* provides building blocks for various applications including optimization of camera ISP [27], image codecs [28], forensics [29], or steganography. The toolbox is open source and can be obtained at https://github.com/pkorus/neural-imaging.

II. Related Work

We review related work in: (a-c) photo manipulation detection (passive & active); (d) computational imaging & neural sensors; (e) end-to-end optimization of photo acquisition & distribution; and (f) signal authentication in other domains.

a) Passive Detection with Image Forensics: Image forensics analyses subtle traces left by various steps during photo acquisition, distribution, post-processing or manipulation [30, 31]. Validation of *physical integrity* (consistency of lighting, shadows, perspective, etc.) is manual or semi-automatic as it relies on high-level understanding of image content and laws of physics. Verification of *digital integrity* (consistency of low-level signal features) is automatic and uses formal statistical modeling or machine learning. Some of the most successful tools include analysis of: 1) sensor fingerprints [1–3]; 2) lossy compression (e.g., local inconsistency in JPEG compression [32–34]); and 3) image residuals or their statistical features [29, 35, 36]. Initially, forensic detectors relied on formal mathematical modeling (e.g., hypothesis testing) and machine learning on hand-crafted features (e.g., co-occurrence). Recent work focuses on deep learning (DL) to obtain low-level features automatically [29, 36–39].

Despite two decades of active research, adoption of image forensics remains limited; to the best of our knowledge, only PRNU fingerprints satisfy the Daubert standard for evidence admissibility in US courts [40]. One of the main limitations is the fragility of forensic traces to common post-processing and lossy compression in distribution channels [38]. Many methods are reliable mainly for native camera output and the situation may soon change for worse due to increasing adoption of complex computational methods directly inside of the cameras, e.g., night-mode or high dynamic range imaging [8]. Secondly, forensic traces are susceptible to spoofing (known as *counter forensics*) [26, 41, 42]. While some of these attacks leave traces of their own (and can be detected), reliability of forensic analysis in the real-world remains unclear.

b) Active Protection and Digital Watermarking: Active protection is a compelling alternative to passive techniques, but requires a SC workflow where an explicit protection step attaches side information to captured assets to describe their authenticity and provenance. Digital watermarking can serve as a communication layer and embed this information directly in image pixels [43] which addresses the loss of meta-data. NNs can learn effective embedding strategies [44], including optimization for analytically intractable channels, e.g., display-to-camera or printer-to-camera communication [17, 18].

Authentication systems based on digital watermarks have been extensively studied [30, 45] and can provide advanced integrity verification features, including precise manipulation localization or even restoration of original content [46–48]. The embedded side-information can be sensitive to any (fragile watermarking) or selected manipulations (semi-fragile watermarking) and the watermarks can be designed to survive various post-processing. This approach has been an active research field in the first decade of the century, but despite impressive results the interest eventually subsided. Lack of viable business models created little incentives for camera vendors to adopt the technology, especially since it involves additional image distortion and is non-trivial to deploy securely. One of the main barriers involved management and storage of keys, which turned out to be the weakest point of commercial signature-based systems from Kodak and Nikon [30]. Secure enclaves were not available and encryption keys could simply be extracted from memory chips.

We currently see increasing interest in building a robust SC infrastructure. Cameras are displaced by smart-phones with constant connectivity, computation accelerators and trusted storage (used e.g., for biometric templates). Emerging provenance standards will likely establish a key management infrastructure that could be reused by various protection methods.

c) Signatures, Hashing & Block-chains: A secure camera based on digital signatures has been discussed two decades ago [49]. In the most recent incarnations, the signatures can be stored in a block-chain, as explored in academic research [50], commercial products [21, 22], and standardization efforts [51]. Digital signatures lack support for post-processing (e.g., compression) and can be easily stripped from media containers. One way of addressing these problems involves provenance tracking and requires regeneration of the signatures at every step. This assumes a complete ecosystem with compatible devices, photo editors and distribution platforms (route taken by C2PA). Alternatively, one may use robust hashing, which remains invariant to post-processing and may capture spatial information to enable manipulation localization [52]. To address meta-data removal, one may store its copy online and use a watermark to embed a lookup ID [24, 50, 51].

d) Computational Photography and NN Accelerators: Current research on camera ISPs heavily relies on DL. While NNs for individual operations (e.g., demosaicing) have been known for a long time [53], recent work improves quality [54], combines several steps (e.g., demosaicing and denoising [55]), or even replaces the entire pipeline [12, 56, 57]. Modern NNs are highly effective in extreme imaging conditions, e.g., low-light [12], single shot HDR [58] or super-resolution [59]. Practical implementation is facilitated by increasing availability of NN accelerators [60] which are currently emerging in imaging sensors [14–16]. *Neural sensors* enable simultaneous acquisition and inference directly on raw pixel measurements. This enables novel capabilities, like per-pixel shutter control [61] or the proposed *computational sensor fingerprints*. In addition to research prototypes, first commercial sensors of this kind are making their way to the market (Sony IMX500 [13]).

e) End-to-end Acquisition and Distribution Optimization: Our recent work explored joint optimization of various components in the image acquisition and distribution pipeline to address some of the key issues in multimedia security. First, we demonstrated that it is possible to learn synthetic forensic traces that facilitate manipulation identification and survive highly lossy channels (25% down-sampling and strong JPEG compression) [11, 27]. We achieved this by fine-tuning a neural ISP model together with a forensic analysis network [29]. Secondly, we showed that lossy compression can also be trained to retain forensic traces with better fidelity [28]. We learned a neural image codec and demonstrated that it can be fine-tuned to improve manipulation identification accuracy even at very low bit-rates, well below the practicality of JPEG (QF ≈20). The current study extends this line of work and aims to optimize sensor-level traces.

f) Signal Authentication in Other Domains: DL can be used for signal authentication in the Internet-of-Things [62]. A long short-term memory (LSTM) network can extract stochastic signal features used for constructing a spread spectrum watermark. To improve security, the watermark changes dynamically to prevent eavesdropping and estimation attacks. A recent line of work started exploring artificial fingerprints that could be introduced to training data and later detected in images sampled from modern generative ML models [63].

TABLE I
NOTATION AND SYMBOLS

$\circledast$		normalized correlation coefficient
$\oplus$		channel-wise tensor concatenation
$\mathbf{x}$	$[0,1]^{\frac{h}{2}\times\frac{w}{2}\times 4}$	RAW image represented as *rggb* channels
$\mathbf{y}$	$[0,1]^{h\times w\times 3}$	color images with *RGB* channels
$\mathbf{k}_*$	$\sim\mathcal{N}(\mathbf{0},\mathbf{I})\in\mathbb{R}^{\frac{h}{2}\times\frac{w}{2}\times 4}$	sensor fingerprint (RAW domain)
$\mathbf{k}'_*$	$\mathcal{M}(\mathbf{k}_*)\in\mathbb{R}^{h\times w\times 3}$	sensor fingerprint (RGB domain)
$\mathbf{x}_0$	$\mathcal{E}(\mathbf{x},\mathbf{k}_0)$	RAW image with embedded fingerprint $\mathbf{k}_0$
$\mathbf{y}_0$	$\mathcal{I}(\mathbf{x}_0)$	color images with embedded fingerprint $\mathbf{k}_0$
$\mathbf{r}_0$	$\mathcal{R}(\mathbf{y}_0)\in[0,1]^{h\times w\times 3}$	high-frequency residual
$\tilde{\mathbf{r}}_0$	$\mathbf{y}_0-\mathbf{y}$	hypothetical true residual
$\mathcal{M}$	$\mathbb{R}^{\frac{h}{2}\times\frac{w}{2}\times 4}\rightarrow\mathbb{R}^{h\times w\times 3}$	demosaicing filter
$\mathcal{E}$	$(\mathbf{x},\mathbf{k}_0)\rightarrow\mathbf{x}_0$	fingerprint encoder / embedder
$\mathcal{D}$	$(\mathbf{y},\mathbf{k}_0)\rightarrow d\in\mathbb{R}$	fingerprint detector
$\mathcal{I}$	$\mathbf{x}\rightarrow\mathbf{y}$	camera ISP
$\mathcal{C}$	$\mathbf{y}\rightarrow\mathbf{y}$	JPEG codec ($\mathcal{C}_{\text{QF}}$ whenever QF is relevant)
$\mathcal{R}$	$\mathbf{y}_0\rightarrow\mathbf{r}_0$	a residual filter
$\mathbf{k}'_{0/*}$	$\mathbf{k}'_{0/r}$, $\mathbf{k}'_{0/d}$, $\mathbf{k}'_{0/e}$	estimated fingerprints (Section IV-B)
$\mathbf{y}_{0/*}$	$\mathbf{y}_{0/+}$, $\mathbf{y}_{0/\nabla}$, $\mathbf{y}_{0/\approx}$	image with spoofed fingerprint $\mathbf{k}_0$ (Section IV-B)

III. NOTATION & PROBLEM STATEMENT

Throughout the paper we use lowercase symbols for scalars, lowercase bold symbols for vectors/tensors, and calligraphic symbols for functions. To simplify notation, we assume the index of a chosen tensor element is implicit, e.g., x represents a single element of $\mathbf{x}$, and $\mathbf{x}_{[\pm s]}$ represents a neighborhood of size $(2s+1)\times(2s+1)$ around that element. We use the following letter convention for classes of symbols: $\mathbf{x}$ denotes RAW images, $\mathbf{y}$ color RGB images, and $\mathbf{k}$ the fingerprints. We represent RAW images as *rggb* stacks with different Bayer-filtered components stacked along the channel dimension. Subscripts correspond to additional characteristics of the objects, e.g., $\mathbf{y}_0$ corresponds to an image with embedded fingerprint $\mathbf{k}_0$. We summarize the notation in Tab. I.

We consider classic and computational sensor fingerprints in a common framework (Fig. 1). A sensor fingerprinting system includes an *encoder* (implicit for PRNU and explicit for CSF) and a *detector* which assesses the likelihood of two outcomes (correct fingerprint present or absent). A deployed detector has access to a secret key κ_0 and a signed authentication context, e.g., via meta-data or a second small-payload watermark.

A fingerprint $\mathbf{k}_0$ is embedded in the RAW image $\mathbf{x}$ and detected in a color image $\mathbf{y}$. The fingerprint is static and latent for PRNU, and dynamic and observed for CSF - it is generated based on a secret key κ_0 and an authentication context (e.g., timestamp, location, image content) which ensures it changes dynamically. Fingerprints generated based on image content are denoted as $\mathbf{k}_{\{x/y\}}$ for RAW and RGB inputs, respectively. The bracketed subscript notation denotes multiple options (i.e., $\mathbf{k}_{\{x/y\}}\rightarrow\mathbf{k}_x$ and/or $\mathbf{k}_y$). It should not be confused with the double subscript notation (e.g., $\mathbf{k}'_{0/d}$ denotes an adversarial estimate of the fingerprint obtained by targeting the detector).

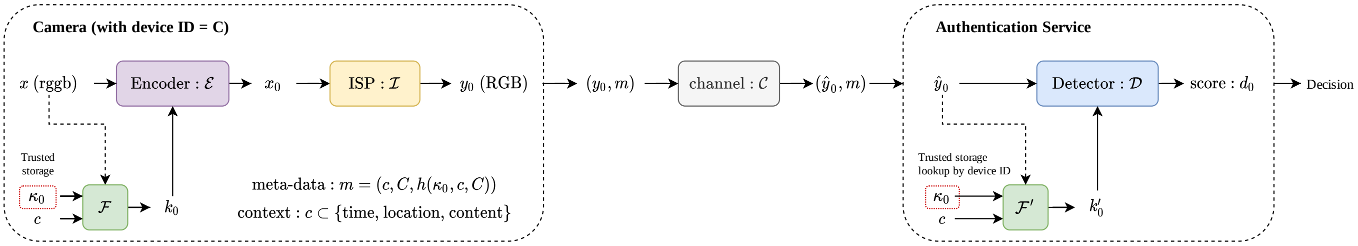


Fig. 1. Schematic illustration of a sensor fingerprinting system: a fingerprint $\mathbf{k}_0$ (generated based on a secret key and authentication context) is embedded in the RAW image $\mathbf{x}$ and passed through the camera ISP to produce a color image $\mathbf{y}_0$; after transmission via a lossy channel, the fingerprint can be detected directly from a (distorted) RGB image $\hat{\mathbf{y}}_0$; the detection is provided by an *authentication service* that looks up the key and regenerates the expected fingerprint.

In this case, wildcards ($*$) are used to denote multiple options (e.g., $\mathbf{k}'_{0/*}$ denotes any estimated fingerprint).

A. Photo Response Non-Uniformity and Embedding Models

Due to manufacturing imperfections, pixels in imaging sensors exhibit minor photo response variations (*photo-response non-uniformity* or *fixed pattern noise*) that can be used as a fingerprint of each device. Its forensic analysis assumes a simple pixel-wise sensor noise model[2]:

$$x_0 = (1 + \alpha k_0)x + \eta, \tag{1}$$

where $\mathbf{x}_0$ is a captured RAW image; $\mathbf{x}$ is its idealized noise-free version; $\mathbf{k}_0$ is the sensor fingerprint (PRNU); and η aggregates all remaining sources of noise. Thanks to the additional modulation strength α this embedding model is equivalent to multiplicative spread spectrum watermarking [66] and allows for controlling the distortion-detection trade-off in our simulations. In practice, α is implicitly determined by the camera hardware. As a result, the sensor fingerprint can be seen as a free static watermark intrinsically embedded by the sensor.

Even though the PRNU describes sensor behavior at the RAW level, it is common to assume the same model holds for RGB images[3]: $y_0 = (1 + \alpha k_0)y + \eta'$. The fingerprint $\mathbf{k}_0$ is latent and needs to be estimated; the optimal MLE estimator uses residual images $\mathbf{r}_0 : r_0 = y_0 - \hat{y} \approx yk_0$ where $\hat{y}$ is an estimate of the noise-free image obtained with a denoising filter. The number of residuals needed for reliable estimation depends on image content, but typically recommended numbers range between 30 and 100 (see Supplement B for our synthetic evaluation). Bright and flat images are preferred due to the multiplicative nature of the noise and difficulties in separation from high-frequency content.

Fingerprint detection involves correlation of the image residual $\mathbf{r}$ with the (estimate of the) PRNU fingerprint $\mathbf{k}'_0$, yielding $d = \mathbf{r} \circledast \mathbf{k}'_0 \in [-1, 1]$. The problem is commonly formulated as hypothesis testing with the following conditional distributions:

$$d \sim \begin{cases} \mathcal{N}(\mu, \sigma_0) & H_0 : \text{fingerprint } \mathbf{k}_0 \text{ is present} \\ \mathcal{N}(0, \sigma_1) & H_1 : \text{fingerprint } \mathbf{k}_0 \text{ is absent} \end{cases} \tag{2}$$

where $\mu > 0$ is the expected correlation when the correct fingerprint is present. Adoption of a Gaussian noise model is dictated by convenience, and despite mismatched support with the correlation coefficient, it works well in practice.

This binary-decision formulation is used for attribution. For manipulation detection, sliding window analysis is used. Multiplicative nature of the fingerprint complicates the problem since the expected correlation depends on image content, i.e., $\mu = \mu(\mathbf{y}_{[\pm s]})$, which is addressed by linear regression on visual content features [1]. Local tampering probability can be computed based on a Bayesian formulation spanning a local neighborhood [2] and multiple scales [3].

B. Applications and Limitations

The sensor noise model (1) implies several key limitations:

1) *weak security* - PRNU is a static signal easy to estimate and spoof, e.g., to conceal traces of content manipulation; simple addition is effective ($\mathbf{y} + \beta\mathbf{r}_0$), but more advanced attacks exist too [69]. The triangle test may reveal spoofing in certain conditions [70, 71].
2) *unexpected privacy leak* - PRNU is always present and static, which allows for deanonymization of photographers by linking a sensitive photo to other images with established authorship (e.g., on social media) [19].
3) *unreliability for dark & textured content* - multiplicative nature makes the PRNU weak in dark regions; in textured areas it is difficult to separate from high-frequency content; this complicates the analysis and even helps the attackers (the prediction model reveals the expected correlation and facilitates precise spoofing).
4) *low analysis resolution* - analysis requires a large sample size for sufficient statistics, especially in unreliable regions (dark, textured) or distorted images (compressed, resized); the standard window size (128 px) is larger than for other forensic methods and misses small manipulations (compressed images require even larger windows).
5) *pixel-perfect synchronization* - analysis requires pixel synchronization between the test image and the sensor; many common operations violate this assumption and the problem is addressed by brute-force search in a parametrized space of the operator; while for some operations (cropping, scaling) the search can be performed effectively, most operations are intractable.

[2] Preliminary studies with synthetic signals report that this model may not be fully accurate [64]. Recent work in image denoising [65] adopts a more comprehensive image formation model, but to the best of our knowledge such models have not been considered in forensics yet.

[3] A recent study addressed this by probabilistic raw pixel estimation [67]. Similar ISP inversion problems also occur in recent denoising literature [68].

6) *poor scalability* - scalability in time is hindered by expensive search for synchronization; scalability in space is limited by fingerprint size (dozens of MB per device)[4].

Designing a novel sensor fingerprinting system can address many of these issues and allows for adaptation to various application requirements and threat models.

C. Generalization of the Embedding and Detection Models

The sensor noise model (1) is pixel-wise and assumes the fingerprint components follow an IID normal distribution, i.e., $\mathbf{k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This assumption holds for raw pixels, but it is typically kept in RGB domain as well, despite correlations introduced by the camera ISP (e.g., via demosaicing). Effectively, the standard model assumes $y_0 \approx y + \alpha y k_0'$ where the RGB-domain fingerprint $\mathbf{k}_0'$ is different (although highly correlated) from the ground truth $\mathbf{k}_0$. Since we are working with synthetic signals, we can measure and visualize this behavior (see Supplement B).

The pixel-wise spread spectrum embedding (1) intrinsically adopted by PRNU naturally generalizes to a more expressive model, where each pixel is obtained as a function of a small neighborhood of both the content and the fingerprint $y_0 = y - \mathcal{E}'(\mathbf{y}_{[\pm s]}, \mathbf{k}'_{0[\pm s]})$ where $\mathcal{E}'$ is a generic content-dependent fingerprint embedding function defined on neighborhood of size $(2s+1) \times (2s+1)$. While PRNU uses a simple pixel-wise multiplicative form of content dependence, data driven techniques could learn a better mapping. Since embedding happens in the raw domain, we use the following model:

$$x_0 = x - \mathcal{E}(\mathbf{x}_{[\pm s]}, \mathbf{k}_{0[\pm s]}) \ . \tag{3}$$

PRNU fingerprint detection involves normalized correlation between an image residual $\mathbf{r}$ and the fingerprint estimate: $d = \mathbf{r} \circledast \hat{\mathbf{k}}_0'$. The problem can be generalized in several ways. We obtained the most reliable results with a *correlation-based* model which separately processes the fingerprint and the image and uses normalized correlation as the final score:

$$d = \mathcal{D}_e(\mathbf{y}, \mathbf{k}_0') = \mathcal{D}_y(\mathbf{y}) \circledast \mathcal{D}_k(\mathbf{k}_0') \in [-1, 1] \tag{4}$$

We also experimented with an *end-to-end* model which processes everything jointly and directly outputs a real-valued detection score:

$$d = \mathcal{D}_d(\mathbf{y}, \mathbf{k}_0') \in \mathbb{R} \tag{5}$$

Since the fingerprint is generated dynamically, it eliminates the need for estimation. This saves storage and improves detection performance (Fig. B.3 and Fig. 3).

D. Model Architectures and Training

Given embedding and detection models (3)-(5), the goal is to jointly learn a CSF system $(\mathcal{E}, \mathcal{D})$ optimized for a given acquisition and distribution pipeline. All considered models can be naturally implemented as CNNs. For simplicity, we assume feed-forward fully convolutional NNs, which can scale across image resolutions. The encoder is a residual model:

$$\mathcal{E}(\mathbf{x}, \mathbf{k}_0) \rightarrow \mathbf{x}_0 := \begin{cases} \hat{\mathbf{z}}_0 & = \mathbf{x} \oplus \mathbf{k}_0 \\ \mathbf{z}_i & = \mathrm{c2d}_{f \times s \times s}(\hat{\mathbf{z}}_{i-1}) \\ \hat{\mathbf{z}}_i & = \sigma(\mathbf{z}_i) \\ \mathbf{x}_0 & = \mathbf{x} - \mathrm{c2d}_{4 \times 1 \times 1}(\hat{\mathbf{z}}_n) \end{cases} \tag{6}$$

where $\mathrm{c2d}_{f \times s \times s}$ is a 2d convolution layer with f filters of size $s \times s$ and σ is a non-linear activation function. The input is a channel-wise concatenation ($\oplus$) of the image and the fingerprint. Note that the last layer uses 1×1 convolutions for pixel-wise projection back to the RAW domain. We follow a convention with standardized representation based on *rggb* channels (see Tab. I).

The detector uses a similar design. The correlation-based model separately projects the image and the fingerprint to single-channel representations $\mathbf{z}_y$ and $\mathbf{z}_k$, respectively. It then computes normalized correlation to yield the detection score. Each branch uses the same model architecture - $\mathcal{D}_{k/y}(\hat{\mathbf{z}}_0)$ - but uses different input $\hat{\mathbf{z}}_0$:

$$\mathcal{D}_{\{k/y\}}(\hat{\mathbf{z}}_0) \rightarrow \mathbf{z}_{\{k/y\}} := \begin{cases} \mathbf{z}_i & = \mathrm{c2d}_{f \times s \times s}(\hat{\mathbf{z}}_{i-1}) \\ \hat{\mathbf{z}}_i & = \sigma(\mathbf{z}_i) \\ \mathbf{z}_{\{k/y\}} & = \mathrm{c2d}_{1 \times 1 \times 1}(\hat{\mathbf{z}}_n) \end{cases} \tag{7}$$

One branch processes the fingerprint ($\hat{\mathbf{z}}_0 = \mathbf{k}'$), while the other processes the image ($\hat{\mathbf{z}}_0 = \mathbf{y} \oplus \mathcal{R}(\mathbf{y})$; we augment the image processing branch with a trainable residual layer $\mathcal{R}(\mathbf{y})$ that suppresses image content [29]).

The end-to-end model processes the image and the fingerprint jointly and uses fully connected layers to yield the final detection score:

$$\mathcal{D}_d(\mathbf{y}, \mathbf{k}') \rightarrow d := \begin{cases} \hat{\mathbf{z}}_0 & = \mathbf{y} \oplus \mathcal{R}(\mathbf{y}) \oplus \mathbf{k}' \\ \mathbf{z}_i & = \mathrm{conv2d}_{f \times s \times s}(\hat{\mathbf{z}}_{i-1}) \\ \hat{\mathbf{z}}_i & = \sigma(\mathbf{z}_i) \\ \hat{\mathbf{p}}_0 & = \sum_{w,h} \hat{\mathbf{z}}_n \\ \mathbf{p}_i & = \mathrm{fc}_{o_i}(\hat{\mathbf{p}}_{i-1}) \\ \hat{\mathbf{p}}_i & = \sigma(\mathbf{p}_i) \\ d & = \mathrm{fc}_1(\hat{\mathbf{p}}_m) \end{cases} \tag{8}$$

where fc_o denotes a fully connected layer with o outputs and global average pooling (GAP) is used to standardize an internal representation $\hat{\mathbf{p}}_0$ returned by the last convolutional layer.

For simplicity, convolutional layers share hyper-parameters (stride of 1 and kernel size s=3) and successive fully connected layers halve the number of outputs. Model size can be adjusted using the number of layers (n) and filters (f). Intermediate layers use *LeakyReLU* activations and final layers use none. We illustrate all considered architectures in Fig. C.1.

The detector operates in the RGB domain which requires fingerprint mapping from raw values. In ideal conditions, this should reflect the mapping performed by the camera ISP. In practice, we use simple bilinear interpolation which in preliminary experiments yielded performance equivalent to preprocessing with the true demosaicing model.

[4] PRNU is a full-resolution real-valued signal. Scalability can be mitigated by quantization. While simple binarization leads to poor accuracy, more elaborate dead-zone quantization can be considered in large-scale applications [72].

TABLE II
SPOOFING THREAT MODELS: REPRESENTATIVE ATTACKS AND ATTACKER'S KNOWLEDGE: ■ INDICATES BLACK-BOX MODEL ACCESS; ▽ INDICATES GRADIENT-BASED OPTIMIZATION.

Attack vector	**Attacker's knowledge**					
	$\mathbf{x}$	$\mathbf{y}_0$	$\hat{\mathbf{k}}_0$	$\mathcal{D}()$	$\mathcal{D}$	$\mathcal{E}$
Phase I : fingerprint estimation						
image residual		✓	n/a			
adversarial w/ proxy detector		✓	n/a		▽	
adversarial w/ proxy encoder	✓	✓	n/a			▽
Phase II : adversarial image generation						
additive residual transfer			✓			
adversarial opt. w/ authorized detector				▽		
adversarial opt. w/ proxy detector			✓		▽	
approximate fingerprint embedding			✓			■

Our *vanilla training protocol* aims to minimize a combination of image distortion and fingerprint detection losses:

$$\mathcal{L}_\text{v} = \alpha \left\| \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{x}_0) \right\|_2 + \mathcal{L}_\text{d}(d_0, d_1) \tag{9}$$

where $\mathcal{P}$ maps raw images into a domain suitable for perceptual comparison. For simplicity, we map into the RGB domain using the camera ISP ($\mathcal{P} = \mathcal{I}$) but one could also use modern perceptual metrics based on deep visual representations [73]. We control the embedding strength via a regularization term α. The detection terms $d_{\{0/1\}}$ correspond to positive/negative detection scores, i.e.,

$$d = \begin{cases} d_0 = \mathcal{D}(\mathbf{y}_0, \mathbf{k}'_0) \\ d_1 = \mathcal{D}(\mathbf{y}_0, \mathbf{k}'_1) \end{cases} \tag{10}$$

where $\mathbf{k}'_1$ denotes a different (orthogonal) fingerprint.

The fingerprint detection loss depends on the adopted model architecture. Let $\mathcal{L}_\text{l}$ denote a logistic loss penalty $\mathcal{L}_\text{l}(d) = \log(1 + e^d)$. For the correlation-based model, we used:

$$\mathcal{L}_\text{d}(d_0, d_1) = \mathcal{L}_\text{l}(d_1 - d_0) + \lambda_n \left\| d_1 \right\|_2 \tag{11}$$

which drives negative detections (d_1) towards 0 and maximizes the difference between the detection scores. For the end-to-end model, we simply used:

$$\mathcal{L}_\text{d}(d_0, d_1) = \mathcal{L}_\text{l}(d_1) + \mathcal{L}_\text{l}(-d_0) \tag{12}$$

which drives the scores to negative/positive values, respectively (Fig. 2 illustrates how this impacts detection statistics). Finally, the training algorithm is simply SGD that seeks:

$$\underset{\mathcal{E},\mathcal{D}}{\text{argmin}} \underset{\mathbf{x}}{\mathbb{E}} \underset{\mathbf{k_0}}{\mathbb{E}} \underset{\mathbf{k_1}}{\mathbb{E}} \underset{\text{QF}}{\mathbb{E}} \mathcal{L}_\text{v} \tag{13}$$

where the expectation is computed over JPEG QF, images $\mathbf{x}$ and fingerprints $\mathbf{k}_{\{0/1\}} \sim \mathcal{N}(0, \mathbf{I})$ sampled in each step.

IV. Threat Models and Countermeasures

In this section we define threat models and attack vectors and discuss possible countermeasures.

A. Threat Models

Sensor fingerprinting systems are susceptible to two types of attacks: spoofing and removal. We focus on spoofing since it has a greater cost of allowing doctored content to pass as authentic or falsely attributing a photo to a wrong individual.

In the following discussion, we refer to two possible deployments of CSF components:

- an *authorized* model uses the true fingerprint generated based on a secret key and context; the model controls the access interface and may use additional defenses to thwart/delay adversarial queries.
- a *proxy* model denotes a deployed leaked model where the adversary may feed any input as the fingerprint and disable any defenses or restrictions on query access.

For brevity, we denote the authorized and proxy detectors as AD and PD, respectively.

Spoofing can be carried out in various ways, depending on the attacker's sophistication and knowledge/access level. We distinguish 3 main attack techniques:

- *additive residual transfer* (ART) is the simplest attack and can be carried out without access to any system components; it works well against standard PRNU fingerprints;
- *approximate fingerprint embedding* (AFE) uses a known (e.g., leaked) embedding model ($\mathcal{E}$) and an *adversarial fingerprint estimate* $\mathbf{k}_{0/*}$; it involves a single forward pass and hence black-box access (■) is sufficient.
- *adversarial image optimization* (AIO) applies incremental updates to the image driven by the gradient of the detector (▽) or its approximation from black-box optimization (e.g., NES); it requires either a PD ($\mathcal{D}$) or query access to the AD ($\mathcal{D}()$).

In addition to adversarial image generation, many attacks also involve a prior *fingerprint estimation* phase, which can be implemented using image residuals or gradient-based optimization targeting proxy models. We summarize the corresponding threat models and the necessary knowledge on the attacker's side in Tab. II and formally define all attacks in Sections IV-B and IV-C.

We generally assume the attacker has access to a photograph $\mathbf{y}_0$ (with an embedded correct fingerprint $\mathbf{k}_0$). If the fingerprint is not content-dependent, $\mathbf{y}_0$ may not be too difficult to obtain (e.g., by taking a photo with a spoofed context) or may even be readily available (when dealing with local content manipulation). When using fingerprints derived from image content, the attacker needs to look for hash collisions within the same authentication context. While technically still possible [74], it significantly raises the bar for a successful attack.

When dealing with optimization-based attacks, we assume *white-box* model access which may be overly permissive but paints the picture of a pessimistic security assessment. In such deployed proxy models, the attacker can feed arbitrary inputs as the candidate fingerprints which significantly expands the attack surface. We show representative attack combinations in Fig. C.3. We acknowledge that our classification only scratches the surface, since the attacks have many different variations that could include not only basic attack settings and distortion norms/budgets, but also partial knowledge of the targeted system, e.g., gray-box access to specific models or prior statistical knowledge of various signals that could regularize the search space. Out of necessity, we focus our attention on several representative attacks.

B. Fingerprint Estimation

In this phase, the goal of the attacker is to obtain a sufficiently accurate estimate $\mathbf{k}_{0/*}$ of the true fingerprint $\mathbf{k}_0$. We consider two main sources of information leakage: image residuals and an implicit loss landscape of the embedding/detection models. The former involves computation of a *high-pass residual* and treating it (after normalization) as a fingerprint estimate. This strategy is effective against standard PRNU where the problem is exacerbated by the static nature of the fingerprint and prone to continuous refinement as more images become available.

The residual can be computed by convolution with filter $\mathbf{f}$ and a fingerprint estimate is then obtained by simple normalization:

$$\mathbf{k}'_{0/\mathrm{r}} = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|_2}, \text{ where } \mathbf{r}_0 = \mathbf{f} \circledast \mathbf{y}_0 \tag{14}$$

In most situations we use a popular 3×3 high-pass residual filter ($\mathbf{f}_1$), and include additional filters ($\mathbf{f}_2$, $\mathbf{f}_3$ and a DWT filter commonly used in PRNU analysis [1]) in generalization experiments. The filters are defined as follows:

$$\mathbf{f}_1 = \frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}, \quad \mathbf{f}_2 = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \tag{15}$$

$$\mathbf{f}_3 = \frac{1}{4}\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 2 & 1 & -1 \\ -1 & 2 & 4 & 2 & -1 \\ -1 & 1 & 2 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

White-box access to CSF models enables gradient-based fingerprint optimization attacks. The attacker can start from a randomly sampled fingerprint and successively update his estimate. When using a proxy detector, the goal is to maximize detection[5] of the known fingerprinted image $\mathbf{y}_0$:

$$\mathbf{k}'_{0/\mathrm{d}} = \operatorname*{argmax}_{\mathbf{k}} \mathcal{D}(\mathbf{y}_0, \mathbf{k}) \tag{16}$$

When using a proxy encoder $\mathcal{E}$, the goal is to reproduce the valid fingerprinted image $\mathbf{y}_0$:

$$\mathbf{k}_{0/\mathrm{e}} = \operatorname*{argmin}_{\mathbf{k}} \|\mathcal{I}(\mathcal{E}(\mathbf{x}, \mathbf{k})) - \mathbf{y}_0\|_2 \tag{17}$$

The attack on the proxy encoder assumes availability of the clean raw signal $\mathbf{x}$ which is not naturally available without compromising camera hardware. With careful hardware design, this attack vector should remain impractical, but it is currently not clear what would be the impact of approximate attacks, e.g., where knowledge of the expected image statistics is leveraged to estimate a clean image.

Both adversarial attacks can be implemented as iterative algorithms based on gradient descent/ascent:

$$\mathbf{k}'_{0/\mathrm{d}} := \begin{cases} \mathbf{k}'_{[i]} + \gamma \nabla_{\mathbf{k}'} \mathcal{D}(\mathbf{y}_0, \mathbf{k}'_{[i]}) \\ \mathbf{k}'_{[0]} \sim \mathcal{N}(0, \mathbf{I}) \end{cases} \tag{18}$$

$$\mathbf{k}_{0/\mathrm{e}} := \begin{cases} \mathbf{k}_{[i]} - \gamma \nabla_{\mathbf{k}} \left\|\mathcal{I}(\mathcal{E}(\mathbf{x}, \mathbf{k}_{[i]})) - \mathbf{y}_0\right\|_2 \\ \mathbf{k}_{[0]} \sim \mathcal{N}(0, \mathbf{I}) \end{cases} \tag{19}$$

We normalize the gradient to unit L_2 norm and choose the step size γ and the number of steps differently in various contexts. During in-depth exploratory analysis, we also used a more advanced optimizer (Adam).

Finally, we note that depending on the target, the attacks yield either a RAW or RGB-domain fingerprint that may require remapping depending on the attack strategy. This can be implemented by demosaicing/sub-sampling, respectively.

C. Adversarial Image Generation

We distinguish 3 types of attacks: *additive residual transfer* (ART), *adversarial image optimization* (AIO) and *approximate fingerprint embedding* (AFE).

a) ART: Residual transfers represent naive attackers with access only to the fingerprinted image $\mathbf{y}_0$. The attacker estimates the residual $\mathbf{r}_0$ and adds it to the target image $\mathbf{y}$:

$$\mathbf{y}_{0/+} = \mathbf{y} + \beta \mathbf{r}_0 \left\|\mathbf{r}_0\right\|_2^{-1} = \mathbf{y} + \beta \mathbf{k}_{0/\mathrm{r}} \tag{20}$$

The attack strength β controls the trade-off between image distortion and spoofing success rate. To unify selection of β, the residual is normalized to unit variance. In our experiments, we use the residual filter $\mathbf{f}_1$ and sweep $\beta \in [0.001, 0.04]$.

b) AIO: Adversarial optimization uses gradients of the proxy detector to reach the desired target response. Hence, the attack is universal and can be used both for spoofing and removal under the same generic formulation:

$$\mathbf{y}_{0/\nabla} = \operatorname*{argmin}_{\mathbf{y}} \mathcal{L}_{\mathrm{s}}\left(d_{0/*}\right) = \operatorname*{argmin}_{\mathbf{y}} \mathcal{L}_{\mathrm{s}}\left(\mathcal{D}(\mathbf{y}, \mathbf{k}'_{0/*})\right) \tag{21}$$

The attack requires a fingerprint estimate $\mathbf{k}'_{0/*}$ (Sec. IV-B) and an objective function $\mathcal{L}_{\mathrm{s}}$. We found that a variation of the logistic loss works well in practice:

$$\mathcal{L}_{\mathrm{s}}(d) = \log\left(1 + e^{\pm(d - d_\tau)}\right) \tag{22}$$

It corresponds to a gentle push over/below the decision boundary d_τ (for spoofing/removal, respectively). We use a *multi-step projected gradient descent* (PGD) attack with the L_∞ distortion model:

$$\mathbf{y}_{0/\nabla} := \begin{cases} \mathbf{y}_{[i]} - \gamma \operatorname{sgn}\left(\nabla_{\mathbf{y}} \mathcal{L}_{\mathrm{s}}\left(d_{0/*}\right)\right) & \\ \mathbf{y}_{[0]} = \mathbf{y} & \text{for spoofing} \\ \mathbf{y}_{[0]} = \mathbf{y}_0 & \text{for removal} \end{cases} \tag{23}$$

To control imperceptibility, we constrain the distortion budget ϵ. When using multiple steps m, we reduce the step size accordingly ($\gamma = \frac{\epsilon}{t}$). In the end, we quantize the result to

[5] We also experimented with matching the expected or actual detection scores (argmax $\|\mathcal{D}(\mathbf{y}_0, \mathbf{k}) - d_0\|_2$), but such attacks were not successful.

Algorithm 1 Pseudo-code for content fingerprint generation.

```
Input: $\mathbf{y}$ (or $\mathbf{x}$)                    ▷ input image patch
Input: $p$                                               ▷ projection patch size ($p \times p$)
Input: $n_\mathrm{g}$                                    ▷ number of Gaussian filtering steps
Input: $n_\mathrm{h}$                                    ▷ number of hash bits
Input: $\tau$                                            ▷ random projection threshold
Input: $q$                                               ▷ number of sampled hash bits
Input: $(w, h)$                                          ▷ fingerprint patch size ($w \times h$)
Input: $(\kappa, b)$                                     ▷ contex: camera key, patch location
  $\mathbf{y} \leftarrow$ Standardize($\mathbf{y}$)      ▷ standardize and reshape ($1 \times p^2$)
  Seed PRNG with ($\kappa$)
  $\mathbf{P} \sim \mathcal{U}(0,1)^{n_h \times p \times p}$   ▷ sample random projection matrices
  for $i \in 1 \;...\; n_g$ do
      $\mathbf{P} \leftarrow$ GaussianBlur($\mathbf{P}$)    ▷ blur to emphasize low frequencies
  end for
  $\mathbf{P} \leftarrow$ Normalize($\mathbf{P}$)        ▷ normalize and reshape ($n_h \times p^2$)
  $\mathbf{h} \leftarrow |\mathbf{y}\mathbf{P}^T| > \tau$   ▷ robust-hash computation
  $\mathbf{u} \leftarrow \mathbf{0}^{n_h \times w \times h}$
  for $i \in 1 \;...\; n_h$ do
      $[p_1, ..., p_q] \leftarrow q$-element permutation of $\mathbf{h}$
      Seed PRNG with $(\kappa, b, h[p_1], ..., h[p_q])$
      $\mathbf{u}[i] \sim \mathcal{U}(-1,1)^{w \times h}$   ▷ sample individual uniform components
  end for
  return $\sqrt{\frac{3}{n}} \sum_n \mathbf{u}$         ▷ aggregate to a $w \times h$ fingerprint, $\approx \mathcal{N}(0, \mathbf{I})$
```

obtain valid `uint8` images. This makes the attack flexible and remains fast and manageable in large-scale evaluation[6].

c) AFE: Having access to the encoder (and ideally the camera ISP as well), the attacker can spoof the fingerprint by embedding its approximation $\mathbf{k}_{0/*}$. (For PRNU this is similar to the transfer attack, but is more faithful as it modulates the embedding strength in a multiplicative manner.) This attack assumes the attacker possesses a raw version of the target image (or can invert the ISP [68, 76]). Formally:

$$\mathbf{y}_{0/\approx} = \mathcal{I}\left(\mathcal{E}(\mathbf{x}, \mathbf{k}_{0/*})\right) \tag{24}$$

D. Fingerprint Generation

We evaluate two main approaches to defend against spoofing: generation of the fingerprint based on local image content, and adversarial training which models and penalizes spoofing in the training loop. We discuss both approaches below.

Standard PRNU fingerprints are static, which makes them straightforward to estimate and spoof. In contrast, computational fingerprints can be generated dynamically which allows for better control over their security and privacy properties. Depending on the application at hand and the assumed threat model, various options can be considered:

- *static fingerprints* (e.g., unique for a camera model) could be useful for non-adversarial applications and would alleviate privacy leaks related to unique device fingerprints.
- dynamic *sampled fingerprints* are generated based on a secret key and context of capture (time, location, nonce, etc.); they require possession of the right source image with the valid fingerprint and enforce estimation from a single photograph. (In some settings, e.g., when addressing local photo manipulation, the source image may be readily available.)
- dynamic *content fingerprints* are generated based on local image content and further raise the bar for the attacker by requiring search for fingerprint collisions.

Controlled fingerprint generation allows for adaptation to various applications. It may also be desirable to embed multiple fingerprints with different properties.

E. Content Fingerprints

Content fingerprints provide a separate layer of security which directly prevents spoofing by requiring the attacker to look for fingerprint collisions within the same capture context. The fingerprint can be derived from a robust hash which is sensitive to content replacement, but remains robust to benign processing (brightness adjustment, compression, etc.).

For simplicity, we use a well studied hash function [77] and add an image standardization step to facilitate cross-domain matching, i.e., the encoder operates in the RAW domain, while the detector in the RGB domain. For RGB images, we downsample the patch to fixed dimensions, take the average over color channels and normalize the input to have 0 mean and unit standard deviation. For RAW images, the process is similar, but takes into account pixel alignment in the Bayer array and adds gamma correction before normalization.

The next step is to derive a robust hash. For each requested bit of the hash (n_h), a random projection matrix is sampled ($\mathbf{P} \sim \mathcal{U}(0,1)^{p \times p}$), blurred a few times to focus on low frequencies and normalized. Absolute value of projected image content is then compared to a threshold τ. Finally, a $w \times h$ fingerprint $\mathbf{k}_{\{x/y\}}$ is generated by progressive accumulation of uniformly distributed random noise images $\sim \mathcal{U}(-1,1)^{w \times h}$ seeded by camera key, patch location and q randomly chosen bits of the hash. Such an approach leads to an approximately Gaussian fingerprint that changes with content replacement, but remains similar under benign post-processing. We summarize the process in Algorithm 1 and refer to [77] for details. Learning a better visual hash is an interesting problem for future work.

F. Adversarial Training

An *adversarial training protocol* includes additional defense terms $\mathcal{L}_\mathrm{a}$ in the training loss:

$$\mathcal{L}_\mathrm{adv} = \alpha \left\| \mathbf{y} - \mathbf{y}_0 \right\|_2 + \mathcal{L}_\mathrm{d}(d_0, d_1) + \lambda \mathcal{L}_\mathrm{a} \tag{25}$$

e.g., to penalize detection of adversarial images in the AD[7]:

$$\mathcal{L}_\mathrm{a} = \mathcal{L}_{-}(d_{0/*}) : d_{0/*} = \mathcal{D}(\mathbf{y}_{0/*}, \mathbf{k}'_0) \tag{26}$$

where $\mathcal{L}_{-}$ denotes a negative detection penalty for the current detector architecture. This approach is analogous to adversarial training from computer vision [78] and aims to directly reject

[6] We also experimented with C&W attacks under the L_2 distortion model [75] and with full gradient magnitude information. However, such attacks proved difficult to work with due to essential projection to 8-bit precision. Rounding every 10-20 steps often works well, but we were unable to choose attack parameters that would consistently yield good results.

[7] We also experimented with other defenses, including penalties on adversarial fingerprint estimation, e.g., correlation with the true fingerprint. While we obtained promising preliminary results, this approach proved difficult to work with and comprehensive threat modeling turned out to be impractical. We discuss our key observations and the issues we encountered in Section V-F.

spoofed samples. In this paper, we focus on two adversarial inputs: $\mathbf{y}_{0/+}$ and $\mathbf{y}_{0/\nabla}$ which correspond to ART and AIO attacks, respectively.

During adversarial training we sample an additional batch for the attacker, e.g., for fingerprint transfer or adversarial spoofing. In transfer attacks, we use random strength $\beta \sim \mathcal{U}(0.01, 0.02)$. For adversarial spoofing, we use *multi-step PGD* (Section IV-C) with L_∞ distortion budget $\epsilon \sim \mathcal{U}(\{\frac{1}{255}, \frac{2}{255}\})$, a random number of steps (uniform choice from $\{1, ..., 5\}$), and the correct secret fingerprint. The resulting images are quantized with standard `uint8` precision.

V. Experimental Evaluation

In this section, we compare the computational and intrinsic sensor fingerprints and discuss their relationship with digital watermarking. We begin by describing our experimental setup and proceed to assess detection performance, visual examples and security properties.

A. Simulation Environment and Setup

We implemented all experiments in our *neural imaging toolbox*, a high-fidelity simulation environment built from scratch in Python 3.x and Tensorflow 2.x. The toolbox allows for end-to-end modeling and optimization of the entire photo acquisition and distribution pipeline and enables interesting feedback loops, e.g., optimization of the camera ISP [11, 27], lossy image codecs [28] or imaging sensors (this study) based on image analysis performance at the end of complex distribution channels. Our toolbox provides various components that can be assembled into arbitrary pipelines to model diverse imaging applications; some of the key components include the camera ISP (conventional and neural), JPEG codecs (standard and fully differentiable), and a lossy learned codec.

All experiments used the same common setup (Fig. 1) and the same components (JPEG codec, camera ISP, etc.) - the models differed only in the sensor/detector blocks. We simulate dynamic fingerprints by sampling from a Gaussian distribution. For PRNU, we consider both estimated and ground truth fingerprints and both the standard DWT and learned CNN denoisers (Supplement B). For the CSF we use compact CNNs with $n = 4$ layers and $f = 16$ feature channels which comes down to ≈22k parameters (the correlation-based detector with 2 branches has ≈42k parameters). We used small models to facilitate deployment in a constrained runtime of the sensor. Since the specification of the inference accelerator is unknown, and the reported computational resources vary significantly [13–15], we leave exploration of optimal model architectures for future work.

We implemented our own JPEG codec, which approximates rounding with a soft relaxation to enable gradient computation in the backward pass [27] (standard rounding is used in the forward pass). This leads to a differentiable codec that closely approximates the standard libJPEG (Fig. A.2 and Fig. A.3). For camera ISP, we used a standard pipeline with simple NN-based demosaicing. We discuss both components in detail in supplementary materials (Supplement A).

In most experiments we use photos from Nikon D90 (RAISE [79] dataset) and instantiate different devices by sampling multiple sensor fingerprints ($\mathbf{k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$). During development, we also used Canon EOS-40D (MIT-5k [80] dataset) to validate generalization of basic components. For both cameras, we collected 150 full-resolution images and split them into 100:50 for various training/validation tasks. Given full resolution images, we sample patches randomly (while maintaining CFA alignment). Training patches are sampled in each iteration (which serves as an additional augmentation mechanism), while validation patches are sampled only once; for each validation image, we sampled 5 patches, leading to 250 test images. We work with 128×128 px patches, which are commonly used in local PRNU analysis. For model training, we include JPEG compression with QF $\sim \mathcal{U}(80, 100)$ and later vary the QF in various validation experiments.

When measuring detection performance, we focus on two main metrics: AUC corresponds to the area under the ROC curve; and TPR-1 corresponds to the true positive rate in the fixed 1% false positive rate regime.

B. The Distortion-Detection Trade-off

The distortion-detection trade-off shows how effectively a fingerprinting system exchanges the incurred image distortion for detection performance. To simulate different camera instances, we sampled 100 fingerprints. In Fig. 2 we compare detection statistics for both the correlation-based and end-to-end CSF systems with the same JPEG compression strength as used at the training time (QF$\sim \mathcal{U}(80, 100)$). In Fig. 3 we show the entire detection-distortion trade-off for stronger compression outside of the training range (QF=50) and compare example CSF systems against PRNU fingerprints. We sweep various embedding strengths to explore the limits of the inherent embedding-detection model and not a particular instance determined by a specific sensor.

For PRNU, we include results with both CNN/DWT-based denoisers and ground-truth/estimated sensor fingerprints. Using ground-truth fingerprints is possible only in simulations and can be seen as an upper bound of the detection capability and is equivalent to spread spectrum watermarking (similarity, which we explore in detail in Section V-C). For fingerprint estimation, we used 50 residuals from random images. While PRNU is an intrinsic watermark with no explicit distortion control, we obtain the trade-off curve by modulating the embedding strength α in (1). In practice, each sensor exhibits its own inherent embedding strength; comparison of detection responses in values reported for real PRNU [81] leads us to suspect that the typical value could be $\alpha \approx 0.004$. Regardless of the embedding strength, the plot shows that the learned CSF systems are more effective and can deliver more reliable detection at lower cost (distortion). For reference, we show full ROC curves in the bottom subplot of Fig. 3.

This confirms that a learned system can outperform intrinsic sensor fingerprints by a large margin. Adding proactive content protection mechanisms into digital cameras does not need to introduce excessive image distortion.

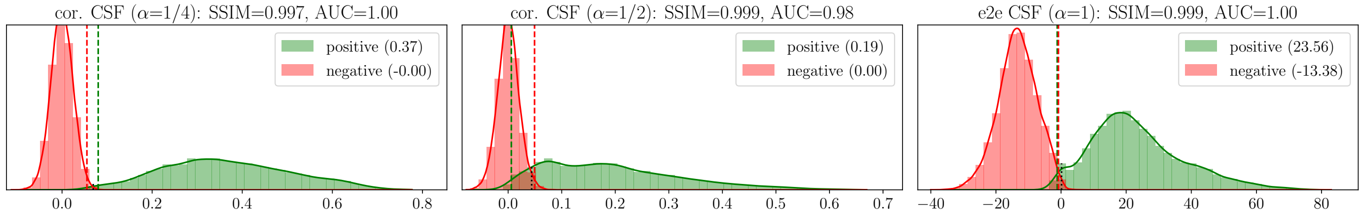


Fig. 2. Distributions of the detection statistics for vanilla CSF systems (both *correlation-based* and *end-to-end*) trained at different quality levels α. The models were tested on image patches compressed with JPEG at random quality levels from [80,100]. Numbers in brackets show the sample means.

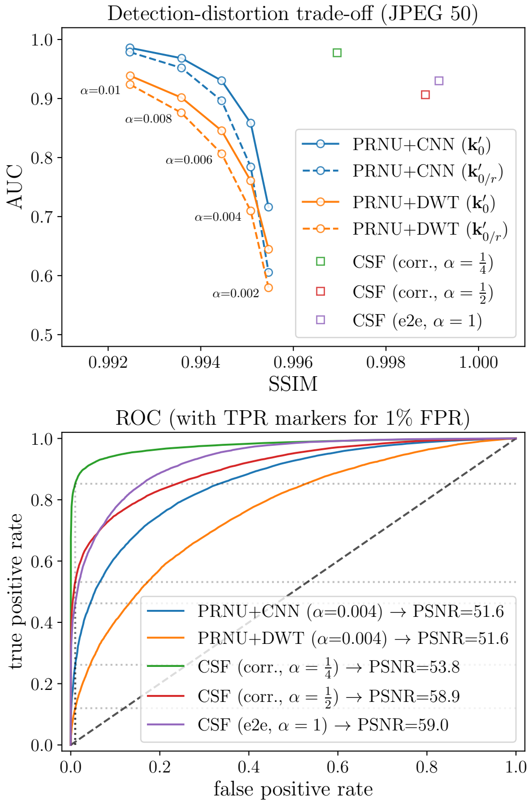


Fig. 3. The detection-distortion trade-off and ROC curves for various sensor fingerprinting systems tested on JPEG images (QF=50); CSF was trained on JPEG with QF sampled from [80,100].

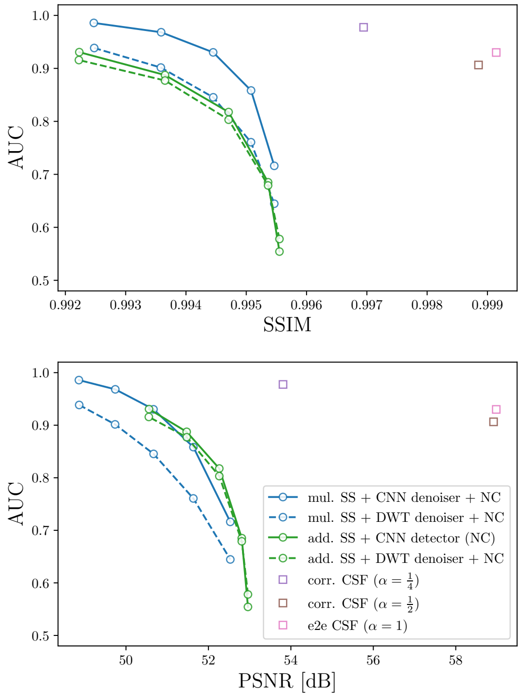


Fig. 4. Comparison with classic spread spectrum watermarking (detection of a known fingerprint in RGB images compressed with JPEG QF=50).

TABLE III
SUMMARY OF THE COMPARED EMBEDDING-DETECTION STRATEGIES

Embedding rule	Detector (processing)	Detection statistic
$x(1+\alpha k_0)$	DWT denoiser	NCC
$x(1+\alpha k_0)$	CNN denoiser[1]	NCC
$x+\alpha k_0$	DWT denoiser	NCC
$x+\alpha k_0$	2-branch CNN[2]	NCC
$x-\mathcal{E}(\mathbf{x}_{[\pm s]},\mathbf{k}_{0[\pm s]})$	2-branch CNN[2]	NCC
$x-\mathcal{E}(\mathbf{x}_{[\pm s]},\mathbf{k}_{0[\pm s]})$	e2e CNN[2]	fully connected NN

[1] pretrained separately for multiplicative noise extraction (Supplement B-A)
[2] trained for the task at hand (Section III-D)

C. Comparison with Digital Watermarking

As discussed in Section III-A, both the intrinsic and computational sensor fingerprints can be seen as digital watermarks. In particular, PRNU is a simple "free" watermark with no control over its properties (including the embedding strength); it is equivalent to multiplicative spread spectrum watermarking [66]. On the other hand, the proposed CSF can be seen as a generalization of spread spectrum watermarking with automatic end-to-end adaptation to complex distribution channels and different embedding and detection domains.

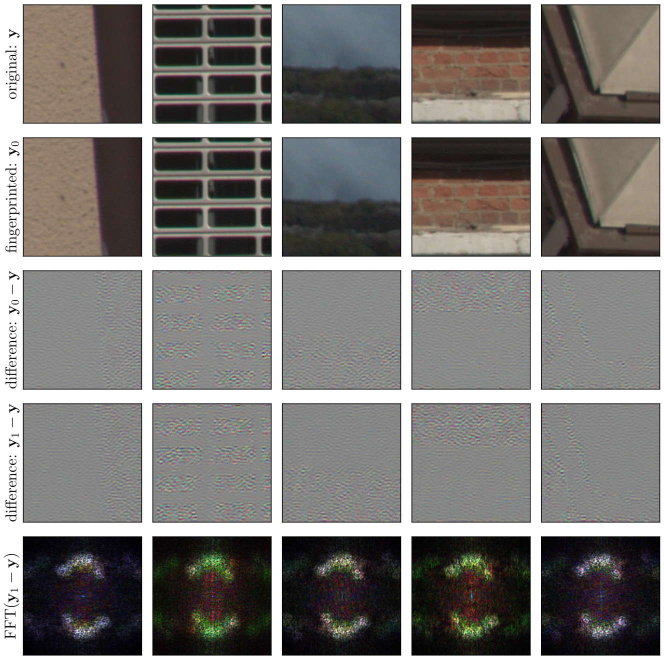
In this experiment, we provide a detailed comparison of our model with spread spectrum watermarking. We carefully compare all systems in the same conditions and include both the additive and multiplicative embedding rules:

$$x_0 = x + \alpha k_0 \tag{27}$$

$$x_0 = x(1 + \alpha k_0) \tag{28}$$

The remaining pipeline remains unchanged, i.e., we still perform cross-domain embedding/detection and use the camera ISP and lossy compression as the distribution channel. Since the tested watermark is known to the detector, this corresponds to our experiments with ground truth fingerprints.

For each of the embedding strategies, we test two different detectors. In addition to the standard one, we also include


original: $\mathbf{y}$
fingerprinted: $\mathbf{y}_0$
difference: $\mathbf{y}_0 - \mathbf{y}$
difference: $\mathbf{y}_1 - \mathbf{y}$
FFT($\mathbf{y}_1 - \mathbf{y}$)

(a) CSF $\alpha = 0.25$, SSIM = 0.996, PSNR = 51.6 dB

Fig. 5. Example fingerprinted images (2nd row) and isolated embedding distortion. Rows 3-4 compare the distortion for two random fingerprints. Successive columns compare changes with image content. The fingerprint remains imperceptible despite delivering good detection performance.

learning-based models: for multiplicative spread spectrum, we include the CNN-based model that was separately pre-trained for noise extraction (Supplement B-A); for additive spread spectrum, we use the correlation-based detector (Eq. 7) with 2 branches of 4 convolutional layers (32 filters each) trained in the full detection loop with QF$\sim \mathcal{U}(80, 100)$ in the channel. We summarize all tested models in Tab. III.

We show the obtained results in Fig. 4 which compares the detection-distortion trade-off (for QF=50) using both SSIM and PSNR. Both spread spectrum techniques lead to similar detection performance and preference for either depends on the metric of interest. Benefits of learning-based detectors varied from negligible to incremental, which suggests that the embedding strategy is indeed the bottleneck. Despite small CNN models the CSF system consistently delivered better results than our spread spectrum baselines. This confirms that neural networks are an effective solution to learning end-to-end data hiding systems for complex channels (here a combination of camera ISP and lossy compression with cross-domain embedding-detection). At the same time, we acknowledge that our baselines were standard spatial-domain watermarks and better systems can be obtained with additional research and development (e.g., transition to a transform domain). That being said, it would also be interesting to explore other neural network architectures, especially with real-world hardware constraints.

D. Visual Examples

Fig. 5 shows typical examples of fingerprinted images ($\mathbf{y}_0$) and true residuals ($\mathbf{y}_{\{0,1\}} - \mathbf{y}$). We show how the embedding distortion changes with image content (column-wise) and sampled fingerprints (rows 3 and 4 compare residuals obtained with two orthogonal fingerprints $\mathbf{k}_0$ and $\mathbf{k}_1$). The last row illustrates the frequency allocation of the learned embedding strategy (FFT domain). The distortion usually appears as a semi-regular texture with orientation and frequency selection changing across training repetitions. The magnitude of embedding distortion changes locally based on image content, but we did not observe strong content-adaptivity of the learned embedding strategy. The incurred image distortion is mostly imperceptible and leads to good detection performance at high fidelity levels (e.g., $\alpha = 0.25$ with SSIM≈0.997; cf. Fig. 2 and Fig. 3). Interestingly, the frequency allocation varies with training repetition, image fidelity and detection statistic (we show more visual examples in Fig. C.2).

TABLE IV
PARAMETERS OF THE ROBUST HASH AND CONTENT FINGERPRINT

Parameter	**Symbol**	**Value**
patch size for projection	p	64
projection threshold	τ	10
number of Gaussian filtering steps	n_g	4
number of hash bits (total)	n_h	50
number of hash bits (sampled)	q	[4,30]

E. Impact of Content Fingerprints

In this section, we assess cross-domain (RAW-RGB) fingerprint similarity and evaluate its impact on detection performance. At the sensor level, we compute a robust hash from a raw image $\mathbf{x}$ and derive a fingerprint $\mathbf{k}_x$. On the detector's side we use a JPEG-compressed RGB image, yielding $\mathbf{k}_y$. We investigate how the number of sampled hash bits (q) affects both detection performance and spoofing attempts in two representative black-box and white-box attacks. We seed the PRNG only with bits sampled from the robust hash to simulate camera key and patch location agreement.

The number of sampled hash bits controls the robustness-security trade-off. The fingerprint needs to be sensitive to content replacement and robust to benign post-processing (e.g., lossy compression). Sampling more bits makes collisions less likely, but also reduces legitimate matching performance. We sweep q from 4 to 30 (out of 50; see Tab. IV for all parameters) and assess how the choice impacts several key metrics for a vanilla correlation-based CSF ($\alpha = 0.25$).

Fig. 6(a) shows how q impacts detection performance on 250 images from 100 different cameras (different fingerprints) under strong JPEG compression with QF $\sim \mathcal{U}(50, 100)$. The detection performance peaks for $q \approx 12$ and then starts to slowly deteriorate. Overall, the process is robust and works well despite cross-domain (RAW-RGB) matching and lossy compression. Fig. 6(d) shows histograms of fingerprint correlations for same and different image pairs.

Similar number of sampled bits ($q \approx 15$) seems to suffice to obtain good resistance to spoofing. Fig. 6(bc) show success rates (out of 10,000 attempts) for two representative attacks:

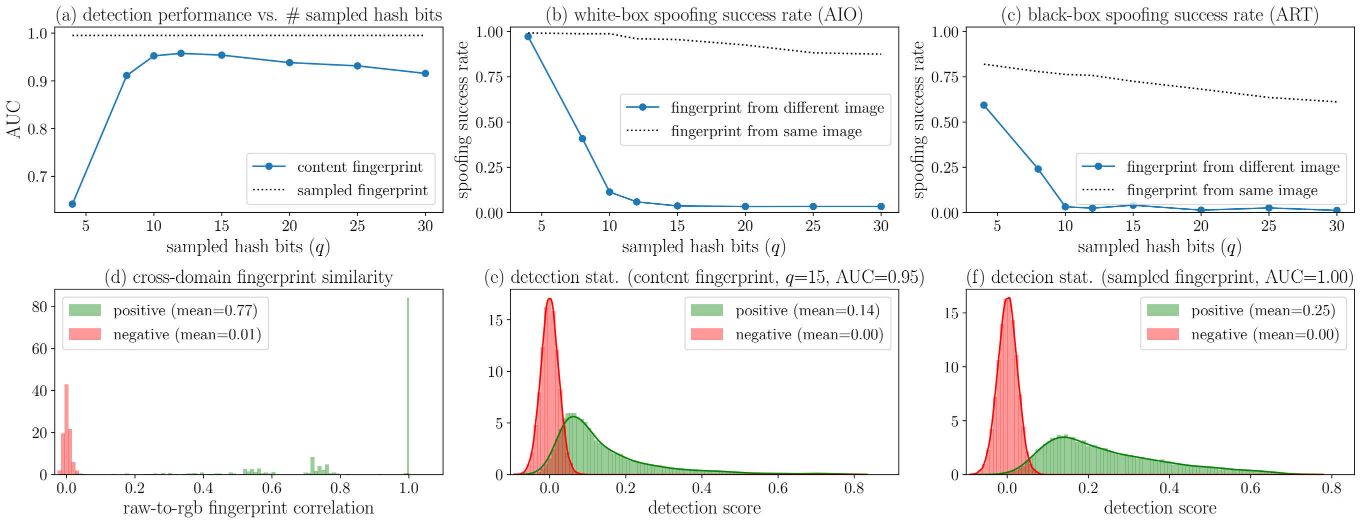


Fig. 6. Impact of content fingerprints (with different number of sampled hash bits q) on the detection performance and spoofing success rates: (a) detection AUC compared with sampled random fingerprints; (b) and (c) spoofing success rates for white-box AIO and black-box ART attacks; (d) cross-domain (RAW-RGB) fingerprint similarity with JPEG compression; (e) and (f) comparison of detection statistics for content and random fingerprints.

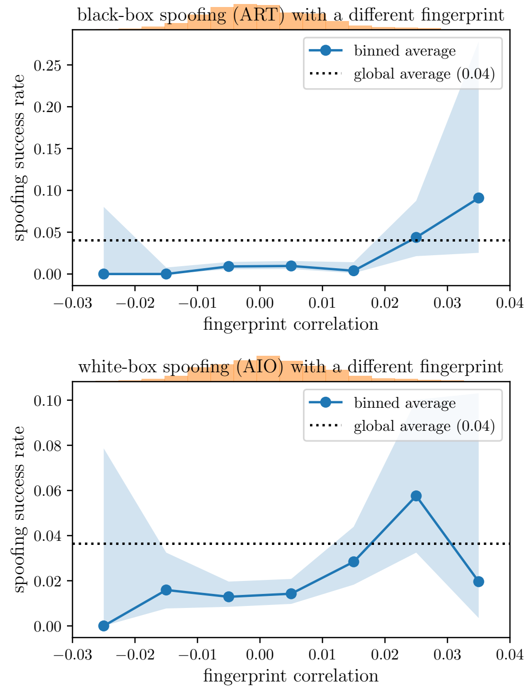


Fig. 7. Detailed breakdown of spoofing success rates by content fingerprint correlation: (top) black-box transfer attack; (bottom) white-box optimization attack. Shaded regions corresponds to the Wilson confidence intervals at the 95% level. Frequency of observed cases is shown as histograms above each plot. For clarity, outliers that inflate the mean are not shown.

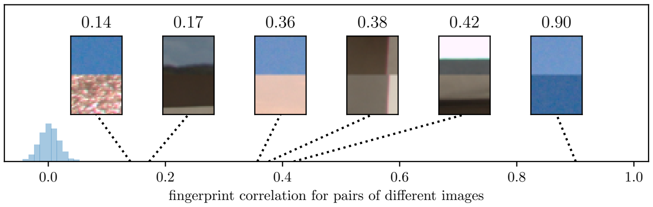


Fig. 8. Illustration of content fingerprint collisions: distribution of fingerprint correlation scores across 2,000 different image pairs and 6 top-ranking cases - despite differences in color and brightness, the patches share similar structure.

- in the *black-box* setting we use the ART attack and add a normalized residual (from a 3×3 filter $\mathbf{f}_1$) modulated to yield MSE $\approx \frac{1}{255}$;
- in the *white-box* setting we perform the multi-step AIO attack driven by an adversarial fingerprint; the attacker uses the PD to maximize the detection response - first to estimate the fingerprint ($\mathbf{k}'_{0/d}$), and then the target image; we constrained the L_∞ budget of the attack to $\epsilon = \frac{1}{255}$.

Both attacks are formally defined in (20) and (23). As expected, the white-box attack consistently reaches better success rates. The solid lines in Fig. 6(bc) depict attacks driven by content fingerprints extracted from a different image (randomly selected). We also show baseline success rates (dotted black line) for spoofing using a fingerprint estimate obtained from the same content (still cross-domain). This validates the attacks and establishes baseline performance.

The discrepancy between the success rates demonstrates that using content fingerprints is an effective way to make spoofing attacks more difficult. We note that the process is stochastic and the performance varies with content similarity. While the reduction is significant, the success rates cannot be made arbitrarily small (in this experiment, the average saturates at ≈4%). This is caused by naturally occurring collisions within our validation images - sampling random patches extracts some nearly-empty patches (e.g., sky) that are likely to collide. This artificially inflates success rates since real-world forgeries are unlikely to involve replacement of empty blocks with other instances of empty blocks. Disregarding outliers and including only cases with fingerprint correlation <0.1 yields average success rates of 1.7% and 0.9%. That being said, a comprehensive assessment that takes into account

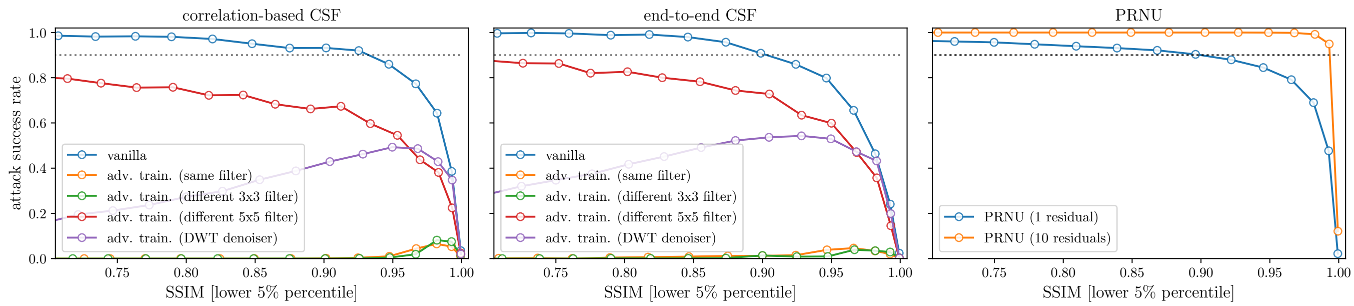


Fig. 9. Reduction in the success rates of the residual transfer attack (ART) for various sensor fingerprinting systems and several residual filters: (col. 1) typical correlation-based CSF system with and without residual transfer penalty ($d_{+/0}$); (col. 2) typical end-to-end CSF system with and without residual transfer penalty; (col. 3) standard PRNU with increasing number of images available to the attacker.

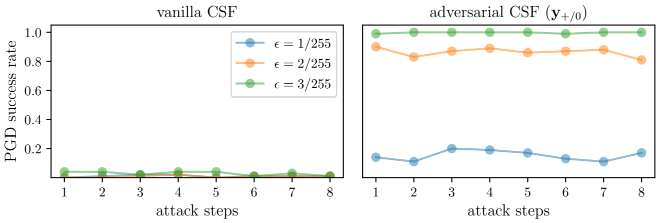


Fig. 10. Unexpected pathology in the adversarially learned detection metric: including residual transfer attacks enables PGD spoofing in the PD based on a random fingerprint.

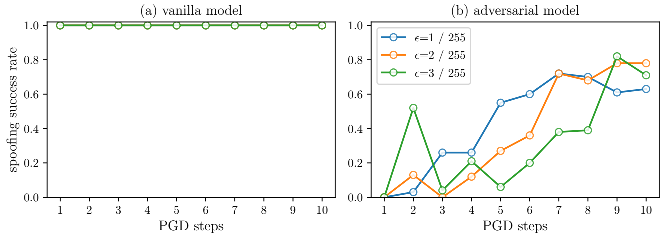


Fig. 11. Success rate of AIO attacks driven by the true fingerprint for a vanilla (a) and adversarial model (b). Attacks against vanilla models always succeed, regardless of the distortion budget (ϵ) or the number of steps. Adversarial model shows some resistance, but fails to generalize to diverse attack settings.

realistic manipulations (including subtle ones, like face or text replacement) is an interesting topic for future work.

To obtain more insights, we break the results down by fingerprint similarity (Fig. 7). Greater fingerprint correlations translate to greater success rates and the increase starts to accelerate $\gtrsim 0.02$. For presentation clarity, we omitted the rare outliers that typically succeed (and inflate the average success rate) and show them separately instead. Fig. 8 shows 6 top-ranking pairs of different images that yielded the highest fingerprint correlation among 2,000 attempts. The highest scoring cases correspond to high-level structure similarity and are most common for empty patches.

Finally, Fig. 6(ef) compare detection statistics for sampled ($\mathbf{k}_0$) and content fingerprints ($\mathbf{k}_{\{x/y\}}$). While the former have the advantage of perfect knowledge (the detector simply re-samples the secret fingerprint), the content fingerprint remains remarkably robust, even in the presence of lossy compression. Detection performance degradation is limited, with AUC dropping from 0.995 to 0.954 and TPR-1 from 0.962 to 0.741 for JPEG QF $\sim \mathcal{U}(50, 100)$. Further parameter fine-tuning or even learning a novel visual hash optimized for this scenario should lead to even better performance. Overall, using content fingerprints appears to be a viable solution with good detection performance and significantly reduced attack surface.

F. Impact of Adversarial Training

We summarize representative results for two example attacks: ART and multi-step gradient-based AIO.

a) ART Attacks: Modeling residual transfer consistently leads to significant reduction in attack success rates. Penalizing addition of a high-frequency residual from a simple 3×3 filter ($\mathbf{f}_1$) nearly eliminates this attack and shows non-trivial generalization to other filters. We show the trade-off between the attack success rate and the incurred image distortion (measured as lower SSIM percentile to better illustrate the effect) in Fig. 9. Successive columns correspond to a correlation-based detector, an end-to-end detector, and standard PRNU.

For vanilla CSF systems (blue line), the attack steadily improves with image distortion and eventually reaches ≈100% success rate. For adversarially trained models, this drops to ≈0% (with a small temporary increase to ≈10%) when the attacker uses the same filter as the defense ($\mathbf{f}_1$). The effect still persists, although to a various degree, when the filters do not match. For a different 3×3 filter ($\mathbf{f}_2$), the behavior remains unaffected. For the standard DWT filter used in PRNU analysis, the success rate initially increases (typically up to 25-50%) and then drops again for larger image distortion. The defense was the barely effective for a 5×5 filter ($\mathbf{f}_3$).

Overall, modeling transfer attacks has a strong effect with non-trivial (although ultimately insufficient) degree of generalization. We obtained similar results for both detector architectures (Fig. 9). However, the cause for the behavior appears to differ substantially. For the correlation-based detector, embedding appears to move to lower frequencies, which suggests that the defense may simply be shifting the fingerprint outside of the bands retained by the filter. While both the attacker and the defense can adapt, leading to an adversarial dynamic, we expect the attacker has the upper hand.

For the end-to-end detector the embedding distortion remains similar to the vanilla model, which indicates that the learned detection metric changes in an different way. In follow up in-depth experiments we discovered that it can aggravate unexpected failure modes in gradient-based attacks in the PD - an attacker can succeed by following gradients obtained with a random fingerprint and his chances keep increasing with the distortion budget (Fig. 10). It is not clear how to rigorously understand behavior of a learned detection metric and how to discover and prevent similar pathologies. Learned detection metrics appear to be poorly suited for sensitive applications prone to adversarial inputs.

For reference, we show the corresponding results for PRNU (Fig. 9, 3rd column), which exhibits analogous behavior as the vanilla CSF. When multiple images with the same fingerprint are available, the effectiveness of the attack improves rapidly. Averaging merely 10 residuals (from randomly chosen natural images) leads to a 100% effective attack at a negligible distortion, which emphasizes the pitfalls of static fingerprints.

Finally, we note that modeling ART attacks reduced detection performance. For the correlation-based model, we needed to increase importance of the detection loss ($\alpha \searrow 0.1$), yielding still imperceptible distortion with SSIM 0.994. Although AUC dropped only slightly (from 1.0 to 0.98), the deterioration happens primarily in the low FPR regime - TPR-1 dropped from nearly 1 to 0.91. For the end-to-end model the deterioration was not as significant, AUC dropped from 1.0 to 0.99 and TPR-1 from 0.99 to 0.94.

b) AIO Attacks: Adversarial training reduces efficacy of PGD attacks, but does not eliminate them, even for variants presented during training. In this experiment, we used a multi-step PGD attack (Section IV-C) driven by the true fingerprint and constrained by an L_∞ distortion budget ϵ. The attack always succeeds against vanilla trained models - regardless of the distortion budget and the number of steps (Fig. 11a).

For an adversarially trained model the spoofing efficacy can drop substantially, but the effect has insufficient generalization and tends to vary across attack settings and training repetitions. We show success rates for an example end-to-end CSF system in Fig. 11b. The model was trained to withstand ART attacks as well as AIO attacks with $\epsilon \sim \mathcal{U}(\{\frac{1}{255}, \frac{2}{255}\})$ and < 6 steps (with JPEG QF $\sim \mathcal{U}(80, 100)$). We can observe that using more steps leads to more effective spoofing, even for attack configurations observed at the training time. While a different detection architecture could possibly handle this better, we did not observe substantial improvements even for much larger detectors (we tried models with 32 and 64 filters with 85k and 325k parameters, respectively).

G. Key Observations on Security Evaluation

We performed extensive experiments to assess susceptibility to representative attacks within various threat models. We summarize our key observations and example results below.

Our first observation is that single-use dynamic fingerprints are not sufficient to resist spoofing. Nearly all attacks succeeded against both standard PRNU and vanilla CSF systems - although with only 1 image available to the forger, many attacks turned out to be less effective against PRNU's pixel-wise multiplicative embedding. Even though estimation from a single image is prone to contamination with its content and generally yields low correlation with the true fingerprint, such adversarial estimates typically suffice for at least some attacks to fool a vanilla detector.

Secondly, we learned that adversarial training is insufficient and is overly complex for comprehensive attack modeling. While for some attacks significant improvements are possible, the defenses are typically not fully effective and generalization to unseen attack settings varies. Most defenses require a trainable detection metric (used in our end-to-end detector), which remains an inexplicable black box prone to unexpected failure modes upon in-depth exploratory analysis.

Attacks involving proxy models proved particularly challenging. We identified a few generic strategies that can hinder adversarial fingerprint estimation in the PD and inhibit various attacks across a broad range of settings. However, the process is brittle and addressing multiple attack vectors simultaneously often led to unstable training and diverse model behavior, which remains non-trivial to measure. For example, we learned that targeting low similarity with the secret fingerprint proved insufficient. Despite nearly zero correlation and overlapping detection statistics, the adversarial fingerprint may still possess vestigial predictive capacity sufficient to drive gradient-based spoofing attacks to occasional success. We eventually adopted a few mitigation strategies (a vanilla warmup period, periodic checkpoints to keep the best model) and obtained several promising models, but ultimately were unable to control this behavior consistently.

VI. Discussion and Limitations

The emerging neural sensors open a new direction for proactive photo authentication methods and enable content protection at the very beginning of its digital life-cycle. This allows for processing protected raw images with 3rd party digital darkroom software and, given proper implementation in hardware, would not be susceptible to counter-forensics using tampered camera firmware [26]. Running NN models directly on raw pixels allows for end-to-end optimization, including adaptation to complex, analytically intractable channels. Compared to intrinsic sensor fingerprints, we consistently obtained much better robustness at a lower image distortion even for simple models. This demonstrates that adoption of proactive protection mechanisms does not need to negatively impact image quality - one of the key aspects for camera vendors.

A next-generation sensor fingerprinting system can also meet various security & privacy requirements. Control over fingerprint generation creates new opportunities for protection against spoofing attacks - which are trivial for PRNU. We experimented with two different approaches to the problem. First, we performed extensive evaluation of adversarial training which proved to be insufficient and difficult to work with. Significant improvements are possible for some attacks, e.g., residual transfer, but efficacy and generalization capabilities vary. Moreover, obtaining comprehensive resistance to multiple attacks proved to be problematic - in particular when addressing proxy models. We encountered problems with training stability, large variation of security properties and unexpected failure modes. We briefly explained our key observations in Sections V-F and V-G.

We obtained more promising results with content fingerprints, which change depending on the local image content and provide a separate layer of security. As a first step, we used a simple and popular fingerprint derived from a robust visual hash [77] and demonstrated that it works well in a cross-domain (RAW-RGB) setting. It yields minimal deterioration of detection performance while providing strong resistance to spoofing. While it does not eliminate the possibility of spoofing entirely, it significantly raises the bar for an attacker. It currently appears to be the most reliable way of addressing the problem. Although it remains a best-effort solution, it may still be useful in practice. Many image forgeries rely on easily accessible tools and lack sophistication.

Adoption of computational sensor fingerprints can be beneficial beyond adversarial use-cases. Even simple interventions to improve fingerprint detection in difficult areas (dark/textured) would lead to considerable performance improvements and allow for more precise manipulation localization [3]. Moreover, dynamic fingerprint generation eliminates the need for fingerprint estimation and storage, which improves detection performance and scalability. PRNU is a full resolution real-valued fingerprint which requires dozens of megabytes for each supported camera instance.

We experimented with two distinct architectures of the CSF detector: a 2-branch CNN based on *correlation* and an *end-to-end* model based on fully connected layers. They deliver similar detection performance, but the former behaves more consistently and is understandable to humans. While the latter approach is more expressive and could potentially learn a better, data-driven detection metric, it remains an inexplicable black-box prone to unexpected behavior - especially on adversarial inputs. A more rigorous approach based on metric learning [82] and intuitive explanation of the learned detection rule could be an interesting direction for future research given the fundamentally asymmetric nature of the comparison (the image undergoes different distortions than an adversarially estimated fingerprint).

One of the primary limitations of the current work consists in its simulated character. We have implemented a high-fidelity simulation environment that provides multiple components for modeling photo acquisition and distribution pipelines. We performed various experiments to validate the models, including both qualitative and quantitative comparison of our differentiable JPEG codec (Fig. A.2) and generalization of our CNN-based PRNU extraction to real-world images [83] (Supplement A). While these experiments were successful, more effort is needed to validate the CSF system against industry-standard camera ISPs and to deploy it in real hardware.

References

[1] M. Chen *et al.*, “Determining image origin and integrity using sensor noise,” *IEEE Trans. Inf. Forensics Secur.*, vol. 3, no. 1, 2008.

[2] G. Chierchia *et al.*, “A bayesian-mrf approach for prnu-based image forgery detection,” *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 4, 2014.

[3] P. Korus and J. Huang, “Multi-scale analysis strategies in prnu-based tampering localization,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, 2017.

[4] F. Marra *et al.*, “Blind prnu-based image clustering for source identification,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 9, 2017.

[5] E. Durmus, P. Korus, and N. Memon, “Every shred helps: Assembling evidence from orphaned jpeg fragments,” *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 9, 2019.

[6] D. Valsesia *et al.*, “User authentication via prnu-based physical unclonable functions,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, 2017.

[7] S. Taspinar, M. Mohanty, and N. Memon, “Source camera attribution using stabilized video,” in *IEEE Int. Workshop on Information Forensics and Security*, 2016.

[8] M. Hosseini and M. Goljan, “Camera identification from hdr images,” in *ACM Workshop on Information Hiding and Multimedia Security*, 2019.

[9] M. Iuliani, M. Fontani, and A. Piva, “A leak in prnu based source identification—questioning fingerprint uniqueness,” *IEEE Access*, vol. 9, pp. 52 455–52 463, 2021.

[10] S. Joshi *et al.*, “Empirical evaluation of prnu fingerprint variation for mismatched imaging pipelines,” *arXiv:2004.01929*, 2020.

[11] P. Korus and N. Memon, “Neural imaging pipelines-the scourge or hope of forensics?” *arXiv:1902.10707*, 2019.

[12] C. Chen *et al.*, “Learning to see in the dark,” in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2018.

[13] Sony, *Sony to release world’s first intelligent vision sensors with ai processing functionality*, https://www.sony.net/SonyInfo/News/Press/202005/20-037E/, Accessed Aug 2020.

[14] L. Bose *et al.*, “A camera that cnns: Towards embedded neural networks on pixel processor arrays,” in *IEEE/CVF Int. Conf. on Computer Vision*, 2019.

[15] L. Mennel *et al.*, “Ultrafast machine vision with 2d material neural network image sensors,” *Nature*, vol. 579, no. 7797, 2020.

[16] M. F. Amir *et al.*, “3-d stacked image sensor with deep neural network computation,” *IEEE Sensors Journal*, vol. 18, no. 10, 2018.

[17] E. Wengrowski and K. Dana, “Light field messaging with deep photographic steganography,” in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2019.

[18] M. Tancik, B. Mildenhall, and R. Ng, “Stegastamp: Invisible hyperlinks in physical photographs,” in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[19] *Camera forensics*, https://cameraforensics.com/, Visited in Jan 2021.

[20] Witness, *Ticks or it didn’t happen: Confronting key dilemmas in authenticity infrastructure for multimedia*, https://lab.witness.org/ticks-or-it-didnt-happen/, 2019.

[21] Truepic, https://truepic.com, Visited in Aug 2020.

[22] *Serelay*, https://serelay.com, Visited in Aug 2020.

[23] *Verified witnessing: Turn your photos and videos into secure, signed visual evidence*, https://github.com/guardianproject/proofmode, Visited Feb 2021.

[24] The Coalition for Content Provenance and Authenticity, *C2PA Technical Specifications*, https://c2pa.org/, Visited Sept 2021.

[25] *Content authenticity initiative*, https://contentauthenticity.org/, Visited July 2020.

[26] D. Baracchi *et al.*, “Camera obscura: Exploiting in-camera processing for image counter forensics,” *Forensic Science International: Digital Investigation*, 2021.

[27] P. Korus and N. Memon, “Content authentication for neural imaging pipelines: End-to-end optimization of photo provenance in complex distribution channels,” in *IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[28] P. Korus and N. Memon, “Quantifying the cost of reliable photo authentication via high-performance learned lossy representations,” in *Int. Conf. on Learning Representations*, 2020.

[29] B. Bayar and M. Stamm, “Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection,” *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 11, 2018.

[30] P. Korus, “Digital image integrity–a survey of protection and verification techniques,” *Digital Signal Processing*, vol. 71, 2017.

[31] L. Verdoliva, “Media forensics and deepfakes: An overview,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, 2020.

[32] T. Bianchi and A. Piva, “Image forgery localization via block-grained analysis of jpeg artifacts,” *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, 2012.

[33] P. Korus and J. Huang, “Multi-scale fusion for improved localization of malicious tampering in digital images,” *IEEE Trans. Image Processing*, vol. 25, no. 3, 2016.

[34] S. Agarwal and H. Farid, "Photo forensics from rounding artifacts," in *ACM Workshop on Information Hiding and Multimedia Security*, 2020.

[35] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, 2012.

[36] O. Mayer and M. C. Stamm, "Forensic similarity for digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, 2019.

[37] D. Cozzolino and L. Verdoliva, "Noiseprint: A cnn-based camera model fingerprint," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, 2019.

[38] F. Marra *et al.*, "A full-image full-resolution end-to-end-trainable cnn framework for image forgery detection," *arXiv:1909.06751*, 2019.

[39] Y. Wu, W. AbdAlmageed, and P. Natarajan, "Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[40] N. R. Council *et al.*, *Strengthening forensic science in the United States: a path forward*. National Academies Press, 2009.

[41] M. Barni, M. Stamm, and B. Tondi, "Adversarial multimedia forensics: Overview and challenges ahead," in *European Signal Processing Conf.*, 2018.

[42] X. Zhao, C. Chen, and M. C. Stamm, "A transferable anti-forensic attack on forensic cnns using a generative adversarial network," *arXiv preprint arXiv:2101.09568*, 2021.

[43] I. Cox *et al.*, *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[44] J. Zhu *et al.*, "Hidden: Hiding data with deep networks," in *Proc. European conference on computer vision (ECCV)*, 2018, pp. 657–672.

[45] N. Raj and R Shreelekshmi, "A survey on fragile watermarking based image authentication schemes," *Multimedia Tools and Applications*, vol. 80, no. 13, pp. 19 307–19 333, 2021.

[46] X. Zhang *et al.*, "Reference sharing mechanism for watermark self-embedding," *IEEE Trans. Image Processing*, vol. 20, no. 2, 2011.

[47] P. Korus and A. Dziech, "Efficient method for content reconstruction with self-embedding," *IEEE Trans. Image Processing*, vol. 22, no. 3, 2012.

[48] P. Korus, J. Białas, and A. Dziech, "Towards practical self-embedding for jpeg-compressed digital images," *IEEE Trans. Multimedia*, vol. 17, no. 2, 2014.

[49] J. Kelsey, B. Schneier, and C. Hall, "An authenticated camera," in *Annual Computer Security Applications Conf.*, 1996.

[50] D. Bhowmik and T. Feng, "The multimedia blockchain: A distributed and tamper-proof media transaction framework," in *IEEE Int. Conf. on Digital Signal Processing*, 2017.

[51] JPEG, *Towards a standardized framework for media blockchain and distributed ledger technologies*, http://ds.jpeg.org/whitepapers/jpeg-media-blockchain-whitepaper.pdf, Visited Jan 2021, 2019.

[52] C.-P. Yan and C.-M. Pun, "Multi-scale difference map fusion for tamper localization using binary ranking hashing," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 9, 2017.

[53] O. Kapah and H. Z. Hel-Or, "Demosaicking using artificial neural networks," in *Applications of Artificial Neural Networks in Image Processing V*, 2000.

[54] R. Tan *et al.*, "Color image demosaicking via deep residual learning," in *IEEE Int. Conf. Multimedia and Expo*, 2017.

[55] M. Gharbi *et al.*, "Deep joint demosaicking and denoising," *ACM Tran. Graphics*, vol. 35, no. 6, 2016.

[56] E. Schwartz, R. Giryes, and A. M. Bronstein, "Deepisp: Toward learning an end-to-end image processing pipeline," *IEEE Trans. Image Processing*, vol. 28, no. 2, 2018.

[57] A. Ignatov, L. Van Gool, and R. Timofte, "Replacing mobile camera isp with a single deep learning model," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops*, 2020.

[58] G. Eilertsen *et al.*, "HDR image reconstruction from a single exposure using deep CNNs," *ACM Tran. Graphics*, vol. 36, no. 6, 2017.

[59] G. Qian *et al.*, "Trinity of pixel enhancement: A joint solution for demosaicking, denoising and super-resolution," *arXiv:1905.02538*, 2019.

[60] Y. Wang *et al.*, "Practical deep raw image denoising on mobile devices," in *European Conf. Computer Vision*, 2020.

[61] J. N. Martel *et al.*, "Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2020.

[62] A. Ferdowsi and W. Saad, "Deep learning-based dynamic watermarking for secure signal authentication in the internet of things," in *IEEE Int. Conf. on Communications*, 2018.

[63] N. Yu *et al.*, "Artificial fingerprinting for generative models: Rooting deepfake attribution in training data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 448–14 457.

[64] M. Mascicpinto and F. Pérez-González, "Putting the prnu model in reverse gear: Findings with synthetic signals," in *European Signal Processing Conf.*, 2018.

[65] K. Wei *et al.*, "A physics-based noise formation model for extreme low-light raw denoising," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2020.

[66] I. J. Cox *et al.*, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Processing*, vol. 6, no. 12, pp. 1673–1687, 1997.

[67] A. Mehrish, A. V. Subramanyam, and S. Emmanuel, "Robust prnu estimation from probabilistic raw measurements," *Signal Processing: Image Communication*, vol. 66, 2018.

[68] T. Brooks *et al.*, "Unprocessing images for learned raw denoising," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2019.

[69] N. Bonettini *et al.*, "Fooling prnu-based detectors through convolutional neural networks," in *European Signal Processing Conf.*, IEEE, 2018.

[70] M. Goljan, J. Fridrich, and M. Chen, "Defending against fingerprint-copy attack in sensor-based camera identification," *IEEE Trans. Inf. Forensics Secur.*, vol. 6, no. 1, pp. 227–236, 2010.

[71] M. Barni, H. Santoyo-Garcia, and B. Tondi, "An improved statistic for the pooled triangle test against prnu-copy attack," *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1435–1439, 2018.

[72] L. Bondi *et al.*, "Improving prnu compression through preprocessing, quantization, and coding," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, 2018.

[73] R. Zhang *et al.*, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2018.

[74] R. Radhakrishnan, Z. Xiong, and N. D. Memon, "Security of the visual hash function," in *Security and Watermarking of Multimedia Contents V*, International Society for Optics and Photonics, vol. 5020, 2003.

[75] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, IEEE, 2017.

[76] S. Zamir *et al.*, "Cycleisp: Real image restoration via improved data synthesis," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[77] J. Fridrich and M. Goljan, "Robust hash functions for digital watermarking," in *IEEE Int. Conf. on Information Technology: Coding and Computing*, 2000.

[78] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," *arXiv:1706.06083*, 2017.

[79] D. Dang-Nguyen *et al.*, "Raise: A raw images dataset for digital image forensics," in *ACM Multimedia Systems Conf.*, 2015.

[80] V. Bychkovsky *et al.*, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[81] M. Kirchner and C. Johnson, "Spn-cnn: Boosting sensor-based source camera attribution with deep learning," in *IEEE Int. Workshop Information Forensics and Security*, 2019.

[82] Y. Duan *et al.*, "Deep adversarial metric learning," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2018.

[83] D. Shullani *et al.*, "Vision: A video and image dataset for source identification," *EURASIP Journal on Information Security*, vol. 2017, no. 1, p. 15, 2017.

Supplementary Materials for Manuscript ”Computational Sensor Fingerprints”

Paweł Korus and Nasir Memon

List of appendices:

A. Neural Imaging Toolbox Components
B. PRNU Synthesis Results
C. Additional Materials

Appendix A
Neural Imaging Toolbox Components

In Section V, we briefly introduced our simulation toolbox. Here, we provide additional details about the key components used in our experiments.

A. Camera ISP

We use a simple camera ISP with NN-based demosaicing and typical post-processing: standard sRGB color conversion (based on true conversion tables extracted from photo meta-data) and gamma correction. We skip global brightness normalization to enable operation on image patches without a broader context.

The demosaicing model $\mathcal{M}$ is a fully convolutional NN with 4 layers (32 filters of size 3×3) and a final linear projection (1×1 convolution) yielding a residual wrt. bilinear interpolation. The model has 29k parameters and was trained separately to minimize the MSE loss on 8,192 RGB mixed natural images with diverse content (natural photographs and computer graphics) [28]. The images were down-sized to increase the level of detail and eliminate any previous imaging artifacts. Color channels were sampled at the training time according to a random 2×2 Bayer filter. Fig. A.1 shows example images developed by our ISP.

B. Differentiable JPEG Codec

We implemented a *diffJPEG* codec which closely approximates the standard JPEG, but remains differentiable and can be plugged into the training loop. Successive steps are implemented as matrix multiplication, reshaping and convolution layers:

- RGB to/from YCbCr color-space conversions are implemented as 1×1 convolutions.
- Isolation of 8×8 blocks for independent processing is implemented by combining *space-to-depth* and reshaping operations.
- 2D discrete cosine transforms are implemented by matrix multiplication (DxD^T where x denotes an 8×8 array, and D is the transformation matrix).
- Division/multiplication of DCT coefficients by the corresponding quantization steps are implemented as element-wise operations with tiled and concatenated quantization matrices (both the luminance and chrominance channels).
- The actual quantization is approximated by a continuous function $\rho(x)$ - see details below.

The key problem in making JPEG fully differentiable lies in the rounding of DCT coefficients. To obtain a differentiable relaxation, we started with a Taylor series expansion and eventually converged to a single sinusoidal term with adjusted phase matched to a sawtooth function:

$$\rho(x) = x - \frac{\sin(2\pi x)}{2\pi}$$

This approximation is used only in the backward pass (during gradient computation). In the forward pass, we use the standard rounding operator, which leads to compression results equivalent to libJPEG. In Fig. A.2 we compare image quality (measured by SSIM) across various QF settings which reveal a nearly perfect match. In Fig. A.3 we compare an example image patch, which also looks nearly identical across various quality levels. The biggest discrepancies can be observed for extremely low quality factors (QF ≈15) and above QF ≈95, which is the maximal level supported by our version of libJPEG (all values above 95 resulted in identical images).

Fig. A.1. Example image patches (256 × 256 px) developed with our ISP model with neural demosaicing (Nikon D90 from the RAISE dataset [79]).

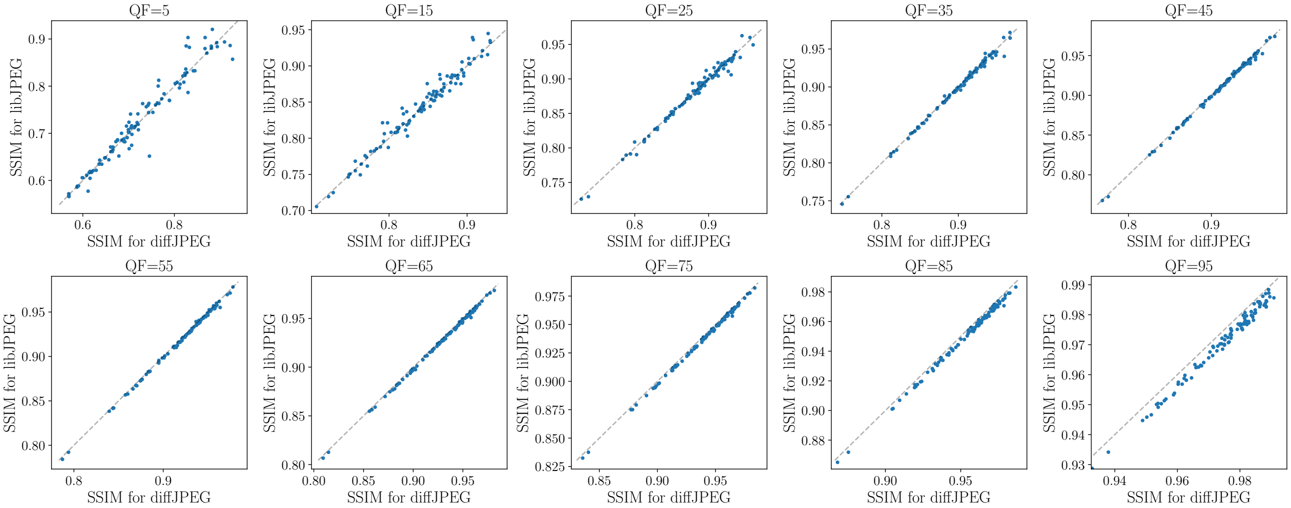


Fig. A.2. Comparison of the incurred image distortion for images compressed with our differentiable diffJPEG codec and the standard libJPEG.

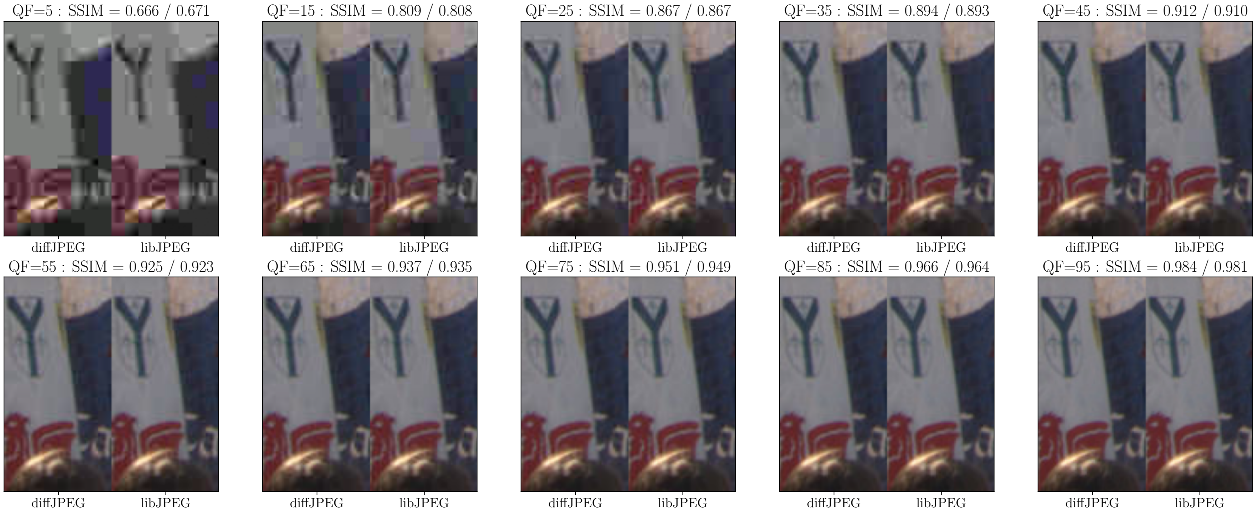

QF=5 : SSIM = 0.666 / 0.671
QF=15 : SSIM = 0.809 / 0.808
QF=25 : SSIM = 0.867 / 0.867
QF=35 : SSIM = 0.894 / 0.893
QF=45 : SSIM = 0.912 / 0.910
QF=55 : SSIM = 0.925 / 0.923
QF=65 : SSIM = 0.937 / 0.935
QF=75 : SSIM = 0.951 / 0.949
QF=85 : SSIM = 0.966 / 0.964
QF=95 : SSIM = 0.984 / 0.981
diffJPEG
libJPEG

Fig. A.3. Example image patch (128 × 128 px) compressed with our differentiable diffJPEG codec and the standard libJPEG.

Appendix B
PRNU Synthesis Results

We provide additional results that explore the behavior of synthetic PRNU fingerprints and highlight their potential for training better denoisers. Since PRNU is a latent signal, it can only be estimated in real-world experiments. Our simulation toolbox brings new capabilities to the research community, such as precise assessment of fingerprint estimation quality and its impact on detection performance, or large-scale error rate assessment by generating an arbitrary number of unique fingerprints. While a detailed exploration of all new capabilities is out of scope of this work, we include some exploratory results below.

A. PRNU Estimators and Denoiser Training

Throughout the paper, we use the MLE estimator[8] with: (1) the standard wavelet-based denoising [1], (2) a neural network-based residual. We use the same generic convolutional network architecture as for the other models (e.g., demosaicing), i.e.: 10 convolutional layers with 32 filters of size 3×3 and a final projection layer with a 1×1 convolution. All intermediate layers use LeakyReLU activations. The model is fully convolutional (can be used for any image size) and outputs a 3-channel residual multiplied by a trainable scalar.

We train the model $\mathcal{R}_\mathrm{k}$ by minimizing the MSE loss w.r.t. the ground truth fingerprint mapped into the RGB domain. We embed the fingerprint using the standard multiplicative model (1). In each step, we sample a training batch $\mathbf{x}$ (cropped randomly from full-resolution images from Nikon D90), a new ground truth fingerprint $\mathbf{k}_0$, an embedding strength $\alpha \sim \mathcal{U}(0.01, 0.05)$, and JPEG quality factor QF $\sim \mathcal{U}(85, 100)$:

$$\underset{\mathcal{R}_\mathrm{k}}{\operatorname{argmin}} \underset{\mathbf{x}}{\mathbb{E}} \underset{\mathbf{k}_0}{\mathbb{E}} \underset{\mathrm{QF}}{\mathbb{E}} \underset{\alpha}{\mathbb{E}} \left\| \mathcal{R}_\mathrm{k} \circ \mathcal{C}_\mathrm{QF} \circ \mathcal{I}(\mathbf{x} + \alpha \mathbf{x} \cdot \mathbf{k}_0) - \mathcal{M}(\mathbf{k}_0) \right\|_2$$

We use zero-mean averaging and dispense with Wiener filtering, which consistently hurt performance in all experiments (both on simulated and real PRNUs; Section B-C).

B. Exploratory Analysis with Synthetic Fingerprints

Fig. B.1 compares fingerprint estimation examples for the wavelet (top 2 rows) and CNN residuals (bottom 2). The images correspond to the red channel of the ground-truth fingerprint in the RAW and RGB domains (col. 1/2) and several estimates obtained from increasing number of natural images compressed with JPEG QF=90 (col. 3-7).

Visually speaking, the CNN extracts a cleaner fingerprint. While the wavelet denoiser retains an obvious blocking grid even after averaging 150 residuals, the CNN yields a grid-free output from only 5 residuals. Given the same number of input images, the CNN's output consistently correlates better with the ground truth fingerprint (col. 1): e.g., using 50 images yields correlations (with $\mathbf{k}_0'$) of 0.21 and 0.36 for the wavelet and CNN denoisers, respectively. While the CNN tends to over-smooth the estimate, the actual fingerprint indeed exhibits spatial correlations in the RGB domain (from demosaicing).

The improved fingerprint estimates lead to better detection performance. Fig. B.2 shows example detection statistics for both hypotheses (obtained from 50 different fingerprints). We used images from Canon EOS-40D (instead of D90 used at the CNN training time) and sampled 300 patches for each simulated camera (note that we keep the same ISP and change only the sRGB conversion tables). We set the embedding strength ($\alpha = 0.004$) outside of the denoiser training range to match typical real-world response strength [81]. The experiment was performed on 128×128 px patches compressed with JPEG QF=90. As expected, we see consistent improvement across various metrics (accuracy and true positive rate at 1% false positive rate). We also show detection statistics for the ground truth fingerprint (∞ images) which shows that there is still space for improvement even w.r.t. 150 images. Fig. B.3 summarizes this behavior for various JPEG quality factors. Detection metrics saturate more quickly than fingerprint correlation and 50-100 images are typically sufficient (depending on compression settings).

C. Performance on Real PRNU and libJPEG

We ran additional experiments to assess model generalization to real-world PRNU. We sampled 150 central 128×128 crops for 33 cameras from the natural image subset of the Vision dataset [83]. The images were split into 50 images for PRNU estimation and 100 for detection assessment. We repeated the experiment 50 times - each time with a different estimation-testing split. We compare 3 methods: (1) standard DWT denoising with Wiener filtering, (2) DWT denoising without Wiener filtering, (3) our CNN model trained on synthetic PRNU. Fig. B.4 shows the obtained results for images compressed with libJPEG QF=90 (different quality factors and estimation-testing splits show analogous behavior).

An aggregated assessment on the entire dataset (1st col.) reveals that: (1) the commonly used Wiener filter deteriorates detection performance, (2) our CNN yields slight improvement over both baselines. Per-camera histogram (2nd col.) shows significant improvement of the CNN for a few cameras and similar results for the rest (without the Wiener filter). To shed some

[8] We used the open source Python implementation from Politecnico di Milano (https://github.com/polimi-ispl/prnu-python), but adapted it to include multiple engines: https://github.com/pkorus/prnulib

light on these cases, we show ROC curves for the worst, intermediate (25th and 75th percentiles) and best CNN performance (col. 3-6).

We can conclude that simulation of sensor fingerprints may be a useful tool for pre-training CNN denoisers which can translate into benefits for real devices. Interestingly, the top 4 cameras with the largest benefit from the CNN were made by Samsung (D01, D08, D22 and D26) and the bottom 5 by Apple (D05, D18, D20, D34 and D09). A full breakdown is shown in Tab. B.1. Further exploration of model architectures and training protocols could potentially leads to better results. We leave this problem for future work.

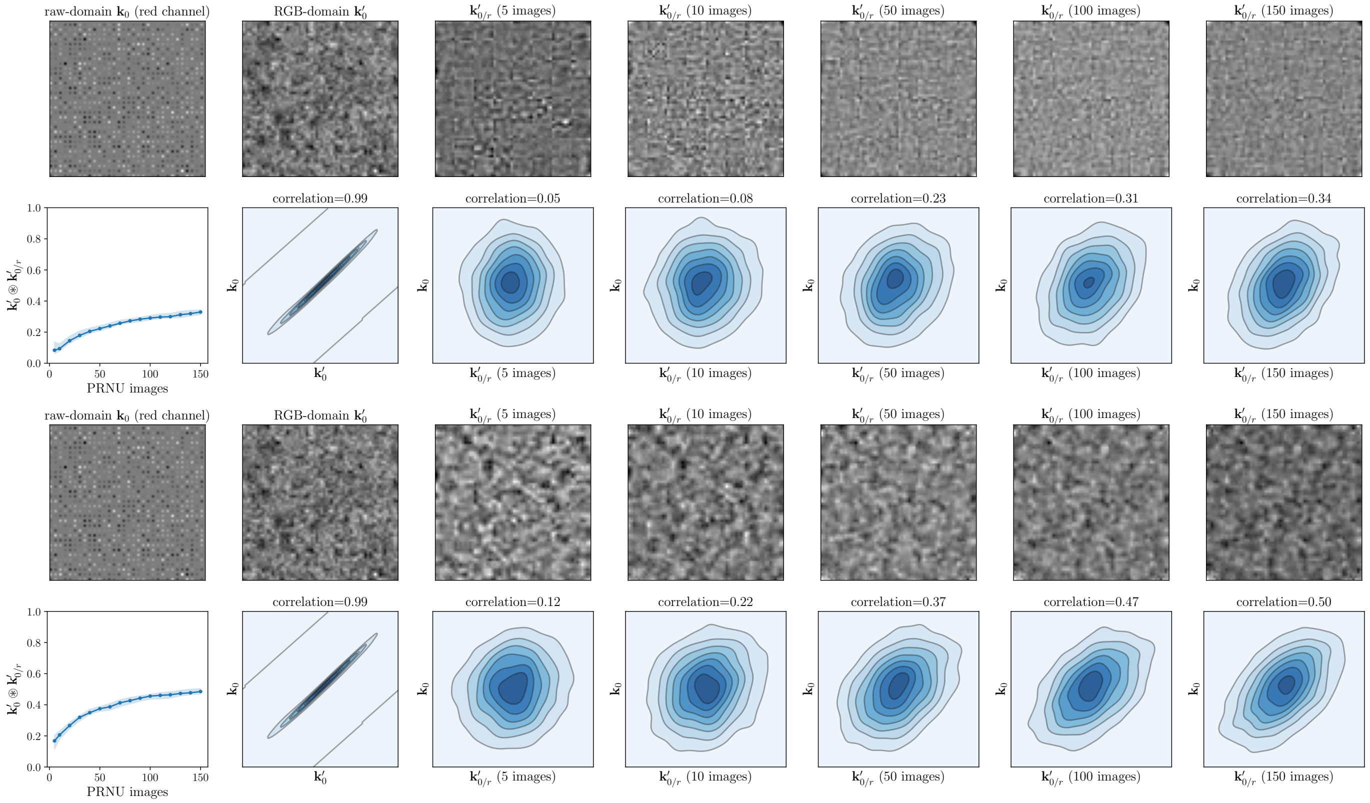


Fig. B.1. Comparison of PRNU estimation results for various numbers of available JPEG images (QF=90) for the wavelet (top 2 rows) and CNN residuals (bottom 2). Visual examples show the red channel of the original fingerprint in the RAW/RGB domains and various estimates with increasing numbers of available images. Scatter plots correlate the estimates with the ground truth.

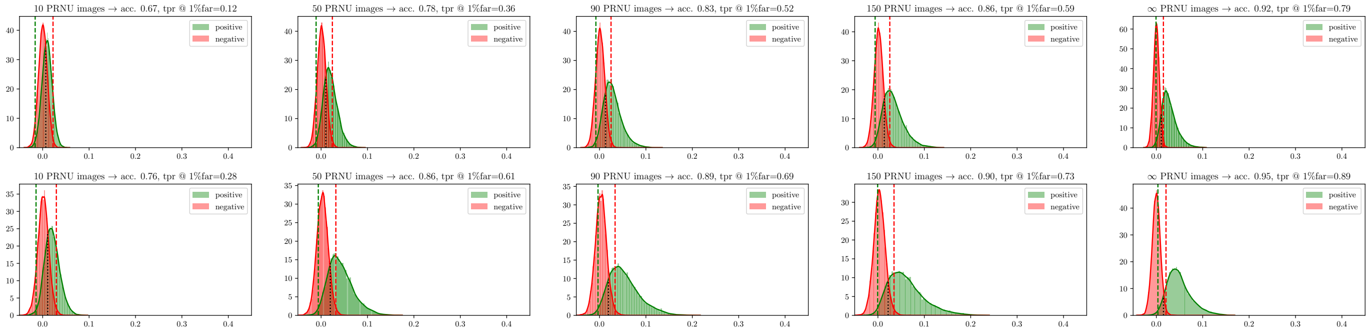


Fig. B.2. Simulated PRNU detection statistics obtained from JPEG images (QF=90) for the standard wavelet denoiser (top) and the CNN (bottom); columns correspond to various numbers of images (residuals) available for estimation - ∞ images represents the latent ground truth sensor fingerprint.

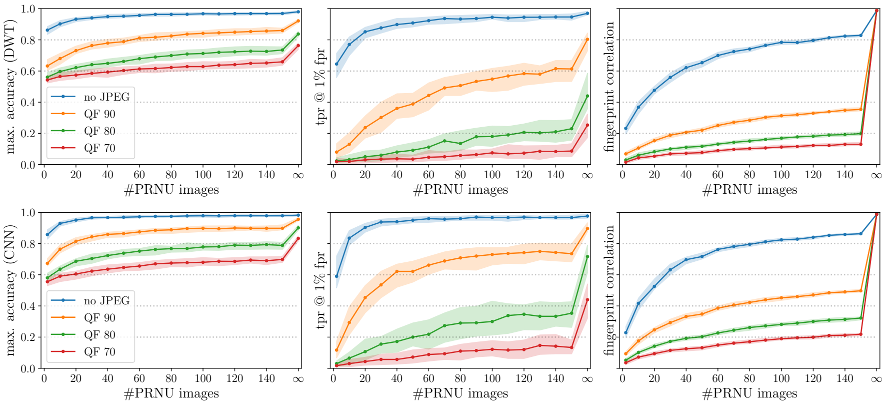


Fig. B.3. Simulated PRNU detection performance (accuracy and true positive rate at 1% false positive rate) and fingerprint estimation performance (correlation) for various numbers of available JPEG images (several QF settings).

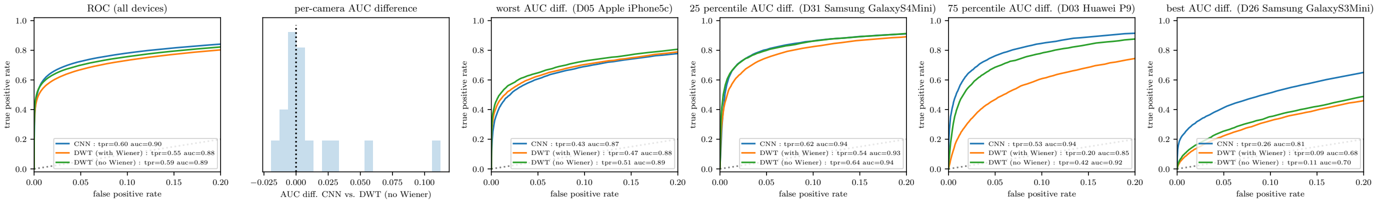


Fig. B.4. Sensor fingerprint (PRNU) detection performance for 33 real-world devices from the Vision dataset: our CNN denoiser trained on synthetic examples generalizes and yields an improvement over the standard DWT-based estimator - the improvement is incremental on average and substantial for some devices.

TABLE B.1
DEVICE BREAKDOWN OF REAL-WORLD PRNU DETECTION PERFORMANCE. TPR IS MEASURED AT A FIXED 1% FPR SETTING

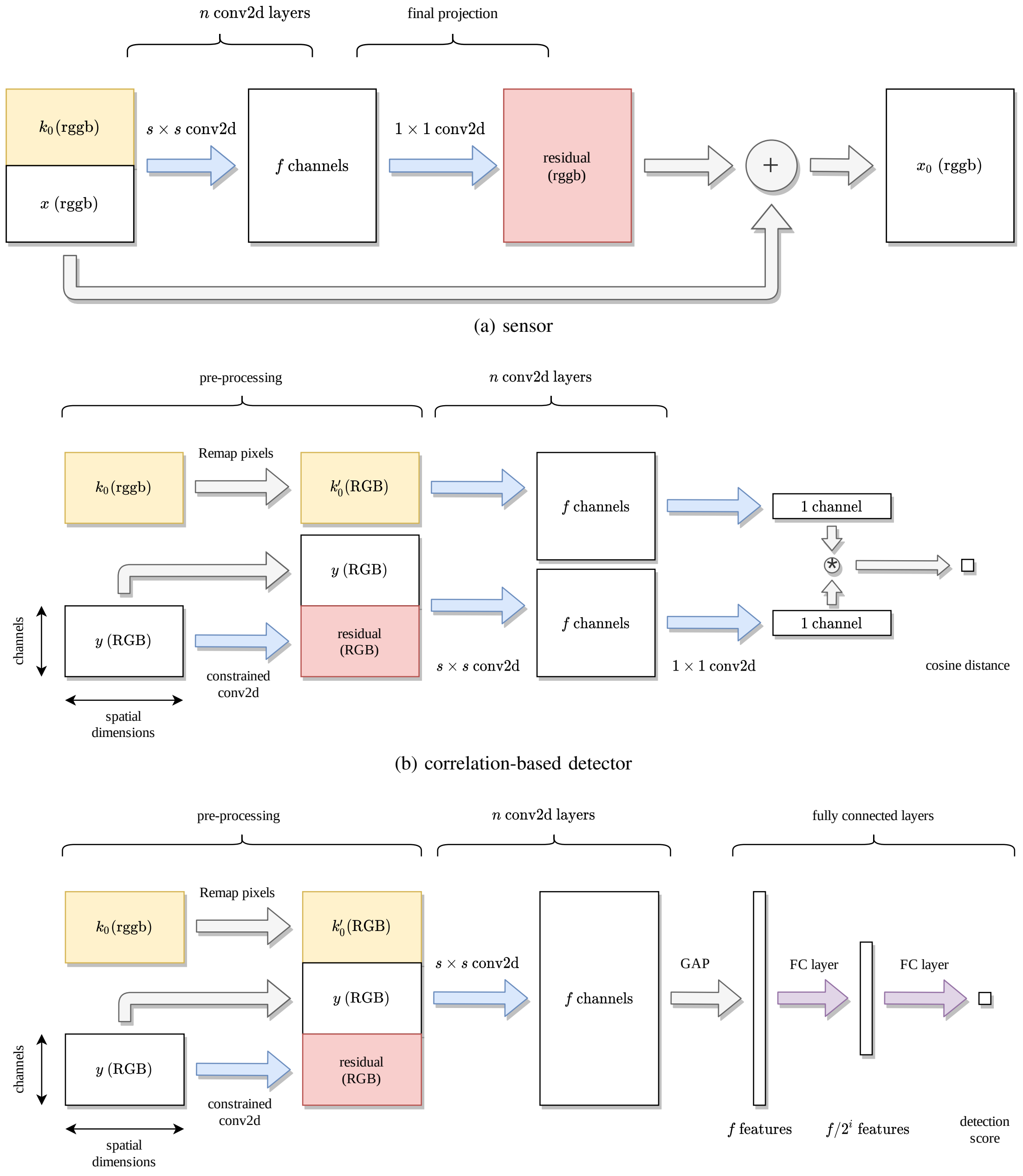
	CNN (no Wiener)			**DWT (Wiener)**			**DWT (no Wiener)**			**CNN vs DWT**	
Camera	d_0	AUC	TPR-1	d_0	AUC	TPR-1	d_0	AUC	TPR-1	ΔAUC	ΔTPR-1
D05_Apple_iPhone5c	0.026	0.867	0.427	0.022	0.878	0.470	0.024	0.887	0.510	-2.23%	-19.54%
D18_Apple_iPhone5c	0.028	0.869	0.517	0.023	0.868	0.509	0.025	0.884	0.544	-1.77%	-5.34%
D20_Apple_iPadMini	0.025	0.828	0.465	0.019	0.828	0.449	0.020	0.838	0.484	-1.17%	-4.13%
D34_Apple_iPhone5	0.031	0.905	0.560	0.025	0.905	0.570	0.027	0.914	0.616	-1.01%	-9.99%
D09_Apple_iPhone4	0.067	0.988	0.888	0.056	0.993	0.915	0.060	0.995	0.934	-0.78%	-5.13%
D04_LG_D290	0.036	0.942	0.690	0.025	0.942	0.620	0.027	0.949	0.668	-0.76%	3.07%
D27_Samsung_GalaxyS5	0.053	0.987	0.849	0.038	0.984	0.848	0.042	0.992	0.914	-0.51%	-7.58%
D32_OnePlus_A3003	0.041	0.838	0.592	0.022	0.832	0.515	0.026	0.842	0.563	-0.44%	4.96%
D31_Samsung_GalaxyS4Mini	0.035	0.939	0.623	0.023	0.930	0.541	0.026	0.943	0.636	-0.36%	-2.09%
D12_Sony_XperiaZ1Compact	0.067	0.972	0.895	0.049	0.973	0.863	0.054	0.975	0.882	-0.30%	1.45%
D14_Apple_iPhone5c	0.044	0.947	0.690	0.031	0.941	0.649	0.033	0.950	0.689	-0.26%	0.09%
D29_Apple_iPhone5	0.055	0.983	0.824	0.035	0.975	0.767	0.039	0.986	0.823	-0.26%	0.10%
D15_Apple_iPhone6	0.047	0.992	0.883	0.040	0.991	0.877	0.043	0.994	0.910	-0.20%	-3.10%
D17_Microsoft_Lumia640LTE	0.038	0.970	0.803	0.031	0.969	0.770	0.031	0.971	0.791	-0.09%	1.54%
D19_Apple_iPhone6Plus	0.063	0.989	0.903	0.042	0.982	0.847	0.047	0.989	0.898	-0.05%	0.58%
D07_Lenovo_P70A	0.124	1.000	0.990	0.066	0.998	0.973	0.077	0.999	0.982	0.09%	0.85%
D21_Wiko_Ridge4G	0.063	0.988	0.907	0.036	0.979	0.818	0.043	0.986	0.881	0.20%	2.80%
D24_Xiaomi_RedmiNote3	0.076	0.995	0.929	0.057	0.991	0.918	0.060	0.992	0.926	0.22%	0.28%
D10_Apple_iPhone4s	0.016	0.818	0.279	0.013	0.808	0.258	0.013	0.815	0.290	0.27%	-4.24%
D11_Samsung_GalaxyS3	0.052	0.972	0.815	0.032	0.961	0.725	0.035	0.968	0.768	0.39%	5.69%
D23_Asus_Zenfone2Laser	0.091	0.998	0.977	0.063	0.993	0.958	0.069	0.994	0.973	0.48%	0.41%
D30_Huawei_Honor5c	0.050	0.971	0.729	0.031	0.956	0.621	0.036	0.966	0.668	0.54%	8.39%
D02_Apple_iPhone4s	0.033	0.908	0.422	0.021	0.886	0.469	0.023	0.903	0.511	0.60%	-21.24%
D35_Samsung_GalaxyTabA	0.027	0.890	0.516	0.022	0.875	0.506	0.024	0.883	0.554	0.81%	-7.49%
D16_Huawei_P9Lite	0.056	0.937	0.667	0.037	0.915	0.636	0.042	0.925	0.666	1.35%	0.09%
D03_Huawei_P9	0.040	0.942	0.530	0.016	0.851	0.205	0.024	0.921	0.424	2.20%	19.97%
D13_Apple_iPad2	0.012	0.760	0.148	0.008	0.727	0.124	0.008	0.735	0.144	3.31%	2.70%
D25_OnePlus_A3000	0.015	0.751	0.289	0.007	0.707	0.166	0.008	0.723	0.211	3.67%	27.11%
D28_Huawei_P8	0.011	0.717	0.096	0.006	0.658	0.060	0.007	0.690	0.085	3.74%	10.88%
D01_Samsung_GalaxyS3Mini	0.019	0.828	0.347	0.010	0.764	0.181	0.011	0.773	0.203	6.60%	41.35%
D22_Samsung_GalaxyTrendPlus	0.020	0.823	0.387	0.010	0.756	0.173	0.011	0.769	0.201	6.67%	48.11%
D08_Samsung_GalaxyTab3	0.018	0.815	0.335	0.006	0.689	0.086	0.007	0.705	0.101	13.55%	69.70%
D26_Samsung_GalaxyS3Mini	0.014	0.808	0.258	0.006	0.684	0.091	0.007	0.696	0.114	13.92%	55.93%
Average	0.042	0.907	0.613	0.028	0.884	0.551	0.031	0.896	0.593	1.47%	6.55%

TABLE C.1
SUMMARY OF THE DATASETS USED IN OUR STUDY

Camera(s)	**Source**	**Resolution**	**Bayer**	**Used for**	**# images**
Nikon D90	RAISE dataset [79]	12 Mpx	GBRG	Framework validation, Training CNN denoiser, Synthetic PRNU experiments, CSF training and validation	150
Canon EOS-40D	MIT 5k dataset [80]	10 Mpx	RGGB	Framework validation, Synthetic PRNU experiments	150
Mixed natural images	Various sources [28]	256×256 px	-	Demosaicing training and validation	8,192 + 500
33 cameras	Vision dataset [83]	128×128 px	-	Real-world PRNU experiments	150×33

APPENDIX C
ADDITIONAL MATERIALS

- Summary of datasets used in this study (Tab. C.1),
- Network architectures for the considered encoder and detector models (Fig. C.1),
- More examples of the learned embedding distortion at various quality levels (Fig. C.2),
- Illustration of different variations of possible spoofing threat models (Fig. C.3).

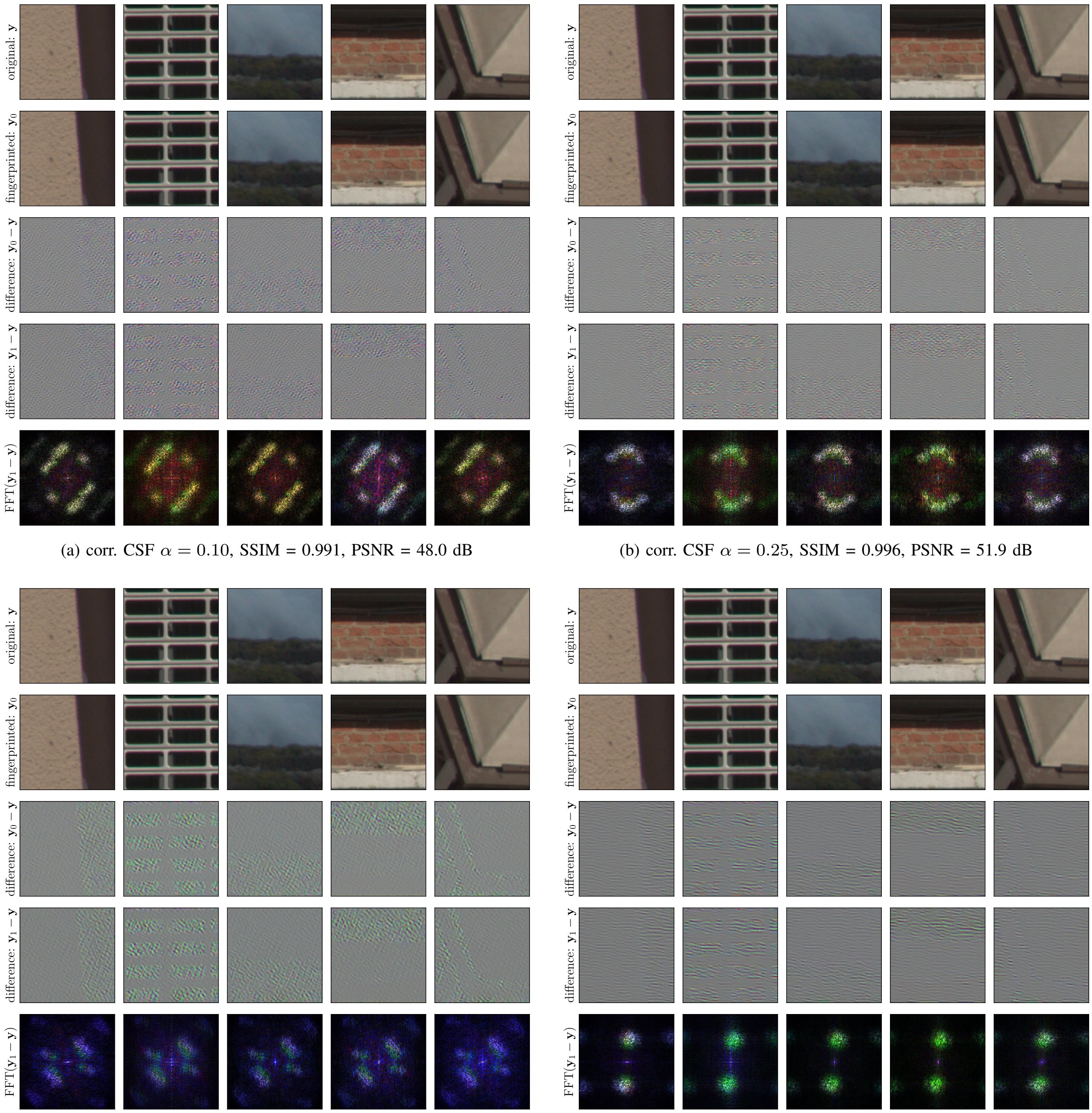


(a) sensor

(b) correlation-based detector

(c) end-to-end detector

Fig. C.1. NN architectures of the sensor and detector. Trainable layers are shown in color: convolutional (blue) and fully connected (violet).



(a) corr. CSF $\alpha = 0.10$, SSIM = 0.991, PSNR = 48.0 dB

(b) corr. CSF $\alpha = 0.25$, SSIM = 0.996, PSNR = 51.9 dB

(c) corr. CSF $\alpha = 0.50$, SSIM = 0.998, PSNR = 55.0 dB

(d) e2e CSF $\alpha = 1.0$, SSIM = 0.999, PSNR = 57.1 dB

Fig. C.2. Example images with an embedded fingerprint (2nd row) and isolated embedding distortion. Rows 3-4 compare the distortion for two random fingerprints. Successive columns compare changes with image content. The fingerprint remains imperceptible despite delivering good detection performance. Each subplot (a-d) corresponds to a different CSF system (correlation or end-to-end) trained at a various quality levels (α).

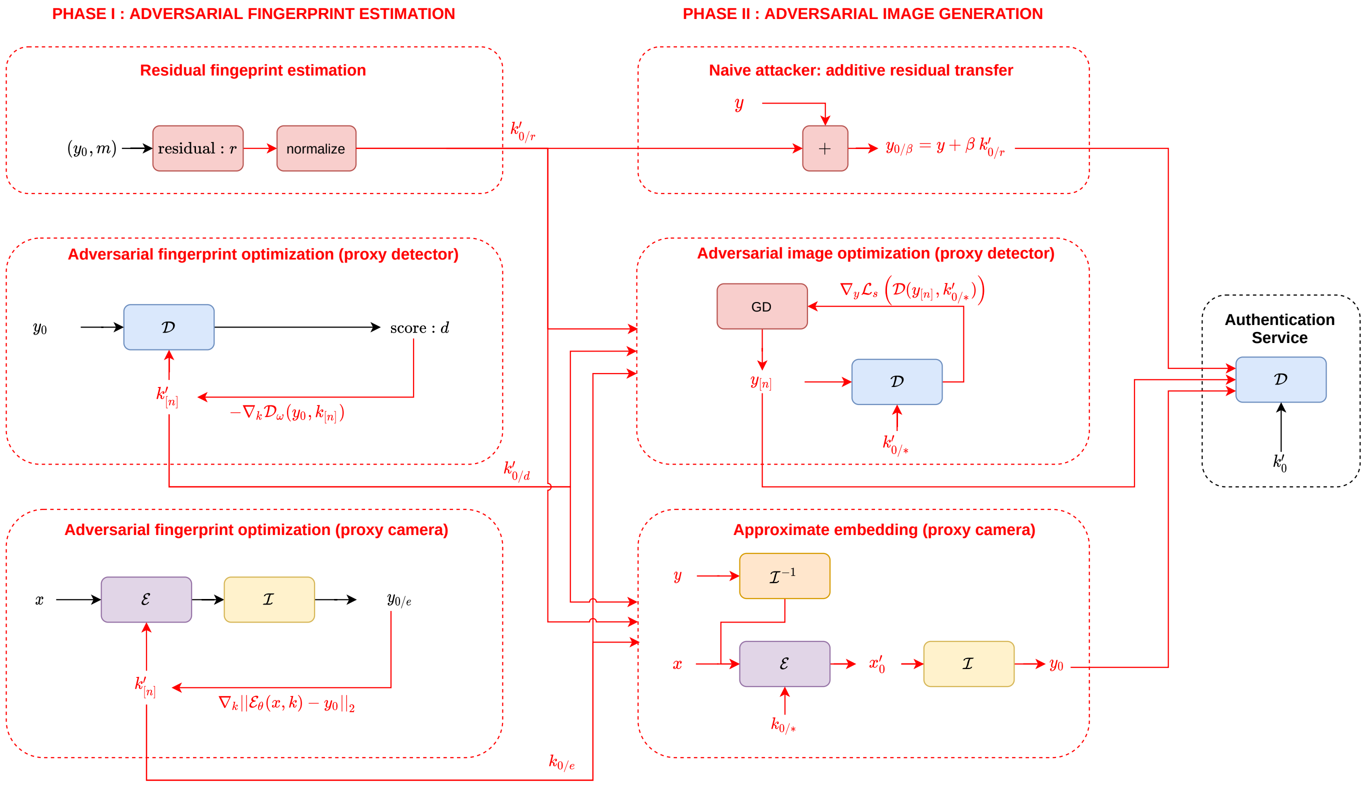

PHASE I : ADVERSARIAL FINGERPRINT ESTIMATION
PHASE II : ADVERSARIAL IMAGE GENERATION
Residual fingerprint estimation
(y_0, m)
residual : r
normalize
$k'_{0/r}$
Naive attacker: additive residual transfer
y
$+$
$y_{0/\beta} = y + \beta\, k'_{0/r}$
Adversarial fingerprint optimization (proxy detector)
y_0
$\mathcal{D}$
score : d
$k'_{[n]}$
$-\nabla_k \mathcal{D}_\omega(y_0, k_{[n]})$
$k'_{0/d}$
Adversarial image optimization (proxy detector)
GD
$\nabla_y \mathcal{L}_s\left(\mathcal{D}(y_{[n]}, k'_{0/*})\right)$
$y_{[n]}$
$\mathcal{D}$
$k'_{0/*}$
Authentication Service
$\mathcal{D}$
k'_0
Adversarial fingerprint optimization (proxy camera)
x
$\mathcal{E}$
$\mathcal{I}$
$y_{0/e}$
$k'_{[n]}$
$\nabla_k ||\mathcal{E}_\theta(x, k) - y_0||_2$
$k_{0/e}$
Approximate embedding (proxy camera)
y
$\mathcal{I}^{-1}$
x
$\mathcal{E}$
x'_0
$\mathcal{I}$
y_0
$k_{0/*}$

Fig. C.3. Illustration of various attack vectors and threat models for the CSF system: an exposed *authentication service* has exclusive access to a secret fingerprint and can be fed adversarial images as input. Adversarial images can be generated (*Phase II*) using various techniques (residual transfer, gradient descent or approximate embedding). The process typically requires an estimate of the secret fingerprint, which can be obtained (*Phase I*) from image residuals, gradients of the detector or gradients of the encoder and the camera ISP. Variants available to the attacker depend on the assumed threat model.