Camera Identification task using VDNet and VDID

Elaborazione e Protezione delle Immagini

Abdullah Chaudhry

Matricola: 7042261
Department of Information Engineering
University of Florence
Via di Santa Marta 3, Florence, Italy
abdullah.chaudhry@stud.unifi.it

Simone Pezzulla

Matricola: 7072831
Department of Information Engineering
University of Florence
Via di Santa Marta 3, Florence, Italy
simone.pezzulla@stud.unifi.it

***Abstract*—In this report, different algorithm are used in order to perform digital camera identification task. We compare the original implementation [2] and try to replace the original noise extractor algorithm with two Bayesian frameworks. The comparison is taken using VISION dataset[7]. In particular we used natural captured images from sensor and social network processed images.**

***Availability and implementation*—Example code is available at https://github.com/DAFARE.**

***Keywords*—PRNU, denoiser, VDID, VDNet, identification**

I. Introduction

As explained in [5], digital images and video continue to replace their analog counterparts, the importance of reliable, inexpensive and fast identification of digital image origin will only increase. Reliable identification of the device used to acquire a particular digital image would especially prove useful, for example, in the court for establishing the origin of images presented as evidence.

The problem of digital image authentication was addressed with different approaches from the simpler one that involves studying the information contained in the image header and any attachments to more complex solutions such watermarks inside the image itself [3][1] which contains information about the camera. Unfortunately, only relatively expensive cameras (DSLR) support this solution. In [8] they proposed supervised learning method where each image is represented using a vector of numerical features extracted from it. A multiclass support vector machine (SVM) classifier is then trained to classify images from five different cameras. The correct classification obtained 95% in the best case. Forensic applications in the court are likely to require a substantially higher level of accuracy.

Finally, the camera identification method based on a sensor's pattern noise implemented in [2] gives significantly more reliable results compared to state-of-the-art approaches. In this report we propose two new methods and compare them with the original.

In the next Section, we give a description of the 2 methods proposed in this report for noise extraction in order to perform a camera identification task. In particular, in Section III, we focus on the properties of VDNet[10]. We repeat the same process for VDID[4] in Section IV. We briefly discuss in Section V about the dataset we used. In Section VI we discuss the implementation details adopted and show the results in section VI-B. Finally, the report concludes in section VII with some remarks.

II. Research activity

This project is based on the work of [2], the goal is to extract sensor's fingerprint from the captured image and compare this reference with other query fingerprints in order to perform a camera identification task. As suggested in [2], they use Photo-Response Non-Uniformity (PRNU) extracted from flat and not saturated images as a unique fingerprint of digital camera and use Peak-correlation-to-correlation-ratio (PCE) for the identification task. We use different algorithms in order to extract and compare digital camera fingerprints. In this report we extend the implementation provided in [2] by adding 2 new noise extraction methods discussed in the following sections. Later we will also show results on images processed from Facebook and Whatsapp.

III. VDNet

VDNet uses a variational inference for non-iid real-noise estimation and image denoising in a unique Bayesian Network. Specifically, an approximate posterior, parameterized by deep neural networks, is presented by taking the intrinsic clean image and noise variances as latent variables conditioned on the input noisy image.

A. Objective function

Given the clean latent image z_j, the noise level σ_j, the noisy image y_j, the objective function is defined as follows:

$$\min_{W_D, W_S} -\sum_{j=1}^{n} \mathcal{L}(z_j, \sigma_j^2; y_j)$$

where $\mathcal{L}(z_j, \sigma_j^2; y_j)$ can be expressend in functions of μ, m^2, α, β, i.e. the parameters that approximate the posterior of z_j and σ_j.

B. Network Architecture

The Network is composed by two parts:

- D-Net: a U-Net [9] with depth 4, which contains 4 encoder blocks, 3 decoder blocks, and symmetric skip connection under each scale in order to capture multi-scale information of the image
- S-Net: a DnCNN [6] architecture with five layers, and the feature channels of each layer is set as 64.

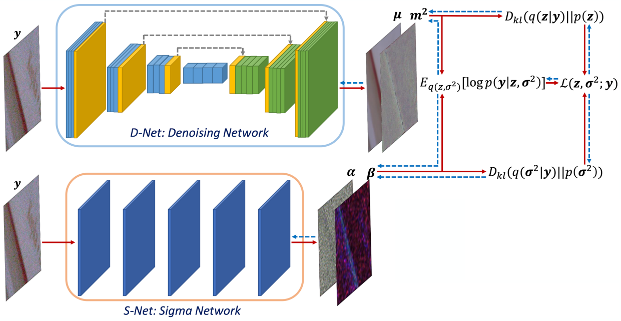

y
D-Net: Denoising Network
y
S-Net: Sigma Network
μ m²
$D_{kl}(q(\boldsymbol{z}|\boldsymbol{y})||p(\boldsymbol{z}))$
$E_{q(z,\sigma^2)}[\log p(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{\sigma}^2)]$
$\mathcal{L}(\boldsymbol{z},\boldsymbol{\sigma}^2;\boldsymbol{y})$
α β
$D_{kl}(q(\boldsymbol{\sigma}^2|\boldsymbol{y})||p(\boldsymbol{\sigma}^2))$

Figure 1: The overall architecture of VDNet

IV. VDID

Variational Deep Image Denoise (VDID) is a bayesian framework that can handle blind scenarios based on the variational approximation of objective functions separating the complicated problem into simpler ones. Its main characteristics are:

- It handles **both AWGN and real-world noise**
- Trained in an end-to-end scheme without any additional noise information
- Requires fewer parameters than state-of-the-art denoisers

The objective is formulated in terms of **maximum a posterior** (MAP) inference. An approximated form of the objective is calculated by introducing a latent variable based on variational Bayes which incorporates the underlying noisy image distribution. Based on the latent space, VDID can focus on simpler subdistributions of the original problem.

A. VDID Architecture

We now discuss about VDID architecture. The Denoiser is fully convolutional neural network adopting ResBlock as the basic building block. The last convolution layer infers the residual image (noise). The Autoencoder is a feedforward convolutional network with a reparametrization trick.

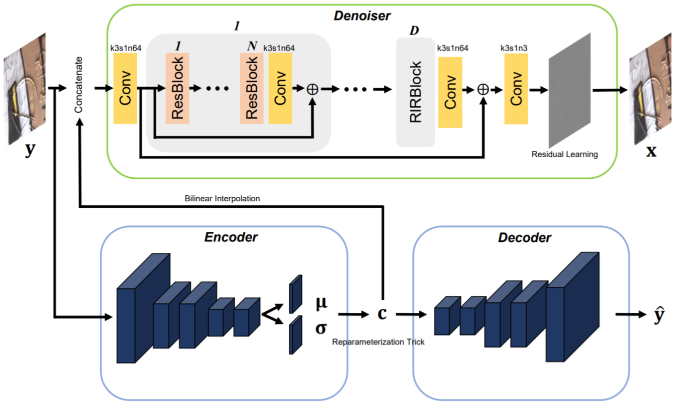


Figure 2: VDID architecture

B. VDID loss

The loss is composed by 3 terms:

1) **Mean absolute error** (MAE) between the ground-truth clean image and the inferred output to minimize the distortion

$$\mathcal{L}_{denoise} = ||x - \hat{x}||_1$$

2) The **KL divergence** between the posterior and the prior. The latent variable c contains an abstract of the noisy image distribution

$$D_{KL}(q(c|y)||p(c))$$

3) The third term is composed by
   - **MAE** between the **noisy input image** and the **decoder's output**
   - **Adversarial loss** [**advloss**]: to better learn the noisy image distribution
   - **Noise estimation loss**: σ_N is the noise level and EST$(\cdot)$ is a simple two-layer CNN.

$$\mathcal{L}_{recon} = ||y - \hat{y}||_1 + \lambda_1 \mathcal{L}_{adv} + \lambda_2 ||\sigma_N - EST(c)||_1$$

The overall loss is the sum of three terms aforementioned. Where the first term is just a naive approach totally relying on the discriminative power of CNN. In the second term the KL divergence forces c to have discriminative power over observed noisy images y. Finally, the third term forces c to include the information on a noisy image.

V. Dataset

For the experiments we used the VISION dataset provided by LESC laboratory [7]. It contains about 30 devices and for each device there are 50 flat and not saturated images from which the reference fingerprints are extracted. Each of them is then compared with 20 natural image query fingerprints (Figure 3 shows the subdirectory structure for a device).

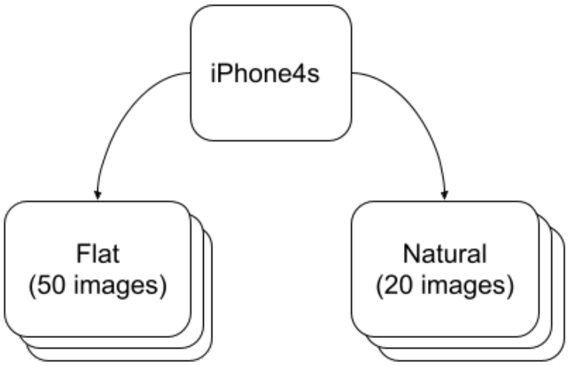


Figure 3: iPhone 4s subdirectory example

VI. Experiments

For implementation settings we only used horizontal images and discarded the vertical ones. For each device we chose randomly 50 flat images and 20 natural images. The experiments were conducted on 9 different devices: iPhone 4, iPhone 4s, iPhone 5c, Samsung Galaxy S3, Samsung Galaxy Tab 3, Samsung Galaxy S3, Samsung Galaxy S5, Wiko Ridge 4G and OnePlus A3000.
Later, we also compared 20 pictures for each device processed through Facebook and Whatsapp.

A. Implementation details

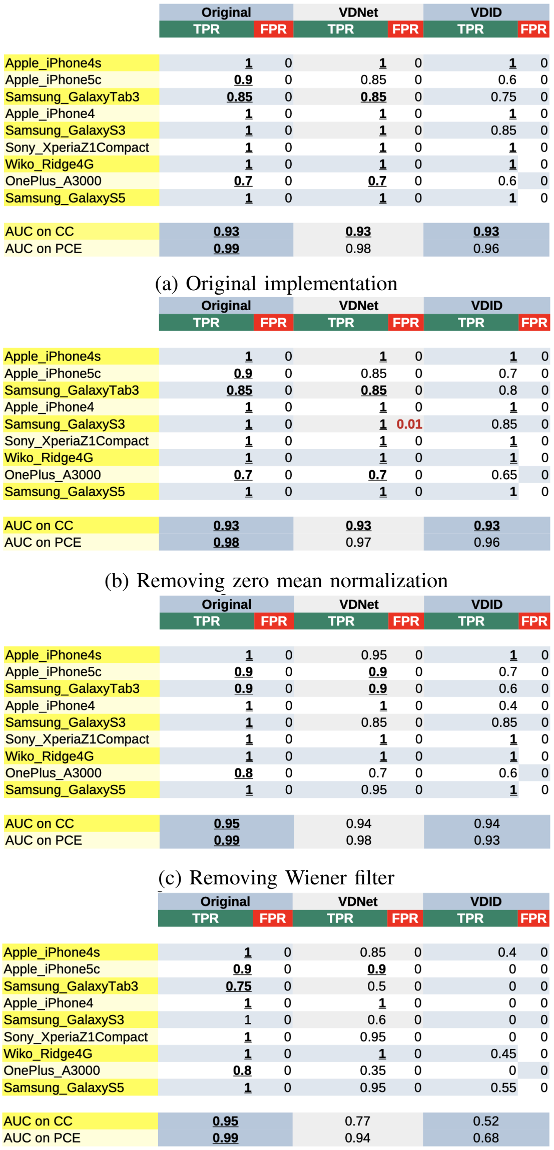
The experiments were carried out in four different configurations. In particular, the first configuration is unaltered compared to the original implementation, while the following three are slight modifications after the noise extraction:

1) Original implementation
2) Removing zero mean normalization from the original implementation
3) Removing Wiener filter from the original implementation
4) Removing both zero mean normalization and Wiener filter from the original implementation

For each strategy we compared the results obtained with original noise extraction implementation with those obtained using VDNet and VDID as noise extractor.
We measured PCR, TPR and FPR scores in order to evaluate the goodness of each solution.

B. Results

Analyzing natural images, as shown in Figure 4, the original method remains the most robust in all the different configurations. Regarding the additional implementation, VDNet has also good performances, specially in the 1st configuration (Figure 4a). The VDID, on the other hand, obtains rather poor results in all the strategies adopted.



	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	**1**	0	**1**	0	**1**	0
Apple_iPhone5c	**0.9**	0	0.85	0	0.6	0
Samsung_GalaxyTab3	**0.85**	0	**0.85**	0	0.75	0
Apple_iPhone4	**1**	0	**1**	0	**1**	0
Samsung_GalaxyS3	**1**	0	**1**	0	0.85	0
Sony_XperiaZ1Compact	**1**	0	**1**	0	**1**	0
Wiko_Ridge4G	**1**	0	**1**	0	**1**	0
OnePlus_A3000	**0.7**	0	**0.7**	0	0.6	0
Samsung_GalaxyS5	**1**	0	**1**	0	**1**	0
AUC on CC	**0.93**		**0.93**		**0.93**	
AUC on PCE	**0.99**		0.98		0.96	

(a) Original implementation

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	**1**	0	**1**	0	**1**	0
Apple_iPhone5c	**0.9**	0	0.85	0	0.7	0
Samsung_GalaxyTab3	**0.85**	0	**0.85**	0	0.8	0
Apple_iPhone4	**1**	0	**1**	0	**1**	0
Samsung_GalaxyS3	**1**	0	**1**	0.01	0.85	0
Sony_XperiaZ1Compact	**1**	0	**1**	0	**1**	0
Wiko_Ridge4G	**1**	0	**1**	0	**1**	0
OnePlus_A3000	**0.7**	0	**0.7**	0	0.65	0
Samsung_GalaxyS5	**1**	0	**1**	0	**1**	0
AUC on CC	**0.93**		**0.93**		**0.93**	
AUC on PCE	**0.98**		0.97		0.96	

(b) Removing zero mean normalization

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	**1**	0	0.95	0	**1**	0
Apple_iPhone5c	**0.9**	0	**0.9**	0	0.7	0
Samsung_GalaxyTab3	**0.9**	0	**0.9**	0	0.6	0
Apple_iPhone4	**1**	0	**1**	0	0.4	0
Samsung_GalaxyS3	**1**	0	0.85	0	0.85	0
Sony_XperiaZ1Compact	**1**	0	**1**	0	**1**	0
Wiko_Ridge4G	**1**	0	**1**	0	**1**	0
OnePlus_A3000	**0.8**	0	0.7	0	0.6	0
Samsung_GalaxyS5	**1**	0	0.95	0	**1**	0
AUC on CC	**0.95**		0.94		0.94	
AUC on PCE	**0.99**		0.98		0.93	

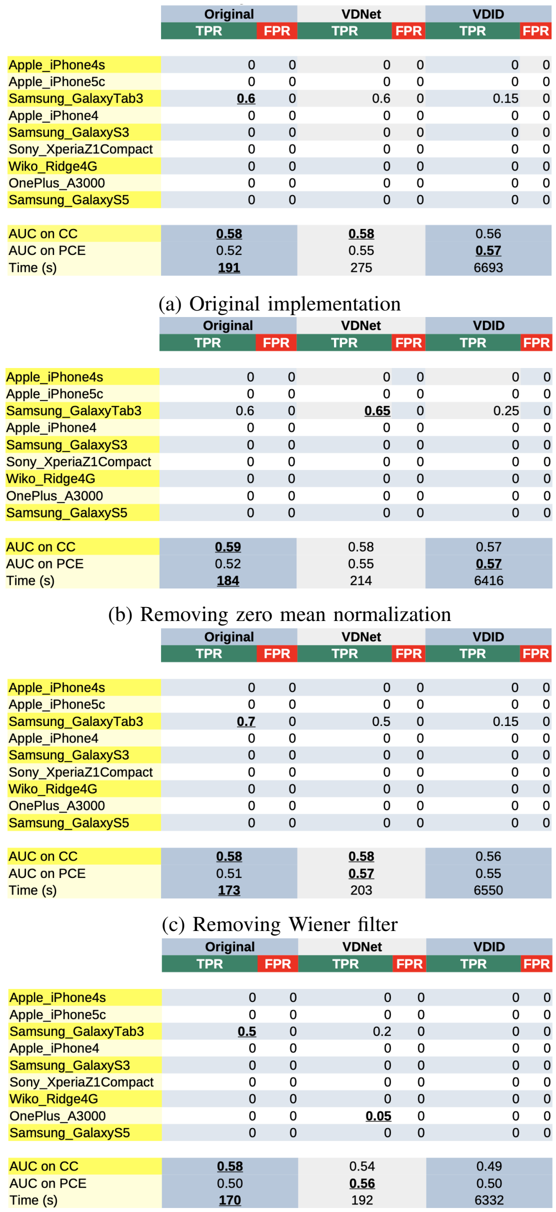
(c) Removing Wiener filter

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	**1**	0	0.85	0	0.4	0
Apple_iPhone5c	**0.9**	0	**0.9**	0	0	0
Samsung_GalaxyTab3	**0.75**	0	0.5	0	0	0
Apple_iPhone4	**1**	0	**1**	0	0	0
Samsung_GalaxyS3	1	0	0.6	0	0	0
Sony_XperiaZ1Compact	**1**	0	0.95	0	0	0
Wiko_Ridge4G	**1**	0	**1**	0	0.45	0
OnePlus_A3000	**0.8**	0	0.35	0	0	0
Samsung_GalaxyS5	**1**	0	0.95	0	0.55	0
AUC on CC	**0.95**		0.77		0.52	
AUC on PCE	**0.99**		0.94		0.68	

(d) Removing both Wiener and zero mean

Figure 4: TPR and FPR comparison using VISION dataset

Removing zero mean normalization (Figure 4b) gives worse results compared to the previous configuration. Only VDID gets slightly better results even though it still remains the worst choice. The best results are obtained in the third configuration (Figure 4c) by the original method removing the Wiener filter. Finally, removing both zero mean normalization and wiener filter in the fourth configuration (Figure 4d) we obtained really poor performances.
Overall we can see that the original method using the third configuration is the most efficient strategy taking TPR values really close to 1 almost for every device. VDNet also achieves similar numerical results to the original implementation. The VDID is a more recent denoiser but it seems to be not a good method for fingerprint extraction in order to perform a camera identification task.



	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	0	0	0	0	0	0
Apple_iPhone5c	0	0	0	0	0	0
Samsung_GalaxyTab3	**0.6**	0	0.6	0	0.15	0
Apple_iPhone4	0	0	0	0	0	0
Samsung_GalaxyS3	0	0	0	0	0	0
Sony_XperiaZ1Compact	0	0	0	0	0	0
Wiko_Ridge4G	0	0	0	0	0	0
OnePlus_A3000	0	0	0	0	0	0
Samsung_GalaxyS5	0	0	0	0	0	0
AUC on CC	**0.58**		**0.58**		0.56	
AUC on PCE	0.52		0.55		**0.57**	
Time (s)	**191**		275		6693	

(a) Original implementation

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	0	0	0	0	0	0
Apple_iPhone5c	0	0	0	0	0	0
Samsung_GalaxyTab3	0.6	0	**0.65**	0	0.25	0
Apple_iPhone4	0	0	0	0	0	0
Samsung_GalaxyS3	0	0	0	0	0	0
Sony_XperiaZ1Compact	0	0	0	0	0	0
Wiko_Ridge4G	0	0	0	0	0	0
OnePlus_A3000	0	0	0	0	0	0
Samsung_GalaxyS5	0	0	0	0	0	0
AUC on CC	**0.59**		0.58		0.57	
AUC on PCE	0.52		0.55		**0.57**	
Time (s)	**184**		214		6416	

(b) Removing zero mean normalization

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	0	0	0	0	0	0
Apple_iPhone5c	0	0	0	0	0	0
Samsung_GalaxyTab3	**0.7**	0	0.5	0	0.15	0
Apple_iPhone4	0	0	0	0	0	0
Samsung_GalaxyS3	0	0	0	0	0	0
Sony_XperiaZ1Compact	0	0	0	0	0	0
Wiko_Ridge4G	0	0	0	0	0	0
OnePlus_A3000	0	0	0	0	0	0
Samsung_GalaxyS5	0	0	0	0	0	0
AUC on CC	**0.58**		**0.58**		0.56	
AUC on PCE	0.51		**0.57**		0.55	
Time (s)	**173**		203		6550	

(c) Removing Wiener filter

	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Apple_iPhone4s	0	0	0	0	0	0
Apple_iPhone5c	0	0	0	0	0	0
Samsung_GalaxyTab3	**0.5**	0	0.2	0	0	0
Apple_iPhone4	0	0	0	0	0	0
Samsung_GalaxyS3	0	0	0	0	0	0
Sony_XperiaZ1Compact	0	0	0	0	0	0
Wiko_Ridge4G	0	0	0	0	0	0
OnePlus_A3000	0	0	**0.05**	0	0	0
Samsung_GalaxyS5	0	0	0	0	0	0
AUC on CC	**0.58**		0.54		0.49	
AUC on PCE	0.50		**0.56**		0.50	
Time (s)	**170**		192		6332	

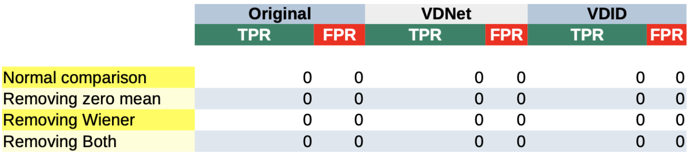
(d) Removing both Wiener and zero mean

Figure 5: TPR and FPR comparison using downloaded images from Facebook

The last experiments we performed consisted in using images processed by social networks such as Facebook and WhatsApp. In this case we got really poor results compared to the previous dataset's images. In Figure 5 we compared results for images processed by Facebook: from all devices only the Samsung Galaxy Tab 3 produced a TPR greater than 0. For this device in particular the original method and VDNet perform quite similarly as before (in the first three configurations). Figure 5b shows a slightly higher TPR in the second configuration.

Finally, WhatsApp processed images gives really poor performances for all devices as shown in Figure 6.

For each configuration we annotated the script run time: the original method and VDNet are executed in few minutes, VDID on the other hand takes a few hours. Comparing the four strategies we can see that removing the Wiener filter which gave the best results can also save a small amount of time.



	Original		VDNet		VDID	
	TPR	FPR	TPR	FPR	TPR	FPR
Normal comparison	0	0	0	0	0	0
Removing zero mean	0	0	0	0	0	0
Removing Wiener	0	0	0	0	0	0
Removing Both	0	0	0	0	0	0

Figure 6: A summary of comparisons using downloaded images from WhatsApp

VII. Conclusion and future developments

In this report we faced the problem of camera identification task from its images based on the sensor's noise pattern. We determined their noise, which serves as a unique identification fingerprint, using three different strategies. Our work is an extension of [2], we implemented VDNet and VDID to extract fingerprints from captured images and compare the results with the original method. We have tried 4 different strategies and found that the removing Wiener filter can improve performances in terms of TPR, PCR and execution time. VDID, on the other hand, performs poorly despite being a better denoiser than VDNet.

The removal of both zero mean normalization and Wiener filter produces worse results in many cases and it doesn't take much better time benefits.

Finally, all methods fail the analysis of images processed by social networks. These methods unfortunately don't work for some newest devices and social network uploaded images. This is due to the fact that image elaboration algorithms may remove the fingerprint.

According to experiments, VDNet brings good results but it has script runtime and TPR score slightly worse compared to the original method. For future works, given the very fast progress of deep learning, it would be interesting to integrate VDNet (with some changes in terms of loss terms or other hyper-parameters) together with the original method as it provides quite similar results.

References

[1] Z. Geradts et al. ""Methods for identification of images acquired with digital cameras". In: *Enabling Technologies for Law Enforcement and Security* (2001).

[2] *Binghamton University*. http://dde.binghamton.edu/download/camera_fingerprin.

[3] P. Blythe and J. Fridrich. ""Secure digital camera". In: *Proc. Digital Forensic Research Workshop* (2004).

[4] Nam Ik Cho Jae Woong Soh. "Variational Deep Image Denoising". In: *ArXiv e-prints* (2021).

[5] Jessica Fridrich Jan Lukas and Miroslav Goljan. "Digital Camera Identification From Sensor Pattern Noise". In: *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY* (2006).

[6] Y. Chen K. Zhang W. Zuo. "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising". In: *IEEE Transactions on Image Processing* (2017).

[7] *LESC laboratory*. https://lesc.dinfo.unifi.it.

[8] H. T. Sencar M. Kharrazi and N. Memon. """Blind source camera identification". In: *Proc. ICIP* (2004).

[9] Philipp Fischer Olaf Ronneberger and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention* (2015).

[10] Qian Zhao Zongsheng Yue Hongwei Yong. "Variational Denoising Network: Toward Blind Noise Modeling and Removal". In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada* (2019).