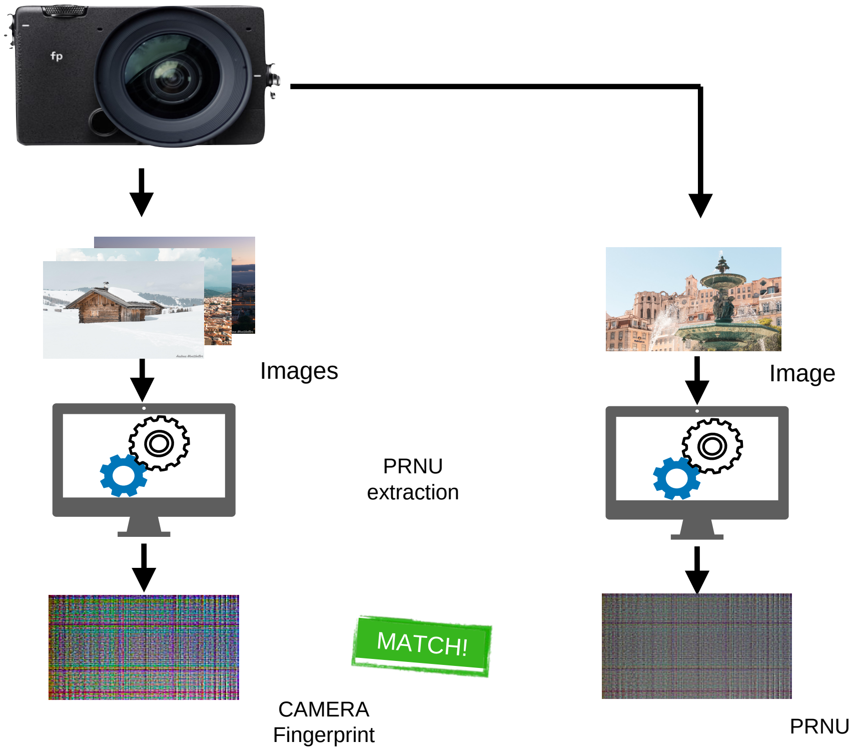
Photo Response non-Uniformity

Andrea Montibeller

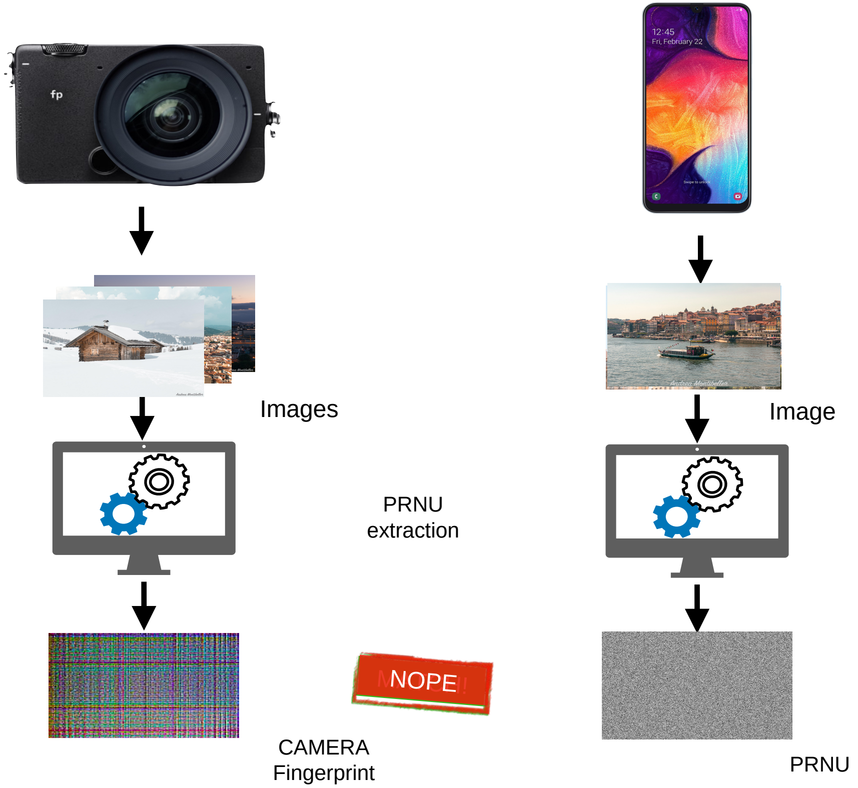
UNIVERSITY OF TRENTO

Photo Response non-Uniformity (PRNU)



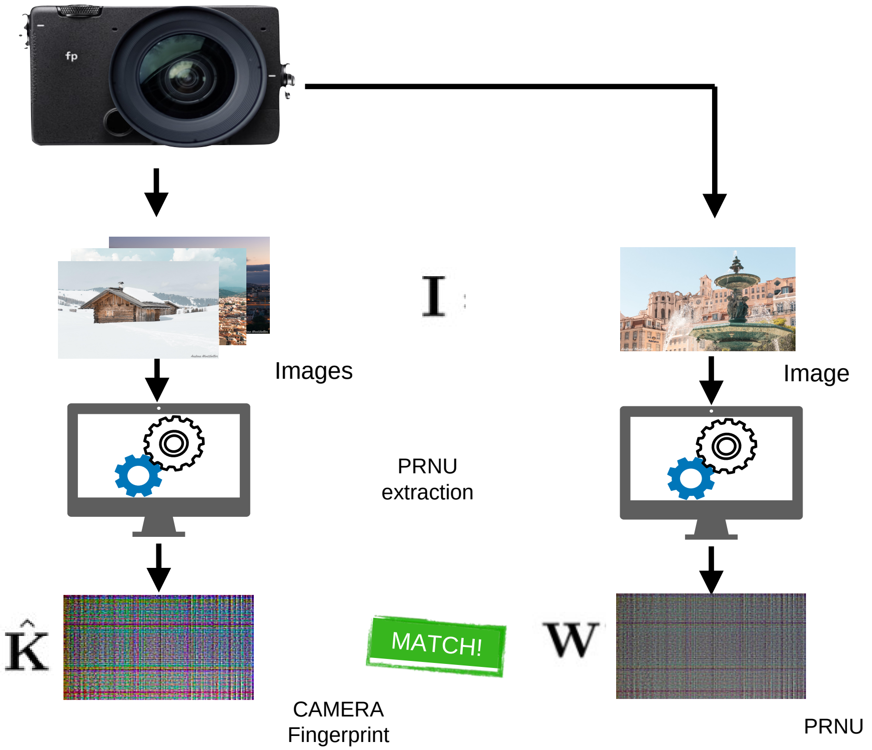
- Invisible and Scant Residual
- Origin: Camera Sensor Manufacturing process
- Used for Camera Source Verification/Attribution
- Also for Tampering Detection

Photo Response non-Uniformity (PRNU)



- Invisible and Scant Residual
- Origin: Camera Sensor Manufacturing process
- Used for Camera Source Verification/Attribution
- Also for Tampering Detection

Photo Response non-Uniformity (PRNU)



$$\mathbf{I} = \mathbf{I}_0 + \mathbf{I}_0 \circ \mathbf{K} + \mathbf{\Theta}$$

- Noise Residual (1 PRNU)

$$\mathbf{W}^{(l)} = \mathbf{I}^{(l)} - F(\mathbf{I}^{(l)}), \quad l = 1, \cdots, L.$$

- Camera Fingerprint (L PRNUs)

$$\hat{\mathbf{K}} = \left(\sum_{l=1}^{L} \mathbf{I}^{(l)} \circ \mathbf{W}^{(l)} \right) \circ \left(\sum_{l=1}^{L} \mathbf{I}^{(l)} \circ \mathbf{I}^{(l)} \right)^{\circ -1}$$

W Noise Residual;
$\hat{\mathbf{K}}$ Camera Fingerprint
$\mathbf{I}_0$ Noiseless Image
$\mathbf{\Theta}$ Random Noise

Peak of Correlation Energy (PCE)

- Used with the PRNU in Source Verification Problems
- PCE > threshold = **Source Identify!**

$$\mathrm{PCE}(\mathbf{W},\mathbf{K}) = \frac{\rho(\mathbf{s}_{peak};\mathbf{W},\mathbf{K})^2}{\frac{1}{mn-\mathcal{N}}\sum_{\mathbf{s}\in\mathcal{I}-\mathcal{N}}\rho(\mathbf{s};\mathbf{W},\mathbf{K})^2} \tag{1}$$

$$\rho(s_1,s_2;\mathbf{W},\mathbf{K}) = \frac{(\sum_{k=1}^{m}\sum_{l=1}^{n}\sum \mathbf{W}[k,l]-\bar{\mathbf{W}})\cdot(\mathbf{K}[k+s_1,l+s_2]-\bar{\mathbf{K}})}{||(\mathbf{W}-\bar{\mathbf{W}})||\,||(\mathbf{K}-\bar{\mathbf{K}})||} \tag{2}$$

$\mathbf{W}$: PRNU of the image or frame (also called noise residual)

$\mathbf{K}$: Camera Fingerprint

$\rho(s_1,s_2;\mathbf{W},\mathbf{K})$: normalized cross-correlation

mn : multiplication between the image (or frame) dimension

$\mathbf{s}_{peak}$: coordinates of the peak of ρ

$\mathbf{s}\in\mathcal{I}-\mathcal{N}$: all the pixels except a small around the peak of ρ

$\mathcal{I}$: the lattice of image/frame coordinates

$\mathcal{N}$: a square region 11×11 around $\mathbf{s}_{peak}$

PRNU Problems

Very sensitive to any post-processing:

- Resizing
- Radial Correction
- Gamma Correction
- Video Stabilization
- HDR Correction
- etc.

Reasons:

- everything that directly affects the image pixels also affect the PRNU

Solutions?

- We need to **reverse** the **post-processing** apply to an image to **retrieve the PRNU reliability**
- It could be time consuming
- Proper optimization required
- Example: https://youtu.be/nz5QYyAW_hY
- or…

Noiseprint

Andrea Montibeller

UNIVERSITY OF TRENTO

What is Noiseprint?

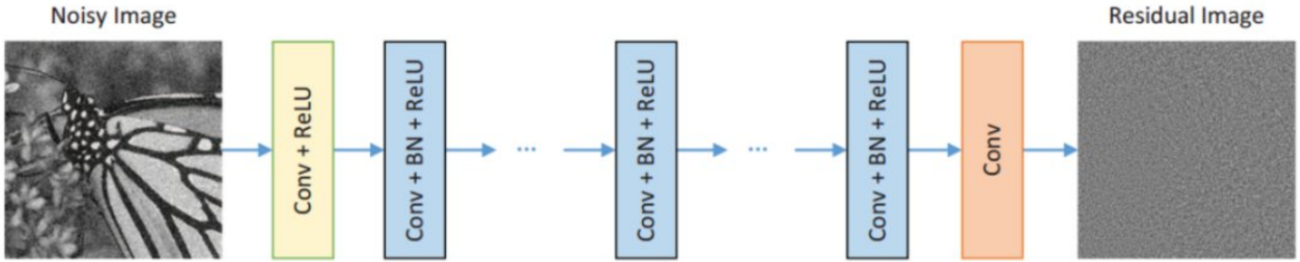
- Deep Learning Based Camera Fingerprint Estimator
- Spatial Transformation Insensitive
- Very effective for tampering detection
- Promising Results but less effective for Device Identification

The Intuition

- PRNU is a noise residual
- PRNU has many problems due to spatial transformation
- PRNU has this problem because relies on signal processing

What if we use deep learning to estimate something like the PRNU?

Let’s begin with DnCNN, the gaussian denoiser of Zhang et. al.



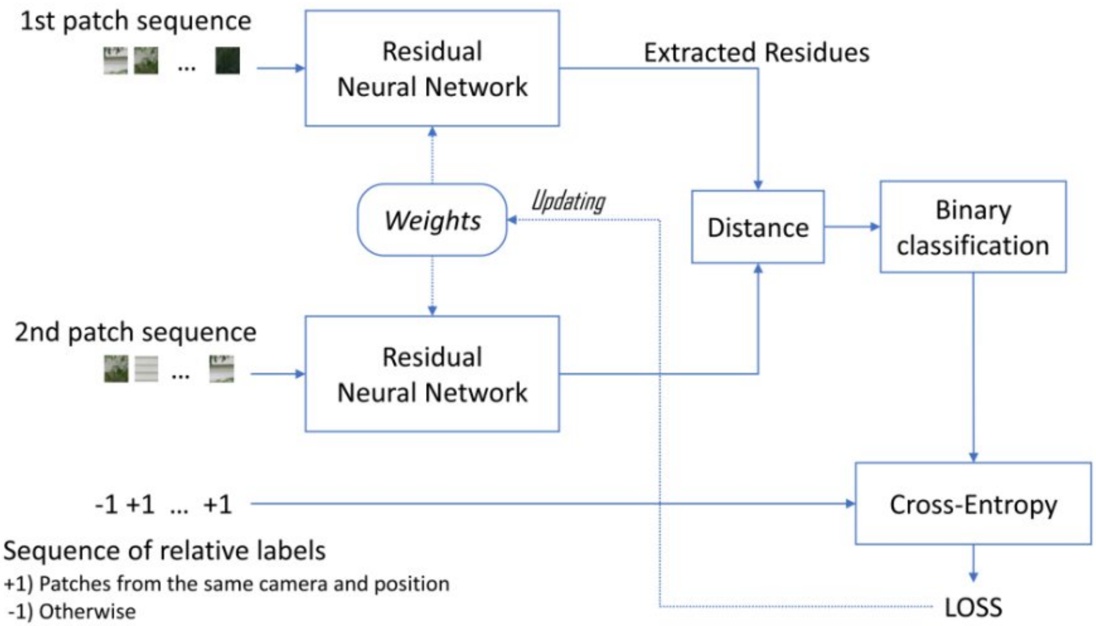
DnCNN: additional information

The GOAL:

Groundtruth | Noisy | Denoised

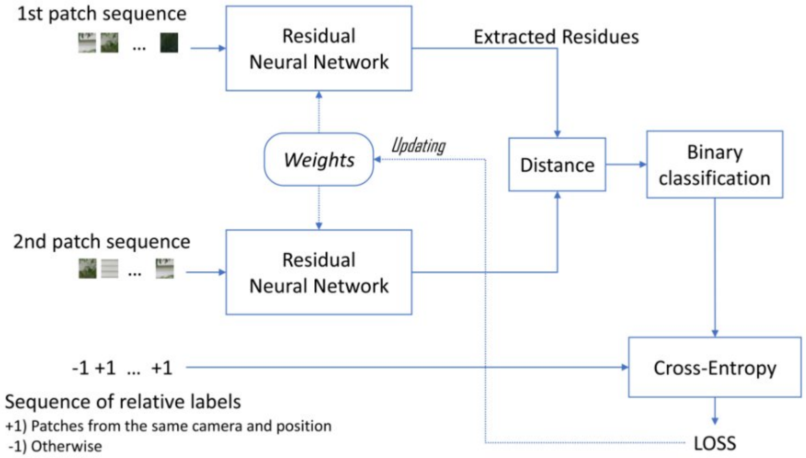
Noiseprint: Implementation

- Batch of 200 patches of size 48x48
- Each batch is divided in 50 groups of 4 patches.
- The patches of each group comes from the same camera model and are cropped with same spatial coordinates
- Goal of the training: capture unique features belonging to the camera models



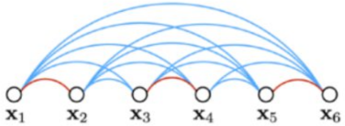
Cozzolino, Davide, and Luisa Verdoliva. "Noiseprint: A CNN-based camera model fingerprint." IEEE Transactions on Information Forensics and Security 15 (2019): 144-159.
https://github.com/grip-unina/noiseprint

Noiseprint: Implementation



1) DISTANCES

$$d_{ij} = \|r_i - r_j\|^2$$



2) SOFTMAX

$$p_i(j) = \frac{e^{-d_{ij}}}{\sum_{j \neq i} e^{-d_{ij}}}$$

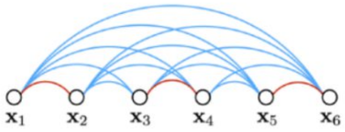
3) LOSS

$$\mathfrak{L} = \sum_i [-\log \sum_{j:l_{ij}=+1} p_i(j)] - \lambda R$$

Noiseprint: Additional Information

1) DISTANCES

$$d_{ij} = \|r_i - r_j\|^2$$



2) SOFTMAX

$$p_i(j) = \frac{e^{-d_{ij}}}{\sum_{j \neq i} e^{-d_{ij}}}$$

3) LOSS

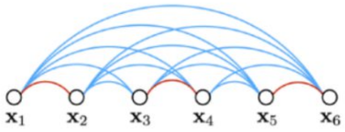
$$\mathfrak{L} = \sum_i \Big[-\log \sum_{j:l_{ij}=+1} p_i(j)\Big] - \lambda R$$

- The red lines refers to the distances computed between the residuals of patches of the same group
- The blue lines refers to the distances computed between the residuals of patches NOT of the same group
- The “red lines” distances will have labels equal to 1
- The “blue lines” distances labels equals to 0

Noiseprint: Additional Information

1) DISTANCES

$$d_{ij} = \|r_i - r_j\|^2$$



2) SOFTMAX

$$p_i(j) = \frac{e^{-d_{ij}}}{\sum_{j \neq i} e^{-d_{ij}}}$$

3) LOSS

$$\mathfrak{L} = \sum_i \Big[-\log \sum_{j: l_{ij}=+1} p_i(j)\Big] - \lambda R$$

- Thus, the goal of the training is to make the distances SMALL when computed between patches of the same group (red lines)

- Large otherwise (blue lines)

- Finally, by using both the “red lines” and “blue lines” distance information and their relative labels, each batch provides examples rather than just , significantly speeding up the training.

Noiseprint: Device Identification Performance

- Results on 6 Devices of 3 Camera Models (Nikon D70, Nikon D200 and Smartphone OnePlus)
- Image Size 1024x1024

	A1	A2	B1	B2	C1	C2
A1	42	8	0	0	0	0
A2	33	17	0	0	0	0
B1	0	0	44	6	0	0
B2	0	0	1	49	0	0
C1	0	0	0	0	33	17
C2	0	0	0	0	9	41

Noiseprint

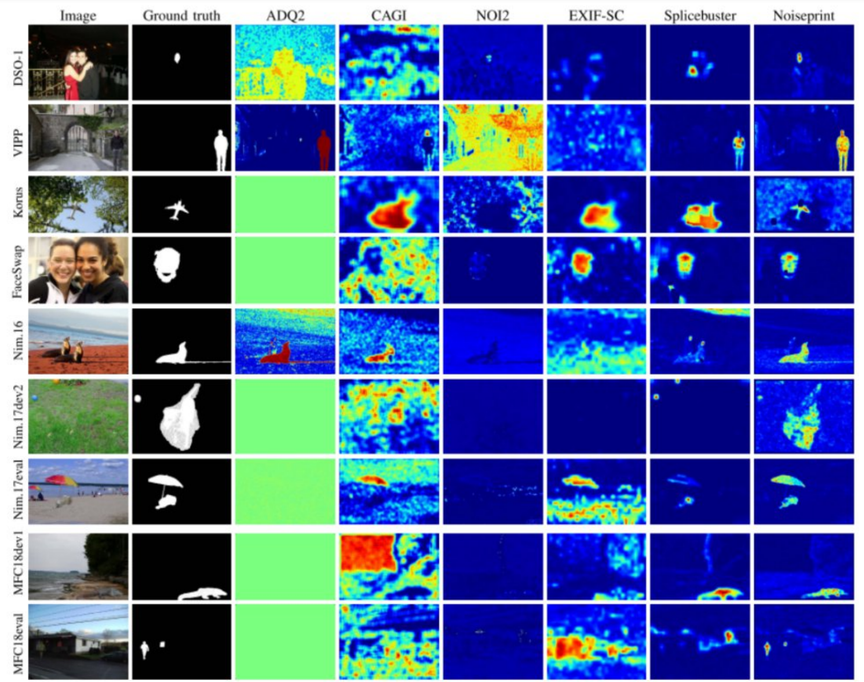
	A1	A2	B1	B2	C1	C2
A1	49	0	0	0	1	0
A2	1	49	0	0	0	0
B1	0	0	50	0	0	0
B2	0	0	0	50	0	0
C1	1	1	2	3	39	4
C2	4	3	3	2	1	37

PRNU

- This PRNU version does not apply any spatial transformation inversion
- Noiseprint capture some camera identification features
- Still, with some device of the same model we have bigger confusion w.r.t. PRNU

Noiseprint: Tampering Detection

- A Real Game Changer



Noiseprint: Tampering Detection

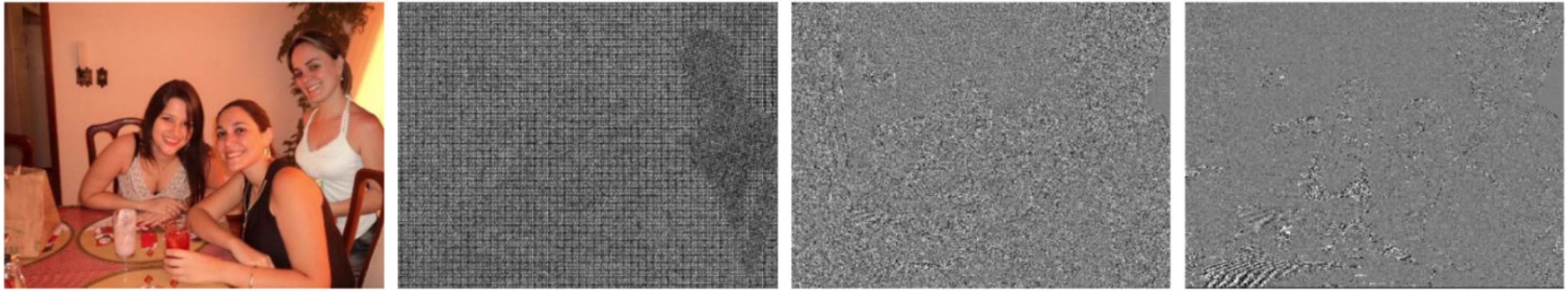


Fig. 2. From left to right: the forged image, its noiseprint, the noise residual obtained using a Wavelet-based denoising filter [27] (a tool commonly used for PRNU extraction) and the noise residual obtained through a 3rd order derivative filter (used in the Splicebuster algorithm [26]).

- Noiseprint can detect both deep based and not deep based forgeries
- This because conventional methods does not “see” the non-continuous artifacts of deep-based post processing methods.
- However, being a deep learning based solution, Noiseprint cannot be used as source of proof in courtroom cases.

[26] Cozzolino, Davide, Giovanni Poggi, and Luisa Verdoliva. "Splicebuster: A new blind image splicing detector." 2015 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, 2015.
[27] Mihcak, M. Kivanc, Igor Kozintsev, and Kannan Ramchandran. "Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising." 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258). Vol. 6. IEEE, 1999.