POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Noise-based anomaly detection for image forgery localization

Laurea Magistrale in Music and Acoustic Engineering - Ingegneria Musicale e Acustica

Author: Manuel Alejandro Jaramillo Rodríguez

Advisor: Prof. Paolo Bestagini

Co-advisor: Sara Mandelli

Academic year: 2022-2023

1. Introduction

We live in an epoch where multimedia acquisition devices such as cameras and smartphones have become very accessible tools of common use in our day life. Constantly, we are sharing and consuming images, videos and recordings through web platforms such as social media. With this growth on the availability of capturing devices, also a huge amount of digital processing techniques have appeared. Some of these techniques allow us to completely alter the content of the media, giving us a powerful tool for enhancing and personalizing our pictures, videos and audios, but also, opening a door for using them with malicious purposes such as fake publicity or impersonation. For instance, online tools like DALL·E [4] and others let the users insert fake faces on images and videos. For these reasons, multimedia manipulations have become a potential risks in different scenarios of our society.

In this context, multimedia forensics, and more specifically image forensics, have become of great interest in the scientific community. Some of the most successful techniques for spotting local image manipulations are data-driven statistical methods [1]. Data-driven approaches harness the power of deep learning and statistical analysis to uncover patterns and anomalies within images that may indicate that some kind of tampering has been made. Leveraging on large datasets, these methods employ sophisticated algorithms to automatically learn and detect complex characteristics of manipulated images at a pixel level.

One of the state-of-the-art methods for image forgery localization is the Noiseprint [1], which exploits unique traces associated with the camera-model used to acquire photographs. In fact, each camera model possesses its own specifications, like the lenses, sensors and the Color Filter Array (CFA), all of them being characterized by specific features. Additionally, diverse digital processing stages may be incorporated depending on the manufacturer and specific to each camera model, like white balancing or color correction. It is expected that localized manipulations will have an effect on the Noiseprint, leaving local traces on it. By extracting the Noiseprint from an image, the authors of [1] end up with an efficient forgery localization method.

In this work, we aim to construct a robust algorithm capable of detecting and localizing image manipulations. To this purpose, we take inspiration from the idea of the Noiseprint. To do so, we start by training a Neural Network (NN)-based denoiser capable of extracting a camera-model fingerprint from images. Then, we ex-

ploit different activation maps of the denoiser as abstract high-level noise-related features for training an Anomaly Detection (AD) algorithm. This algorithm makes use of the extracted features to generate a heatmap that determines the dissimilarity between the input image and a set of normal images. By doing so, we are able to detect any present local variations on the extracted noise fingerprint, exhibiting possible tampered pixels inside the images.

To validate our results, apart from a public dataset based on splicing [2], we construct two more datasets based on generative Artificial Intelligence (AI) models by applying realistic local manipulations to natural images. Our results show remarkable performance in the proposed datasets, outperforming in most of the cases the state-of-the-art.

2. Problem Formulation

In this work, we focus on developing a robust method that is able to detect which pixels of an image have been manipulated by means of splicing. Formally, we define a generic pristine image as $\mathbf{I}$, with size $H \times W$, in which a pixel coordinate is defined as (h, w), $h \in [1, H]$ and $w \in [1, W]$. The locally manipulated version of $\mathbf{I}$ can be written as

$$\bar{\mathbf{I}}(h,w) = \begin{cases} t, & \text{if } (h,w) \in \mathcal{S}, \\ \mathbf{I}(h,w), & \text{if } (h,w) \notin \mathcal{S} \end{cases} \quad (1)$$

where $\mathcal{S}$ is the pixel region under splicing attack and t corresponds to the specific tampered with pixel value.

The local manipulation can be described by a tampering mask, which is a 2D matrix with the same spatial size of the image. We can define the tampering mask as

$$\mathbf{M}(h,w) = \begin{cases} 0, & \text{if } (h,w) \notin \mathcal{S}, \\ 1, & \text{if } (h,w) \in \mathcal{S} \end{cases} \quad (2)$$

Figure 1 shows a sketch of the tackled problem: given a generic manipulated image $\bar{\mathbf{I}}$, our goal is to estimate a tampering heatmap $\mathbf{H}$ that is as close as possible to the mask $\mathbf{M}$. A heatmap $\mathbf{H}$ is a single-channel image with the same spatial dimensions as the input. For each pixel location (h, w), the value of the pixel $\mathbf{H}(h, w)$ represents the probability of the image pixel belonging to the manipulated class.

Figure 1: We show the pristine image $\mathbf{I}$, its manipulated version $\bar{\mathbf{I}}$ (the splicing region is highlighted by red contour) and the corresponding mask $\mathbf{M}$.

3. Proposed method

The proposed method is designed for detecting forgeries in the case of image splicing. To do so, we take advantage of the unique camera model related artifacts that are present on captured images. The method can be separated into two main stages:

Fingerprint extraction. We use a denoising NN architecture to retrieve low level information from the image. In particular, the denoiser is capable of extracting camera model-based artifacts. If the image has been manipulated by splicing, the denoising model extracts a noise fingerprint which shows two different patterns, clearly making a distinction between non-manipulated and manipulated pixels.

This distinction, however, may or may not be perceptible by the human eye, moreover, even if it is perceptible, it might not be easily separable into the two classes, in the sense that a simple thresholding might not be able to separate the two different noisy patterns.

Heatmap generation. In this stage, an AD technique is applied in order to generate a tampering heatmap that effectively quantifies the differences between the patterns, resulting in a probability map that can thresholded to obtain a mask that classifies each pixel as belonging to one of the extracted patterns. In particular, we make use of the noise fingerprint and of the network activation maps acquired in the previous stage.

Figure 2 reports a sketch of the proposed methodology.

3.1. Fingerprint extractor training

In this stage, we aim to develop a NN-based denoiser that is capable of extracting from images the camera model related artifacts. This should be done by removing all the scene content and other types of noises coming from different sources. To do so, we take inspiration

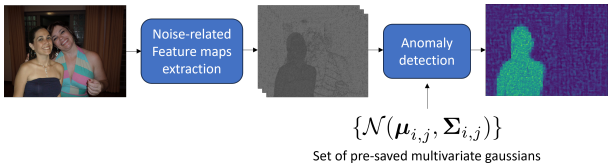

Figure 2: Simplified sketch of the proposed pipeline. From a test image, we extract noise-related feature maps by using a pretrained fingerprint extractor. Finally, we use the feature maps as input of an anomaly detection algorithm to estimate a manipulation heatmap.

from the original Noiseprint paper [1], proposing a Siamese framework for training a denoising network architecture:

1. We select a number of pristine images from which we know the exact camera model that was used for their acquisition.
2. We crop images into squared patches $\mathbf{P}^m_{(h,w)}$ of size $P \times P$, where (h, w) represents the position of the top-left pixel of the patch in the original image, and m the camera model used for capturing the image.
3. Each patch is fed into a denoising network. We extract its related model-related fingerprint, defined as $\mathbf{F}^m_{(h,w)}$.
4. We generate a mini-batch of samples containing the fingerprints of N patches coming from different images and camera models. Then, we pair patches only if they meet the two following conditions: they share the same camera model, and they are extracted from the same pixel positions but from different images.
5. The weights of the denoising NN are iteratively updated by calculating the Distance-Based Logistic (DBL) loss between the fingerprints in the mini-batch. The DBL loss minimizes a distance metric between the fingerprints of paired patches, while maximizing the unpaired ones.

Figure 3 shows a sketch of the training process.

By following this procedure, we are training a denoising NN that is not only able to extract noise fingerprints that are similar for all the images belonging to the same camera model, but also to distinguish between images captured from different models and maximize the dissimilarity between them.

We employ the algorithm described to train two distinct NN models. The first one is a

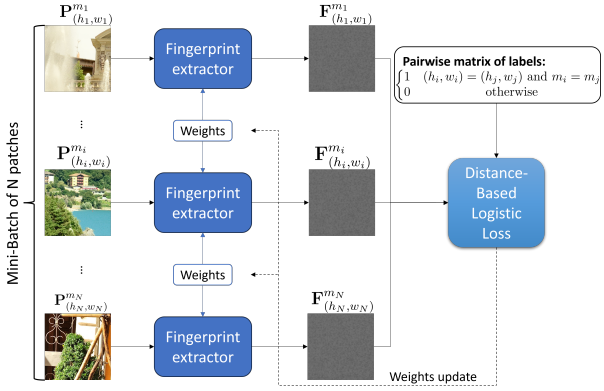

Figure 3: Siamese architecture for training the fingerprint extractors.

Convolutional NN (CNN)-based denoiser called Denoising CNN (DnCNN) [6], which is a fully-convolutional NN used for image denoising. As a second model, we use Restormer [5], which is a Vision Transformer (ViT) architecture used for general image restoration purposes, including denoising.

3.2. Heatmap generator training

For the heatmap generation stage, we take inspiration from the Patch Distribution Modeling (PaDiM) algorithm proposed in [3]. The idea behind PaDiM is to use a CNN for extracting feature maps from a dataset of "normal" images; these maps are used to construct a set of reference local embeddings that follow a specific distribution. Then, a generic image can be tested and compared with the training data distribution to spot local anomalies.

In particular, we propose an Anomaly Detection (AD) algorithm to infer distributions from a set of "normal" training data, which in our case are non-manipulated pristine images. To do so, we exploit denoising NN architectures pretrained as shown in Section 3.1.

3.2.1 Training steps

The training steps of our proposed AD method are as follows (see Figure 4):

1. We select N pristine images coming from M different camera models. Since all these images may have different sizes, we crop them to a common size of $C \times C \times 3$ pixels, starting from the top-left corner of the image. Let us call the set $\{\mathbf{I}^k_c\}$, where $k = 1, ..., N$ enumerates the N images, and c is used to

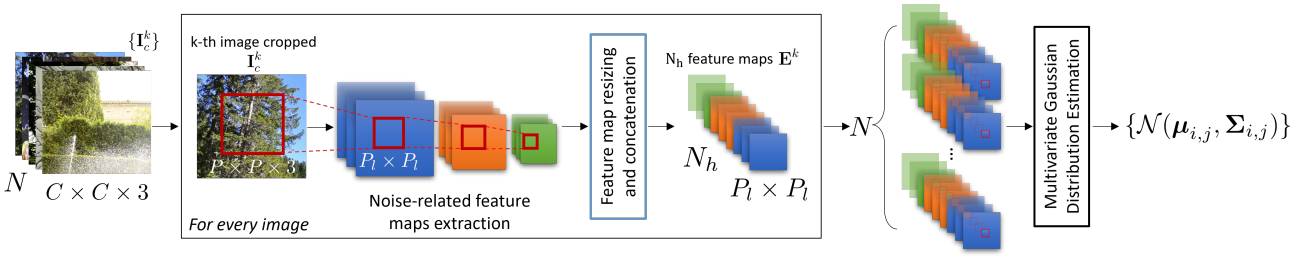

Figure 4: Scheme of the procedure for training the proposed AD algorithm.

clarify that it is a cropped image.

2. The cropped images $\{\mathbf{I}_c^k\}$ are fed into one of the proposed fingerprint extractors. Then, N_h activation maps are selected and hooked from some of the layers of the extractor.
3. From the previously acquired maps, only those pixels corresponding to a patch $\mathbf{P}^k$ of size $P \times P \times 3$ inside every original cropped image $\mathbf{I}_c^k$ are kept, in order to avoid border artifacts. Notice that, in the most general case, feature maps coming from different network layers may have different spatial size. Therefore, every original input patch $\mathbf{P}^k$ has a corresponding activated region in the selected maps which is typically smaller than $P \times P$ pixels, its size changing according to the activation map depth.
4. We upscale all the selected activation regions to have equal spatial dimensions. We end up with all the regions having the same spatial size of $P_l \times P_l$ pixels, were P_l and P_l represents the size of the largest selected activation region.
5. We concatenate the resulting feature maps extracted from all the considered layers. Each $\mathbf{P}^k$ has its own embeddings $\mathbf{E}^k$ with size $N_h \times P_l \times P_l$. Notice that, by following similar considerations to those done in step 3, every $\mathbf{E}_{i,j}^k$, which has N_h total elements and $(i,j) \in [1, ...P_l] \times [1, ...P_l]$, corresponds to a small pixel area in the input patch $\mathbf{P}^k$. Therefore, $\mathbf{E}_{i,j}^k \in \mathbb{R}^{N_h}$ is defined as the embedding vector of a specific pixel region of the input $\mathbf{P}^k$.
6. Considering the contributions of all training images, we end up with a set of N embeddings $\{\mathbf{E}^k\}, k = 1, ..., N$. For each pixel position (i,j) of the embeddings, we estimate a multivariate Gaussian distribution by computing the mean $\boldsymbol{\mu}_{ij}$ and covariance matrix $\boldsymbol{\Sigma}_{ij}$ over the set of N samples.

3.3. Deployment stage

When a query image has to be analyzed, we pass it through the trained fingerprint extractors to extract features by selecting specific network layers. Then, we exploit our proposed AD algorithm to find local anomalies.

To correctly apply the algorithm, we have to make a one-to-one comparison of patch embeddings from test and train images. This means that the spatial size of the input images at test step must be the same as that of the patches used at train step, i.e, $P \times P$. To avoid any resizing operation, we operate in a patch-wise framework.

The procedure for extracting the heatmap from an image of size $H \times W \times 3$ is the following:

1. The image is passed through the fingerprint extractors (DnCNN or Restormer).
2. We select N_h activation maps from the layers of the fingerprint extractors. For the DnCNN we hook the last three network layers, while for the Restormer we only use the estimated fingerprint (last layer).
3. We divide the activation maps into regions, every region corresponding to input image patches of size $P \times P$.
4. We create patches' embedding vectors in the same way they were created at training step.
5. The Mahalanobis distance between each embedding vector and the reference multivariate Gaussian distribution is calculated, resulting in an anomaly score for every pixel position (i,j).
6. The heatmap of the whole image is estimated by properly joining all the obtained score maps from the single patches.

4. Results

In this section, we present the results and performance of our method on the proposed datasets.

Then, we make a comparison with the state-of-the-art techniques. To do so, we rely on three different datasets:

- DSO-1: The DSO-1 dataset [2] comprises 200 images, including indoor and outdoor scenes, with 100 original and 100 manipulated images.
- PNG-based Generative-Based Manipulated Dataset (GBMD): We use DALL·E as a generative AI local tampering method, starting from 50 uncompressed PNG images to generate 200 locally manipulated images.
- JPEG-based GBMD: We generate another dataset based on DALL·E, starting from 50 JPEG compressed images to generate 200 images containing local manipulations.

In all the considered datasets, we do not have 100% control on the applied tampering operations. In the case of DSO-I, we have no information on the devices and camera models used for capturing the images, while on GBMD we have no information on how the image is exactly processed by DALL·E.

We employ the Matthews Correlation Coefficient (MCC) and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) as performance metrics. Both metrics are computed between the estimated heatmap $\mathbf{H}$ and the associated tampering mask $\mathbf{M}$. The higher the metrics (ideally achieving 1), the better the localization results.

4.1. Test on the proposed Datasets

Table 1 reports the results of each of our models on DSO, PNG-based GBMD and JPEG-based GBMD datasets respectively.

Table 1: Results of testing our method on the three proposed datasets. In bold, the best results per dataset.

	DSO		JPEG-based GBMD		PNG-based GBMD	
	DnCNN	Restormer	DnCNN	Restormer	DnCNN	Restormer
AUC	0.951	**0.968**	**0.975**	0.952	**0.834**	0.750
MCC	0.731	**0.843**	0.838	**0.859**	**0.545**	0.520

Both models obtain remarkable results on DSO and JPEG-based GBMD datasets, while on the PNG-based GBMD dataset the performances are slightly lower and seem to be affected by the PNG nature of the original images. In the case of the DSO dataset, which is shared as PNG images, the achieved good results lead us to think that, in some step of the dataset production, some JPEG compression was applied.

Looking deeply at the results on the DSO dataset, we observe better performance coming from the Restormer model, outperforming in a moderate quantity the DnCNN model. The contrary comes out when testing on the PNG-based GBMD dataset, where DnCNN performs slightly better than Restormer. For the JPEG-based GBMD dataset, the overall performances are valid for both the denoising models.

Figures 5 shows three examples of fingerprints and heatmaps estimated with our method for the three considered datasets.

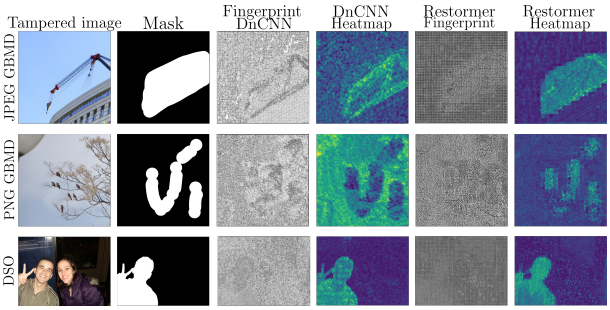

Figure 5: Examples of the results on the proposed datasets.

4.2. Test on post-processed images

We explore how post-processing operations like resizing and JPEG compression applied to the manipulated images can affect the performance.

Table 2 shows the results of our method after JPEG compressing the images. We can notice that compressing the image with the highest possible quality factor does not have a big impact on performances. In fact, if we compare these results with those of table 1, the impact is less than 2% for both models. On the other hand, reducing the JPEG quality factor has an unwanted effect on the results, with this effect being more critical on the Restormer model, even for high quality factors.

Table 2: Results on JPEG compressed images.

	DnCNN			**Restormer**		
QF	100	95	90	100	95	90
AUC	0.943	0.682	0.643	0.959	0.644	0.598
MCC	0.722	0.229	0.178	0.832	0.187	0.135

We report the results of applying 0.8×, 0.9×, 1.1× and 1.2× scaling in Table 3. We can notice

a huge impact in the performance for both models, being the DnCNN the most affected one. It is interesting to observe that Restormer is more successful than DnCNN in case of downscaling, while DnCNN outperforms Restormer in case of upscaling. With these results, we confirm that the camera-model traces that we want to isolate are highly affected when applying rescaling to the images.

Table 3: Results of testing our method on resized images.

	DnCNN				Restormer			
Resizing factor	0.8	0.9	1.1	1.2	0.8	0.9	1.1	1.2
AUC	0.618	0.657	0.696	0.706	0.706	0.714	0.686	0.703
MCC	0.155	0.170	0.221	0.227	0.274	0.279	0.204	0.223

Figures 7 and 6 show results for compressed and resized images, respectively.

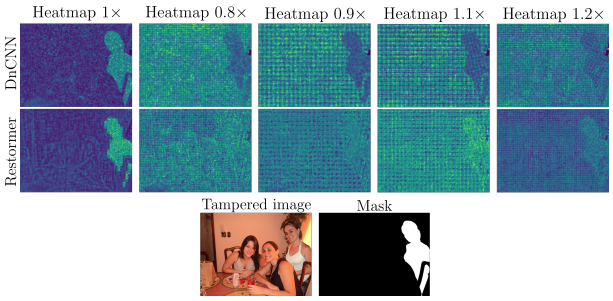

Figure 6: Results on a resized image from the DSO dataset, we show the heatmaps at 0.8×, 0.9×, 1.1× and 1.2×.

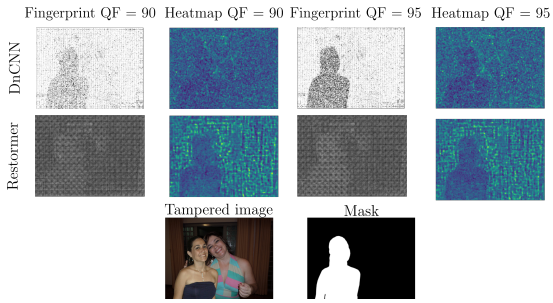

Figure 7: Results on a JPEG compressed image from the DSO dataset, we show the Fingerprints and heatmaps for Quality Factors of 90 and 95.

4.3. Comparison with state-of-the-art

We compare our results with state-of-the-art methods, in particular, with the method from which we take inspiration, the Noiseprint [1]. We make the comparison by taking as a reference our best results per dataset. Results are shown in Table 4.

Table 4: Comparison with the state-of-the-art. For space constraints, D corresponds to DnCNN, R to Restormer, N to Noiseprint. In bold, the best results per dataset.

	DSO			JPEG-based GBMD			PNG-based GBMD		
	D	R	N	D	R	N	D	R	N
AUC	0.951	**0.968**	0.926	**0.975**	0.952	0.961	0.834	0.750	**0.938**
MCC	0.731	**0.843**	0.722	0.838	**0.859**	0.841	0.545	0.520	**0.769**

In case of DSO, we observe a considerable advantage for our method. For the JPEG-based GBMD dataset, both models have almost the same performance. On the PNG-based GBMD, we observe a noticeable out-performance by the Noiseprint. However, this result was expected from the previously analyzed experiments. Indeed, we already noticed that our method has disadvantages when used on images that were not JPEG compressed before the forgery was added.

5. Conclusions

In this work, we faced the problem of image forgery localization, in particular, the cases of image splicing and image manipulation with generative AI technologies. These forgeries can be made with tools that are available to almost any person, which makes the dissemination of manipulated images a problem of great concern.

Inspired by two state-of-the-art algorithms, we proposed a method to expose tampering regions inside images. To do so, we trained denoising NN capable of extracting a camera-model fingerprint from images. Then, we used activation maps coming from different layers of these denoisers to apply an AD procedure, resulting into a heatmap that can be interpreted as a probability map of tampered with pixels.

Our technique shows promising results for all the considered local manipulation techniques. In most of the considered experiments, our method outperforms one of the top state-of-the-art techniques. However, when the forgeries are applied to images that have never been JPEG compressed or when the manipulated images are post-processed, the performances are affected considerably. Future works will be dedicated to further investigations for enhancing the robustness of the proposed method.

References

[1] Davide Cozzolino and Luisa Verdoliva. Noiseprint: A cnn-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15:144–159, 2020.

[2] Tiago José de Carvalho, Christian Riess, Elli Angelopoulou, Hélio Pedrini, and Anderson de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *IEEE Transactions on Information Forensics and Security*, 8(7):1182–1194, 2013.

[3] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: A patch distribution modeling framework for anomaly detection and localization. In *Pattern Recognition. ICPR International Workshops and Challenges*, pages 475–489, Cham, 2021. Springer International Publishing.

[4] Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text. *UGC Care Group I Journal*, 8(14):71–75, 2021.

[5] S. Zamir, A. Arora, S. Khan, M. Hayat, F. Khan, and M. Yang. Restormer: Efficient transformer for high-resolution image restoration. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5718–5729, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[6] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.