Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo XXXII

Settore Concorsuale: 01/B1
Settore Scientifico Disciplinare: INF/01

Classification and Clustering of Shared Images on Social Networks and User Profile Linking

Presentata da:
Rahimeh Rouhi

Supervisore:
Danilo Montesi

Coordinatore Dottorato:
Davide Sangiorgi

Cosupervisore:
Flavio Bertini

Esame finale anno 2020

Dedication

To my parents, the angels with me at anytime and everywhere pushing me forward ... .

Acknowledgments

I would like to express my deepest gratitude to my parents, partner, sisters, brothers in law and nieces for their constant support and love. Regardless of the ups and downs in life, they were always there to help me and stood by my side.

I would like to express my sincere gratitude and utmost respect to my supervisor, Prof. Danilo Montesi, for his tremendous academic support and valuable career advice. I would like to thank him for providing me with a lot of opportunities to shape myself into a research scientist. I am very grateful to my co-supervisor, Dr. Flavio Bertini, for his inspiring guidance and invaluable advice on my Ph.D. progress.

I would appreciate a great help from Prof. Chang-Tsun Li and his supportive team Dr. Xufeng Li and Dr. Yijun Quan, for hosting me during my abroad course at Charles Sturt University, Australia.

I would like to thank my Ph.D. committee Prof. Paola Mello and Prof. Maurizio Gabbrielli for their invaluable time and feedback on my Ph.D. work.

I would also like to thank my lab mates: Dr. Stefano Giovanni Rizzo, Dr. Wilson Evuarherhe Sakpere, Dr. Stefano Pio Zingaro, Dr. Francesco Gavazzo and my friend Dr. Sara KasmaeeYazdi. Completing this work would have been all the more difficult without the support and friendship provided by them.

ABSTRACT

The ever increasing prevalence of smartphones and the popularity of social network platforms have facilitated instant sharing of multimedia content through social networks. However, the ease in taking and sharing photos and videos through social networks also allows privacy-intrusive and illegal content to be widely distributed. As such, images captured and shared by users on their profiles are considered as significant digital evidence for social network data analysis.

The Sensor Pattern Noise (SPN) caused by camera sensor imperfections during the manufacturing process mainly consists of the Photo-Response Non-Uniformity (PRNU) noise that can be extracted from taken images without hacking the device. It has been proven to be an effective and robust device fingerprint that can be used for different important digital image forensic tasks, such as image forgery detection, source device identification and device linking. Particularly, by fingerprinting the camera sources captured a set of shared images on social networks, User Profile Linking (UPL) can be performed on social network platforms.

The aim of this thesis is to present effective and robust methods and algorithms for better fulfilling shared image analysis based on SPN. We propose clustering and classification based methods to achieve Smartphone Identification (SI) and UPL tasks, given a set of images captured by a known number of smartphones and shared on a set of known user profiles. The important outcome of the proposed methods is UPL across different social networks where the clustered images from one social network are applied to fingerprint the related smartphones and link user profiles on the other social network. Also, we propose two methods for large-scale image clustering of different types of the shared images by users, without prior knowledge about the types and number of the smartphones. The former called **H**ybrid M**a**rkov C**l**ustering (HAL) clusters only the images taken by user's smartphones and shared on their profiles. The latter, called **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO) can also cluster images taken from different sources, e.g., the Internet, to simulate the real case scenario in which users share images on their profiles that are not taken by the camera of their smartphones. The HALO method is performed by an outlier detection method. The strengths of our proposed methods include overcoming large-scale and high-dimensional device fingerprint datasets, and the loss of image details caused by the process of content compression performed by social networks. The proposed methods are evaluated on our image dataset[1] and the public benchmarking VISION image dataset[2]. Experimental results as well as the comparisons with state-of-the-art algorithms confirm the effectiveness and the robustness of the proposed methods.

[1]The dataset is available from: `http://smartdata.cs.unibo.it/datasets#images`
[2]The dataset is available from: `https://lesc.dinfo.unifi.it/en/datasets`

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Shared Image Analysis based on Camera Sensor Pattern Noise

In recent years, different social networks have revolutionized the Internet and our society by providing specific types of interaction, for instance by sending texts and sharing images and videos. Many social networks provide their own dedicated applications for major mobile devices (e.g. smartphones). This influences user habits with respect to multimedia content on social networks [1]. In particular, this has led users to take more digital images and share them across various social networks [2], making it a challenging task to control image production and propagation and to use such images as a form of digital evidence.

Images shared by social network users can be considered as complementary clues used to detect evidence references in data analysis, e.g., identity theft, online sexual harassment, piracy, illegal trading, cyber stalking and cyber terrorism [3]. In tracing the history of an image, identifying the source captured the image is of major interest. As an application, in a court of law, the origin of a particular image can provide crucial evidence. The validity of this evidence might be compromised by the reasonable doubt that the image has not been captured by the device it is claimed to be acquired with. When the original images and smartphones are available, the validity of the evidence is pretty easy. However, in practical cases, almost it cannot be assumed that prior information about the original image is available.

Important clues on the camera source can be easily found in Exchangeable Image File Format (EXIF) data [4]. However, since this information can be simply modified or can be removed by online social network platforms (due to user's privacy), it cannot always be applied to data analysis. Hence, blind analysis has to be applied to specify the image source. The blind analysis has attracted a growing interest of researchers during the last years. Particularly, blind techniques exploit traces left by different processing steps in source manufacturing and the image acquisition and storage phases. The traces leave footprints in the image [5]. The Sensor Pattern Noise (SPN), due to camera sensor imperfections, is considered as a unique characteristic to fingerprint a source camera [6]. Regarding the SPN, different techniques such as Smartphone Verification (SV), Smartphone Identification (SI), SPN-based Image Clustering (SIC) and User Profile Linking (UPL) have been presented in data analysis.

1.1.1 Smartphone Verification

Given a set of smartphones and a set of user profiles on social networks, we may want to know whether the images shared on the profiles originated from the smartphones or not. It is usually called *smartphone verification*. More specifically, the verification task is a binary classification that carries out 1-by-1 matching between the pairs of the images and the smartphones. The aim of SV is to verify the source camera that has taken a query image, as shown in Figure 1.1. Firstly, the SPN of the camera is constructed, by averaging the Residual Noises (RNs) extracted from a set of images captured by the camera. The images used for constructing the reference SPN are usually those with high intensity and low

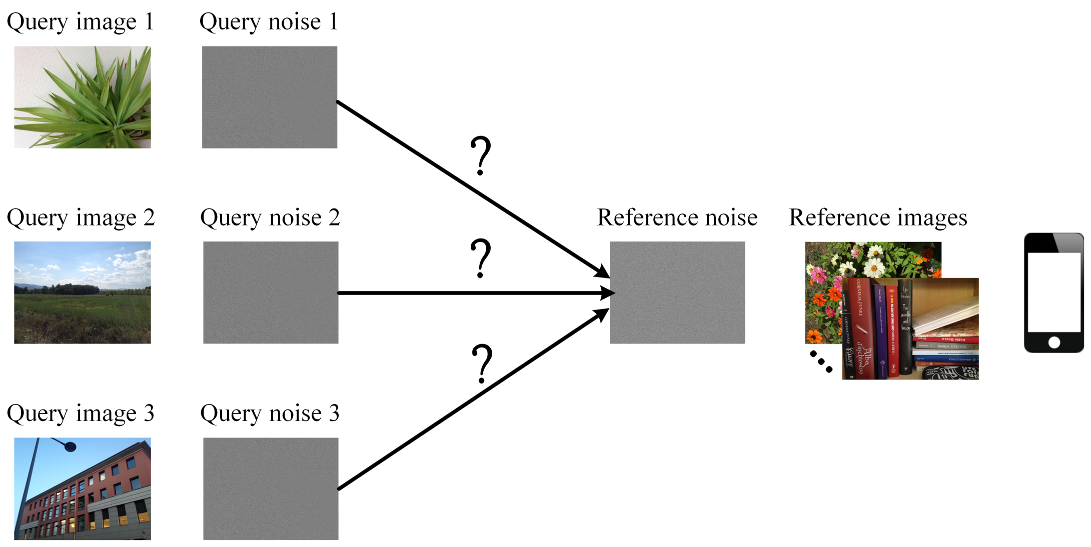


Figure 1.1: Smartphone verification.

texture, e.g., blue sky images as the extracted SPN has a better quality [7]. Next, the RNs of the query images are extracted and compared with the reference camera SPN. If the similarity between the RN of the query image and the reference SPN is high enough, the query image is considered to be captured by the camera.

1.1.2 Smartphone Identification

The second technique is similar to the first, but instead of one, multiple smartphones are available. It is known as *smartphone identification*, which reliably matches a specific image with its related smartphone, as shown in Figure 1.2. Particularly, SI deals with 1-to-m matching problem and determines which smartphone out of, e.g., m, took a given image [8]. First of all, a reference SPN is constructed for each smartphone camera. Then, the RN of the query image is extracted. Next, the similarities between the SPNs and the RN are calculated. If the highest similarity is higher than a predefined threshold, the query image is considered to be taken by the camera with the highest similarity.

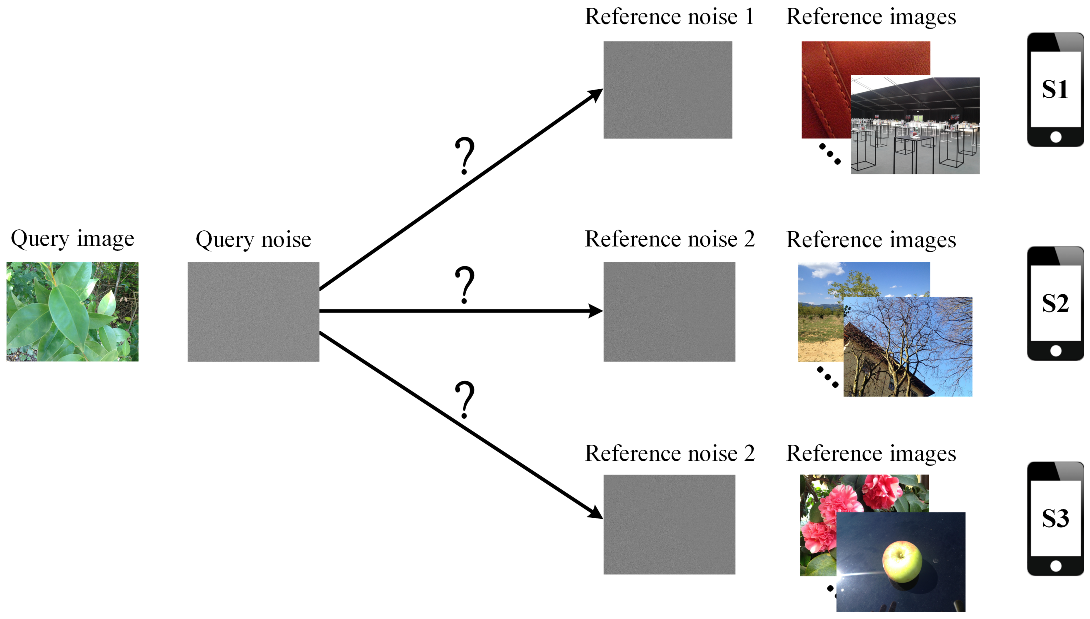


Figure 1.2: Smartphone identification.

1.1.3 Image Clustering

In most real-life cases in data analysis, a large number of shared images on a set of profiles are collected, without any clue of the cameras, and the aim is to group the images into an unknown number of clusters, each of them includes the images taken by the same camera. In the previous tasks, i.e., SV and SI, a set of images taken by the same camera source has to be provided for the construction of the reference SPN. It can be simply fulfilled when the information about the smartphone is available. However, in many real-world scenarios, only a set of images are available, without any information about the types and the number of the sources generated the images.

For example, in Internet child pornography, a large number of illegal images can be collected from pornographic website, but the sources taken these images are not available. If there would be a way to cluster the collected images into a number of groups, each of them includes the images captured by the same camera, it is possible to associate different crime scenes and provide the investigators

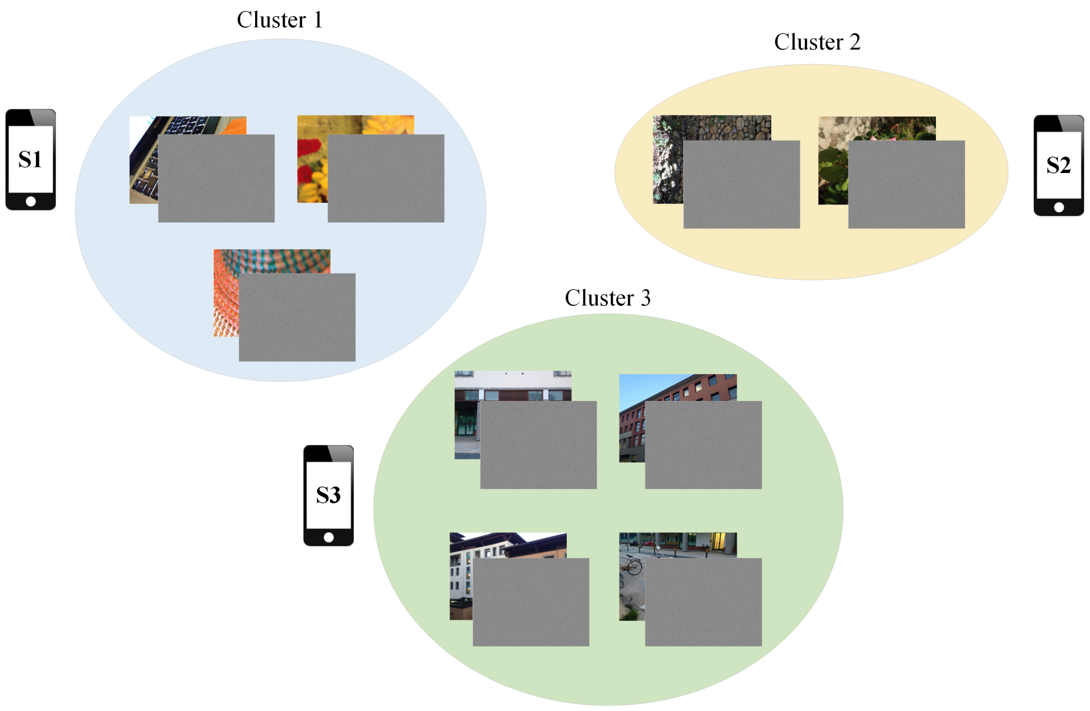


Figure 1.3: SPN-based image clustering.

by more clues to link the evidence to the seized hardware that are owned by the suspects in the future. In Figure 1.3, the task of SPN-based image clustering is shown. By clustering the RNs extracted from the collected images, the corresponding images taken by the same camera are grouped in the same cluster.

1.1.4 User Profile Linking

Since social networks provide users with specific types of interaction, such as sending texts and sharing images and videos, users are usually active across multiple social networks [9].

Once smartphones are fingerprinted in SV, SI, or SIC techniques, by using the shared images on user profiles, with having the profile tags specifying each user on a social network platform, the profiles sharing images from the same camera source are linked, as it was proposed in our previous works [10, 11, 12,

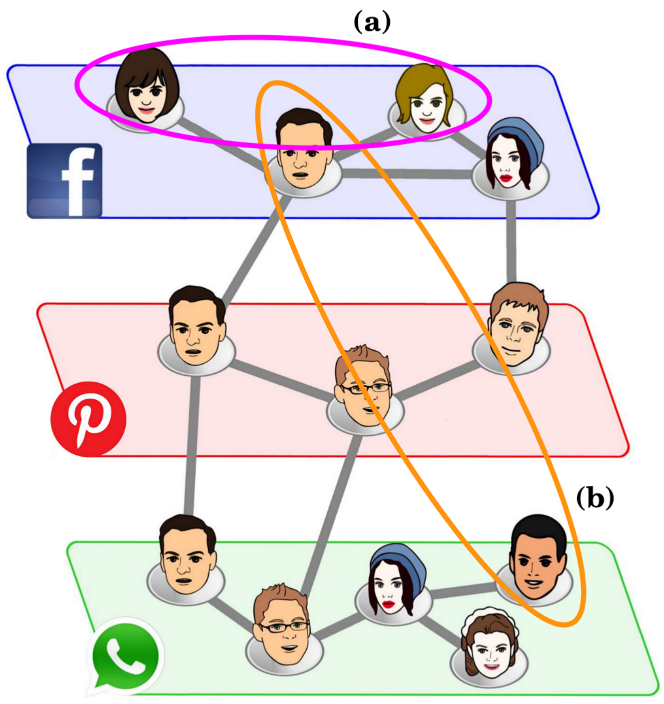


Figure 1.4: Using shared images and fingerprinting the related smartphones: (a) intra-layer UPL and (b) inter-layer UPL can be achieved.

13, 14]. It can be achieved within the same social network, i.e., intra-layer UPL or across different social networks, i.e., inter-layer UPL, see Figure 1.4. It is worth mentioning that a user would be linked to other profiles even if there is not a direct friendship between the profiles on the same or different social network platforms.

1.2 Research Questions

Regarding a set of images taken by a known or an unknown number of smartphones and shared on a set of user profiles on social networks, we aim to address some questions as follows:

If the number of the smartphones is known:

1. Is it possible to cluster the shared images on a social network into a given number of groups, corresponding to the number of smartphones, to generate the smartphone fingerprints and achieve intra-layer UPL?

2. Can the obtained clusters from the shared images on one social network be used to classify a set of shared images on another social network and achieve inter-layer UPL?

If the number of the smartphones is unknown:

3. Is it possible to cluster the captured images by smartphones and shared on user profiles into an unknown number of groups, each of them includes images form the same smartphone?

4. Is it possible to cluster the captured and shared images by smartphones on user profiles in the case of perturbation, where the users also share cut or single images from different sources such as the Web?

1.3 Research Contribution

In this thesis, we propose methods for SI, SIC with or without perturbation as well as UPL tasks within the same or across different social networks. More specifically, we present the methods as follows:

1. We propose a clustering and classification based methods to achieve SI and UPL, where a set of images captured by a known number of smartphones and shared on a set of user profiles are provided. More specifically, we apply k-medoids technique, to cluster both "native" and "shared images" and achieve *original-by-original SI* and *intra-layer UPL*. Whereas, a

classification method based on Artificial Neural Networks (ANNs) is proposed to achieve *social-by-original SI* and *inter-layer UPL*. We classify the "shared images" by exploiting the fingerprints resulting from the k-medoids clustering [15]. The methods are applicable to images compressed on social networks, and there is no need to hack user's smartphone for fingerprinting.

2. We present a **H**ybrid M**a**rkov C**l**ustering (HAL) method by combining hierarchical and Markov clustering algorithms. The HAL method is capable of clustering the images captured and shared on social networks without prior knowledge about the types and number of smartphones. Particularly, the HAL method is precise, scalable and feasible for real life applications.

3. We develop the HAL method to tackle the clustering when users share images, e.g., single images from different sources, cropped images, or images from the Web, such that they cannot be used in fingerprinting their sources. These images make a perturbation, so they have to be detected and removed from the clustering process. We detect the outliers based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique. We call it **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO). We show that the HALO method is stable against the outliers.

4. We have collected and cleaned a dataset including smartphone images available from `http://smartdata.cs.unibo.it/datasets#images`, for the implementation of the proposed methods.

5. We have done software developments for the implementation of the proposed methods efficiently and effectively which is available from `https://tinyurl.com/y2zw8owy`, for social network data analysis.

Chapter 2
Background

2.1 Digital Camera Components

As it can been seen in Figure 2.1, a digital camera has mainly several built-in components such as lens (a), Color Filter Arrays (CFA) (b) and sensors (c).

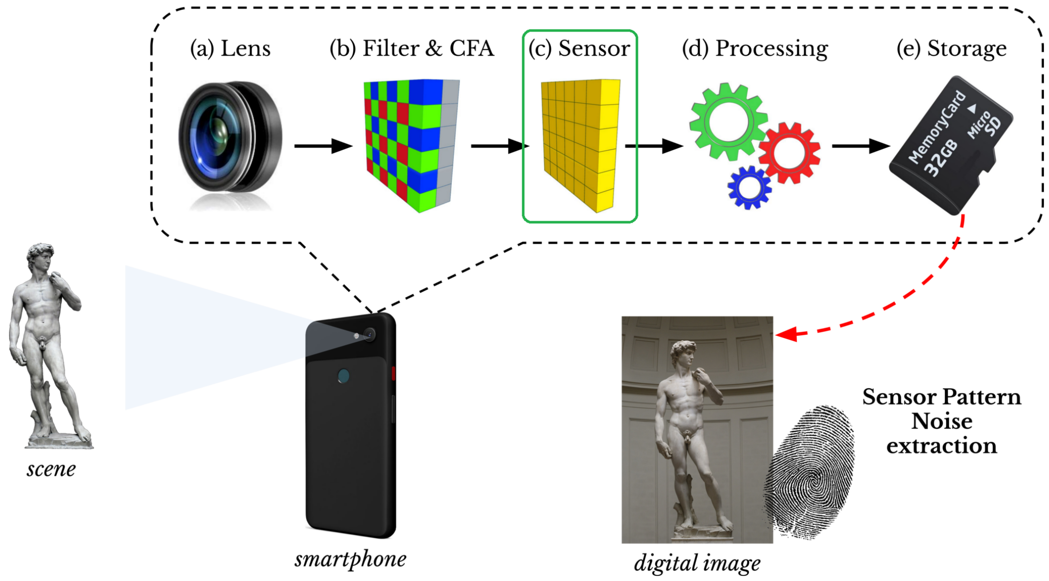


Figure 2.1: Different components embedded in smartphone's camera.

Particularly, the lens produces a similar-prism phenomenon and divides the light beam into a spectrum of rainbow colors. This causes a shift in the point where different wavelengths (colors) converge, that is the characteristic of the lens. Optical anti-aliasing filter and CFA are in front of the image sensor and reconstruct the color information. The color subsampling is affected by noise [16]. All of these components present hardware imperfections created during the manufacturing process that uniquely characterize each smartphone. Particularly, the camera fingerprint formed by sensor imperfections has been known as a reliable characteristic making a smartphone trackable [6, 17, 18].

2.2 Camera Sensor Pattern Noise

The SPN consists of the Fixed Pattern Noise (FPN) and the Photo Response Non-Uniformity (PRNU) noise, as shown in Figure 2.2.

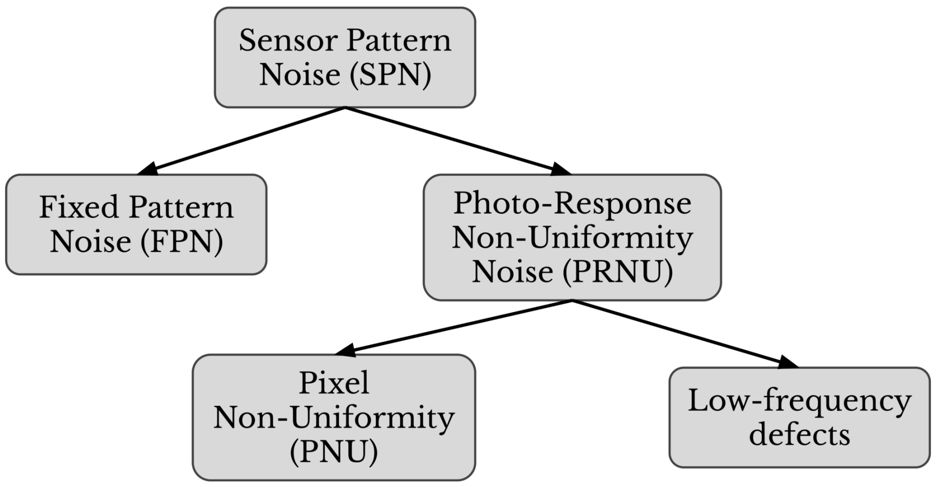

Sensor Pattern Noise (SPN)
Fixed Pattern Noise (FPN)
Photo-Response Non-Uniformity Noise (PRNU)
Pixel Non-Uniformity (PNU)
Low-frequency defects

Figure 2.2: Sensor pattern noise of camera sensor.

The FPN, which is created by dark currents, is the pixel-to-pixel differences when the sensor array is not exposed to light. Since the FPN is an additive noise, some cameras suppress it automatically, by subtracting a dark frame from every captured image. FPN is affected by ambient temperature and exposure. The dominant component in SPN is the PRNU noise. It is generated primarily by Pixel Non-Uniformity (PNU) defined as different sensitivity of pixels to light, which is caused by the inhomogenity of silicon wafers and imperfections. The character and origin of the PNU noise make it unlikely that even sensors from the same wafer would present correlated PNU patterns. So, the PNU noise is not dependent on temperature or humidity. Light refraction on dust particles and optical surfaces and zoom settings also contribute to the PRNU noise. These components are of low spatial frequency in nature and they are not a characteristic of the sensor. Hence, only the PNU component, which is an intrinsic characteristic of the sensor, is used for fingerprinting sensors [19]. PRNU is a strong tool for fingerprinting smartphones as it is unique for an individual device. Besides, it is stable against environmental conditions [7].

2.2.1 Camera Sensor Pattern Noise Extraction

Camera sensor imperfections remain stable as the RNs in the images. Each RN is the difference between the image content and its denoised version acquired by a denoising filter $d()$. The RN of an image I is extracted as follows [6]:

$$RN = I - d(I) \tag{2.1}$$

By averaging the RNs extracted from n images taken by a given smartphone, the SPN, i.e., the camera fingerprint, can be approximated by:

$$SPN = \frac{1}{n}\sum_{j=1}^{n} RN_j \tag{2.2}$$

Based on (2.1) and (2.2), it is obvious that the quality of the extracted RNs and SPN depends on $d()$ and n. Block-Matching and 3D (BM3D) denoising filter introduced by [20] is an accurate way to extract the RNs with better qualities.

Through BM3D, non-unique artifacts are removed by using zero-meaning all columns and rows, and Wiener filtering in the Fourier domain [21], [17, 22].

2.2.2 Camera Sensor Pattern Noise Similarities

To perform different tasks in data analysis such as smartphone verification, smartphone identification and SPN-based image clustering, the similarities between the camera fingerprints, whether RNs or SPNs, need to be calculated. The Normalized Cross Correlation (NCC) similarity between any two camera fingerprints $f_i = [x_1, ..., x_l]$ and $f_j = [y_1, ..., y_l]$ is calculated as follows:

$$\mathcal{A}(f_i, f_j) = \frac{\sum_{n=1}^{l}(x_n - \overline{f}_i)(y_n - \overline{f}_j)}{\sqrt{\sum_{n=1}^{l}(x_n - \overline{f}_i)^2 \sum_{n=1}^{l}(y_n - \overline{f}_j)^2}} \tag{2.3}$$

where $\overline{f}_i$ and $\overline{f}_j$ represent the means of the two fingerprints, respectively. Generally, the correlations between the fingerprints from the same camera are higher than those from different cameras [23]. The NCC has been applied in many SPN-based techniques in the computation of similarities between camera fingerprints in the literature as it is the optimal metric for multiplicative signals such as PRNU [19].

2.3 Classification and Clustering Algorithms

Classification (supervised learning) and clustering (unsupervised learning) are the two types of learning methods which organize objects into groups based on one or more features. They appear to be similar, but they are different in the context of data mining. Classification is the process of learning a model that elucidate different predetermined classes of data. It is a two-step process composed of a learning step and a classification step. In the learning, a classification model is constructed, and it is trained in a supervised approach, such that predefined labels are assigned to objects by features. Then, the trained classifier is given the objects whose labels are unknown, to assign them a label as their class.

On the contrary, clustering is performed in an unsupervised learning approach where similar objects are grouped, based on their features. It does not involve training or learning, and the training sample is not known previously. It organizes objects into clusters where the objects reside inside a cluster will have high similarity and the objects of two clusters would be dissimilar to each other. Here the two clusters can be considered as disjoint [24].

2.4 Outlier Detection

In data mining, an outlier is an object that differs significantly from other objects in a dataset [25]. Typically, outliers are a minority of objects that are inconsistent with the pattern presented by the majority of objects in the same dataset [26]. Cluster analysis and outlier detection are strongly coupled tasks. The resulted clusters can be simply destroyed by a few number of outliers. Outliers are defined by the concept of the cluster and they are recognized as the objects which are not assigned to any cluster. Most of the existing clustering methods, generally and also specifically presented for the application of SPN-based image clustering, assume that all the objects (images in our case) should be assigned to a cluster label, meaning that there are no phases for outlier detection in the clustering methods. This is not always true, especially for the unsupervised clustering. The outliers unavoidably degrade the clustering effectiveness. For instance, only a few outliers easily destroy the cluster structure resulting from k-means and generate bizarre distributions of Gaussian mixture model [27]. Consequently, the clustering should be provided with an outlier detection phase.

2.5 Evaluation Measures

The classification and clustering are evaluated by different measures. There are four definitions based on the agreement between two sets of labels, i.e., ground truth or target labels $T = \{t_1, t_2, ..., t_g\}$ and the resulted labels $C = \{c_1, c_2, ..., c_h\}$, where g and h are the number of the target and the resulted labels, respectively. Given two samples d_i and d_j in a dataset, e.g., $\mathcal{D} = \{d_1, d_2, ..., d_N\}$, where N is the number of samples in the dataset, we have the following definitions:

i. True Positive: $TP = |\{(d_i, d_j) : t_i = t_j \textit{ and } c_i = c_j\}$ meaning that the two samples d_i and d_j belong to the same group in T, and they are also in the same output group in C.

ii. False Negative: $FN = |\{(d_i, d_j) : t_i = t_j \textit{ and } c_i \neq c_j\}$ meaning that the two samples d_i and d_j belong to the same group in T, while they are not in the same output group in C.

iii. False Positive: $FP = |\{(d_i, d_j) : t_i \neq t_j \textit{ and } c_i = c_j\}$ meaning that the two samples d_i and d_j do not belong to the same group in T, but they are in the same output group in C.

iv. True Negative: $TN = |\{(d_i, d_j) : t_i \neq t_j \textit{ and } c_i \neq c_j\}$ meaning that the two samples d_i and d_j do not belong to the same group in T, and they are also not in the same output group in C.

Having the above definitions, *Precision rate* $\mathcal{P}$, *Recall rate* $\mathcal{R}$ also known as *True Positive Rate* (*TPR*), *F1-measure* ($\mathcal{F}1$), *F2-measure* ($\mathcal{F}2$), *Rand In-*

dex (*RI*), *Adjusted Rand Index* (*ARI*), *Purity*, *False Positive Rate* (*FPR*) are depicted by (2.4)-(2.11):

$$\mathcal{P} = \frac{|TP|}{|TP| + |FP|} \tag{2.4}$$

$$\mathcal{R} = \frac{|TP|}{|TP| + |FN|} \tag{2.5}$$

$$\mathcal{F}1 = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{2.6}$$

$$\mathcal{F}2 = 5 \cdot \frac{\mathcal{P} \cdot \mathcal{R}}{4 \cdot \mathcal{P} + \mathcal{R}} \tag{2.7}$$

$$RI = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|} \tag{2.8}$$

where $|.|$ shows the number of the pairs in the corresponding set defined in i-iv. The value of *RI* varies between 0 and 1, respectively showing no agreement and full agreement between the clustering results and the ground truth. For two random clusters, the average of $\overline{RI}$ is a non-zero value. To get rid of this bias, *ARI* was proposed in [28]:

$$ARI = \frac{RI - \overline{RI}}{1 - \overline{RI}} \tag{2.9}$$

Also, we use *Purity* and *FPR* in the evaluations as follows:

$$Purity = \frac{\sum_{i=1}^{|C|} \frac{|\widehat{c}_i|}{|c_i|}}{|C|} \tag{2.10}$$

where $|C|$ is the number of the obtained classes, $\widehat{c_i}$ denotes the number of RNs with the dominant class label in the cluster c_i, and $|c_i|$ is the total number of RNs in c_i.

$$FPR = \frac{|FP|}{|FP| + |TN|} \tag{2.11}$$

In addition, in clustering, the ratio of the number of the obtained clusters that is n_h over the number of ground truth clusters denoted by n_g is calculated as follows:

$$\mathcal{N}_C = \frac{n_h}{n_g} \tag{2.12}$$

We calculate $\mathcal{N}_U$ for the clustering as well:

$$\mathcal{N}_U = \frac{n_u}{N} \tag{2.13}$$

where n_u is the number of unclustered RNs.

Also, for the outlier detection, $\mathcal{N}_O$ is defined as follows:

$$\mathcal{N}_O = \frac{n_d}{n_o} \tag{2.14}$$

where n_d and n_o are the number of the detected and ground truth outliers, respectively.

Chapter 3
Related Works

3.1 Source Camera Identification

Many approaches have been proposed to get smartphone fingerprints using a variety of built-in sensors such as accelerometers [29], gyroscopes [30], magnetometers [31, 32], cameras [33], and paired microphones and speakers [34]. All of them have hardware imperfections, which are created during the manufacturing process, and can be used in the smartphone fingerprinting. The camera has been considered as a sensor that is less invasive and more reliable for source camera identification [35]. Generally, the majority of the source camera identification methods proposed in the literature extract a pattern noise from the captured images. Particularly, a statistical similarity such as correlation between the image and the computed pattern noise is evaluated. The source camera identification methods can generally be categorized into lens system aberrations, Color Filter Arrays (CFA) interpolation, and sensor imperfections [35], corresponding to the camera components shown in Figure 2.1.

The lens causes some aberrations such as spherical, coma, astigmatism, field curvature, radial distortion and chromatic aberration. Choi et al., [36] showed that a high value of accuracy can be achieved in the identification, by measuring the intrinsic lens radial distortion of each camera. They used parameters from aberration measurements to train and test a support vector machine (SVM) classifier. Several studies presented methods based on CFA which consider characteristics of the interpolation algorithm that generate some obvious differences between camera models. Bayram et al., [37], studied the process of CFA interpolation to find the correlation pattern, appearing in each color channel of the image. Expectation and Maximization (EM) algorithm and SVM classifiers were applied, respectively, to identification of the traces of demosaicing process and classification of various source camera groups. The method failed in the identification of similar camera models, due to the identical CFA pattern and the interpolation algorithm they share. Celiktutan et al., [38] proposed a correlation based method for identifying the smartphone cameras, where the correlation existing between different bit planes of the image was calculated using binary similarity measures and image quality metrics. A k-Nearest Neighbor (KNN) classifier was trained by the extracted features to identify cameras. The techniques based on demosaicing artifacts failed to identify the cameras of identical models since they use similar demosaicing algorithm.

Some methods were proposed based on camera Sensor Pattern Noise (SPN) for camera source identification. Lukáš et al., [39] introduce a reliable method for identifying the source camera using Photo Response Non-Uniformity (PRNU) noise. A significant advantage of the PRNU is that it remains stable under different environments, so it can result in a reliable fingerprint that characterizes the digital device effectively, even the identical camera models. The effectiveness of the method proposed in [39] was not found to be satisfactory for the cropped images and images with different sizes. The estimation of the exact PRNU noise from the images was studied in some works by improving the PRNU extraction methods. Chan et al., [40] proposed a confidence map and pixel based weighted correlation method to eliminate the scene effect left in the estimated PRNU noise in order to enhance the accuracy. Gisolf et al., [41] proposed a new method to extract PRNU from the images by using simplified version of Total Variation (TV) based noise removal algorithm, to denoise the data without blurring the edges. Also Cattaneo et al., [42] proposed a feasible distributed and scalable implementation of the work presented by Lukáš et al.

In a number of works, researchers concentrated on extracting some features based on color characteristics, image quality metrics and wavelet transformation for source camera identification. The authors of [43] proposed a method to use the image features for identifying an individual camera, where each image is represented by using a numerical feature vector composed of color related features, image quality measures and statistical features from the wavelet domain. They trained an SVM classifier based on the features extracted from both the spatial domain and the wavelet domain. The reported results for five different cameras show the error rates between 5% and 22%. Celiktutan et al., [44] proposed a feature fusion and decision fusion schemes using binary similarity measures, image quality measures and higher order wavelet statistics. An SVM classifier was trained by the extracted features, to classify the images from several smartphone cameras. Orozco et al., [45] introduced a method using the mixture of two techniques, i.e., sensor imperfections and wavelet transform as well as SVM classifiers to achieve better identification rate. Tsai et al., [46], proposed a method by extracting the features of Charge-Coupled Device (CCD) embedded in most of digital cameras. Hence, the method cannot achieve desirable results with similar CCD. Besides, it is not effective at fingerprinting cameras of identical models [35].

Recently, some methods have been proposed based on deep learning techniques. For example, Baroffio et al., [47] introduced a three convolutional and two fully connected layers architecture, which achieves an error rate about 6%. Tuama et al., [48] proposed an architecture also based on three convolutional layers, unlike the work of Baroffio et al., [47], they applied a single pooling layer after the convolutional layers. Tuama et al., [48] obtained an error rate of less than 10% from their experiments. The work of [49] focused on designing an architecture to create a more effective tool for source camera identification. They proposed an architecture along with the hyperparameters tuning which created a decision system to classify the given image into the corresponding source category. The model was sufficiently complex to effectively distinguish between both problems, an accuracy value of 98.1% for identifying the device manufacturer and 91.1% of accuracy for identifying the exact camera.

3.2 Image Clustering based on Camera Sensor Pattern Noise

Many works have been done on SPN-based image clustering. As a pioneering work, Bloy in [50] presented an unsupervised clustering algorithm considering the RNs as singleton clusters and hierarchically merging the similar clusters. The pairs of the clusters are randomly selected and if their correlation exceeds a threshold, they are merged into a new cluster, which is represented by its centroid, i.e., the corresponding SPN. It is based on the idea that the more images are clustered, the better the quality of SPNs can be obtained. As a drawback, the algorithm produces the threshold based on a quadratic model, which does not generalize well across various source cameras.

In [51], Markov random fields are used to assign iteratively a class label to an image, according to the consensus of a small set of SPNs, called membership committee. This raises an issue on how to choose an appropriate committee, particularly for the datasets with various cluster cardinalities, i.e., asymmetric datasets. The authors of [52] developed a faster algorithm by proposing a new enhancer applied to the extracted SPNs. A random subset, i.e., training set, of the entire dataset is used for clustering, which is followed by a classification step for the remaining fingerprints, i.e., test set. The algorithm merges the clusters hierarchically, and a silhouette coefficient is calculated for each cluster. Particularly, the silhouette coefficient estimates the separation among clusters as well as the cohesion within each cluster. The average of the silhouette coefficients corresponding to the produced clusters in each iteration is considered as a merit of the clustering, showing its quality. Once all fingerprints are merged into one cluster, a set of clusters with the highest value of the merit is selected as the optimal result of the clustering. In [53], a similar unsupervised algorithm was proposed. The main difference is that the evolutionary process of the cluster formation is used in the calculation of the coefficient. The main problem of the proposed hierarchical algorithms is that they are sensitive to noise and outliers as wrong assignments may propagate the error to the following iterations in the clustering. Also, their computational complexities are high especially for high dimensional RNs because all the cluster pairs have to be checked for a merging.

The graph-based algorithms have successfully been applied to SPN-based image clustering by computing a small portion of the full-pairwise correlation matrix. In [54], a method based on k-nearest neighbor technique was proposed, where the clustering is regarded as a graph partitioning problem. Each image is considered as a node and the correlation values between the RNs are considered as the weights of the edges. Next, the nodes are partitioned into disjointed sets by using spectral analysis. Besides the need for the user to provide the number of clusters, a major problem of this method is that its quality depends on the random initialization. In [55], the problems were addressed by means of the normalized cut graph partitioning algorithm [56], which produced better clustering results without providing the number of clusters as an input parameter. However, the stopping condition is met once all the aggregation coefficients computed for the obtained clusters are greater than a pre-defined threshold, which is an important input parameter itself.

In [57], the clustering is performed based on correlation clustering, formulating the graph partitioning problem as constrained energy minimization. A refinement step is applied to generate clusters with higher qualities. The disadvantage is that it needs a parameter set by the user according to preliminary analyses on an appropriate training set. The issue of parameter setting was handled in [58] by consensus clustering applied to all the cluster partitions obtained from correlation clustering, to extract a unique solution. Generally, the average correlations between SPNs of one camera may remarkably differ from that of other cameras. This makes the clustering more difficult. To tackle the issue, in [59], shared nearest neighbors [60] are applied to the full-pairwise correlation matrix, to find clusters with different sizes and densities. A common undesirable trait of the mentioned algorithms is their need for full-pairwise correlation matrix, which may prevent their use for large-scale datasets in practical applications.

Only a few studies considered the scalability aspect of SPN-based image clustering. In [61], large-scale clustering was handled by partitioning the dataset into small batches, which could fit in RAM efficiently, and applying a coarse-to-fine clustering method. An adaptive threshold was proposed for merging the obtained clusters based on the quality of the clusters iteratively updated during the clustering. The authors of [62] used the similar partitioning approach and exploited linear dependencies among SPNs in their intrinsic vector subspaces. It uses a training phase to generate an adaptive threshold for merging the obtained clusters. However, the training may lead to over-fitting in some datasets. Also, compared with the threshold proposed in [61], it tends to be too radical for clusters with large size, which is a critical point in real-life applications. However, these scalable clustering algorithms were only tested on *native images* and their robustness on *shared images* on social networks is still in doubt.

3.3 Outlier Detection in Clustering

Outlier detection, also known as anomaly detection, recognizes the objects deviated from the others and identifies these objects as outliers. In most of the existing works, unsupervised outlier detection was studied, in such a way that each object is given a score based on some criteria, and the objects with large scores are considered as the outlier candidates [27]. Some representative methods include density based Local Outlier Factor (LOF) [63], Connectivity-based Outlier Factor (COF) [64], Local Distance-based Outlier Factor (LODF) [65], Frequent Pattern-based outlier detection (Fp-outlier) [66], Angle-Based Outlier Detection (ABOD) [67], Fast Angle-Based Outlier Detection (FABOD) [68], ensemble-based isolation Forest (iForest) [69], Bi-Sampling Outlier Detection (BSOD) [66], Oversampling Principal Component Analysis (OPCA) [70], cluster-based Text Outliers using Non-negative Matrix Factorization (TONMF) [71]. Recently, some methods based on deep learning were proposed for outlier detection, such as deep one-class SVM [72] and Generative Adversarial Networks (GAN)-based methods [73, 74, 75], learning a non-linear transformation in order to project the original data into hidden space for more effective recognition. These methods are supervised and train the model only with accurate samples and predict the label of new samples whether they are outliers or not. Hence, the training phase is crucial and challenging.

Clustering (unsupervised learning) and outlier detection are mostly considered as two independent tasks in the data mining area, although in the real-life applications, they are coupled task. There are few works presented a unified framework for cluster analysis and outlier detection. For example, a developed version of k-means algorithm was proposed in [76], called k-means--, which detects outliers and groups the remaining objects into k clusters, where the objects with large distance from the nearest centroid are considered as outliers during the clustering process. Langrangian Relaxation (LP), presented in [77], formulates the clustering task with outliers as an integer programming problem, which requires the cluster creation costs as the input parameter. Charikar et al., [78] proposed a bi-criteria approximation algorithm for the facility location with outliers problem. Chen [79] proposed a constant factor approximation algorithm for the k-medoids clustering with outliers. Although some pioneering works introduced new approaches for joining clustering and outlier detection phases, none of these algorithms, except k-means--, are applicable to large-scale datasets. However, the spherical structure assumption of k-means-- and the original feature space limit its capability for clustering complex data. Liu et al., [27] introduced a Clustering with Outlier Removal (COR) method, which partitions an entire dataset into several clusters and one outlier cluster, separately. The COR method transforms the original feature space into the partition space, where according to Holoentropy, the COR is designed to provide simultaneous consensus clustering and outlier detection. Some methods proposed for outlier detection based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [80]. For example, Saki et al., [81] proposed an online framework clustering for classification of streaming data provided by an oulier detection phase based on DBSCAN clustering. To the best of our knowledge, only the work presented in [62] considered the outlier detection, based on DBSCAN technique, and evaluated their method robustness against outliers. However, their method was not evaluated in such a case that the number of outliers exceeds the number of the other images. Hence, more effort is needed to study SPN-based image clustering particularly with the presence of outliers particularly.

3.4 User Profile Linking

Different methods have been proposed for the User Profile Linking (UPL) task. For example, Al Mutawa et al., [82] exploited user activities on social networks. They collected logs filed within the device through a manual investigation and used them the information to match user profiles. Their experiments showed that the method failed for BlackBerry devices. Similarly, [1] monitored user activities and collected a variety of artifacts, such as usernames, passwords, login information, personal information, uploaded posts, and exchanged messages. The authors of [83] used the Jaro-Winkler distance algorithm [84], to compare the account information of users, such as username, friends, and interests, from accounts on different social networks for profile matching. Iofciu et al., [85] introduced a method based on the combination of user IDs and tags to recognize users through the social tagging system.

In [86, 87], some frameworks were presented for UPL across social networks based on profile attributes. The frameworks assign a different similarity measure to each attribute. Gupta et al., [88] introduced a method which is not dependent on login credentials. The behavioral traits of users were applied to link users. Zafarani et al., [89] applied behavioral patterns to establish a mapping among identities of individuals across social media sites. Naini et al., [90] used datasets like call records and matched the obtained histograms of users' data, representing their fingerprints to identify users. However, there are still some problems with these methods. The information of users' identities could be diverse on different social networks, [91]. The users may select different nicknames and e-mail addresses, resulting incorrect matching between the real person and the accounts [89]. Interestingly, Bertini et al., [11] applied images shared on user profiles and performed smartphone verification for UPL on different social networks. Afterwards, the method was developed based on classification and clustering techniques, to achieve intra-layer UPL and inter-layer UPL [13, 14]. The work proposed in [92] was specially dedicated to clustering the shared images on social networks with the application of UPL.

Chapter 4

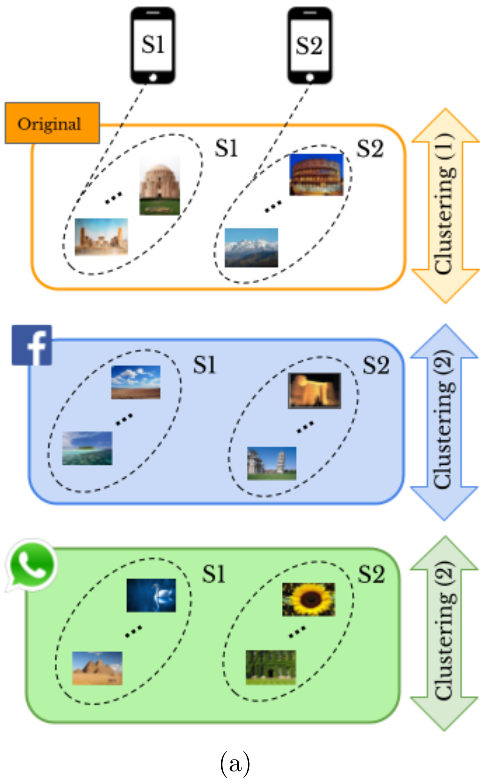
Smartphone Identification and User Profile Linking

Smartphone Identification (SI) is the task of identifying the source cameras generated the images, and User Profile Linking (UPL) is the task of determining whether two profiles with different nicknames or IDs are linked or not. These tasks can be achieved by classification or clustering. In this thesis, we present a solution for SI and UPL within the same or across different social networks based on clustering and classification techniques, providing significant digital evidence for social network data analysis. Through the following sections, the proposed methods are illustrated and the obtained results are discussed.

4.1 Problem Statement

Given a set of images, "native" or "shared images", taken by a given number of smartphones, and a set of user profiles, as shown in Figure 4.1, we aim to perform SI and UPL tasks based on clustering and classification of RNs extracted from the images. A visual example of the proposed methods for two smartphones and two social networks, *Facebook* and *WhatsApp*, is provided in Figure 4.1 (b) and (c). For SI, we consider the following cases.

1.1 *Original-by-original SI* is the task used to detect the source cameras from which a set of "native images" directly coming from smartphones



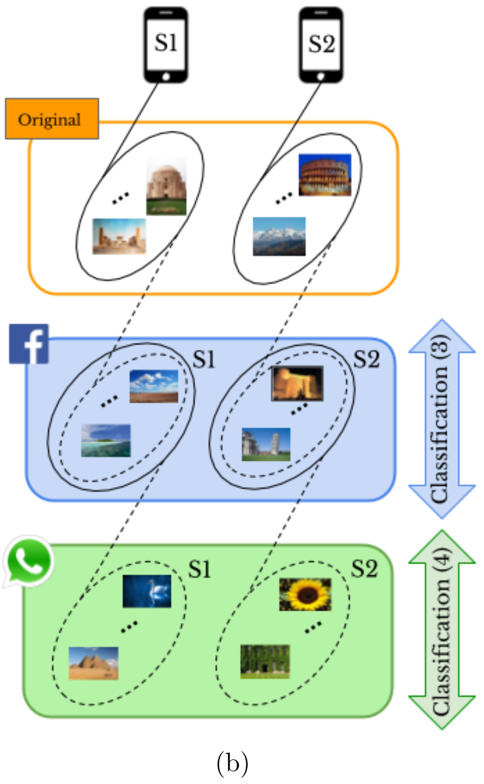


Figure 4.1: A visual example of the proposed methods: (a) clustering and (b) classification based methods for SI and UPL by "native" and "shared images", respectively. The labels (1) to (4) refer to Figure 4.2 presenting all the combination of "native" and "shared images".

have been taken, see the arrow labeled "Clustering (1)" in Figure 4.1 (a).

1.2 *Social-by-original SI* represents the task used to identify the source cameras of a given set of "shared images", see the arrow labeled "Classification (3)" in Figure 4.1 (b). In this case, the "native images" are input data and allow one to define the smartphone camera fingerprints.

Moreover, the UPL task is categorized into two cases: within the same social network and across different social networks, resulting in the following tasks:

2.1 *Intra-layer UPL* is the task used to link a given set of user profiles within the same social network using "shared images", see the arrows labeled "Clustering (2)" on *Facebook* and *WhatsApp* in Figure 4.1 (a). Through this task, the profiles within the same social network that share images from the same source are linked.

2.2 *Inter-layer UPL* represents the task used to link a set of user profiles across different social networks by using "shared images", see the arrow labeled "Classification (4)" in Figure 4.1 (b). Through this task, the profiles from different social networks that share images from the same source are linked.

4.2 Contribution

We consider both "native" and "shared images" for fingerprinting the smartphones. Figure 4.2 shows all the combinations of both types of images. Labels (1) to (4) make a connection with Figure 4.1 (b) and (c), presenting the same meaning. More specifically, we apply the k-medoids technique [93], to cluster "native" and "shared images" and achieve *original-by-original SI* and *intra-layer UPL* (i.e., the green and magenta rounded arrows in Figure 4.2). Whereas, a classification method based on Artificial Neural Networks (ANNs) is proposed to achieve *social-by-original SI* and *inter-layer UPL* (i.e., the blue and red straight arrows in Figure 4.2). In particular, we classify the "shared images" by exploiting the fingerprints resulting from the previous clusters (refer to Section 4.3.3 for more details).

The obtained results show the effectiveness of the proposed methods, even for the images degraded through the compression process on the applied social networks. Moreover, the methods are device-independent and able to distinguish

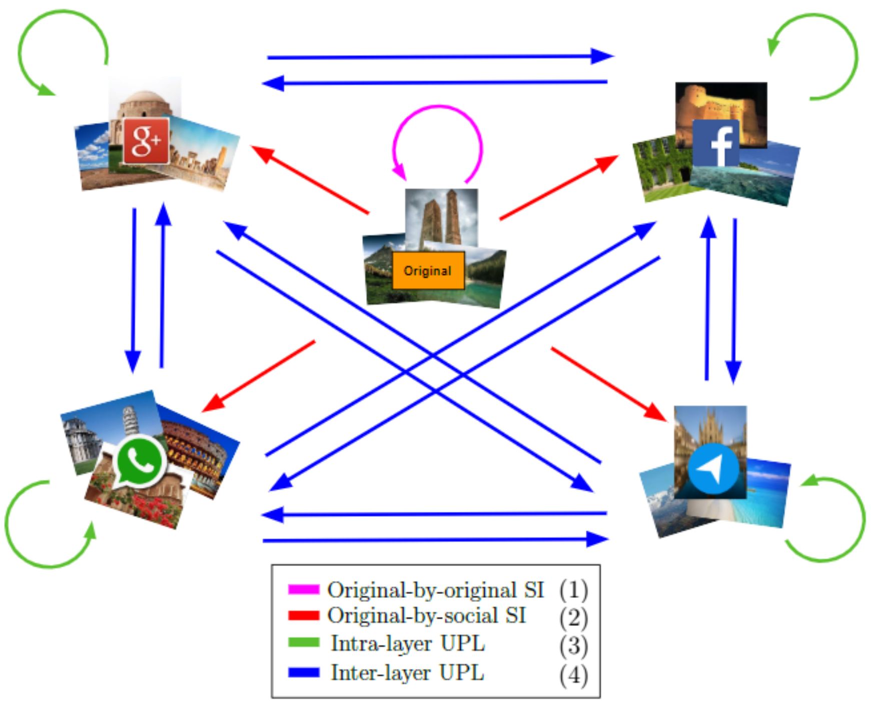


Figure 4.2: Proposed SI and UPL methods applied to “native” and “shared images”. The green and magenta rounded arrows from A to A imply clustering images of A, while the blue and red straight arrows from A to B mean that we use the clustered images of A to classify the images of B.

the same model of smartphones. An important result of our work is applying the *inter-layer UPL* task to link a given set of user profiles on different social network platforms. This is more desirable in shared image analysis because on average, users are active on multiple social networks [94].

The rest of the Chapter is organized as follows. Section 4.3 present the proposed SI and UPL methods. Experiments and their results are discussed in Section 4.4. Some concluding remarks are made in Section 4.5.

4.3 Smartphone Identification and User Profile Linking

4.3.1 Preparation

First of all, since images come from different devices, some pre-processing has to be applied to images. This also helps to result in a reasonable computational cost in terms of both memory and running time of the algorithms.

4.3.1.1 Image pre-processing

Orientation. As the SPN is dependent on the orientation of images, the correct image orientation has to be provided. By the EXIF tool, the metadata information of images can be obtained to get their correct orientation. Then, all the images are rotated to either portrait or landscape orientation [95], see Figure 4.3.

Some smartphone setting and social network platforms remove the orienta-

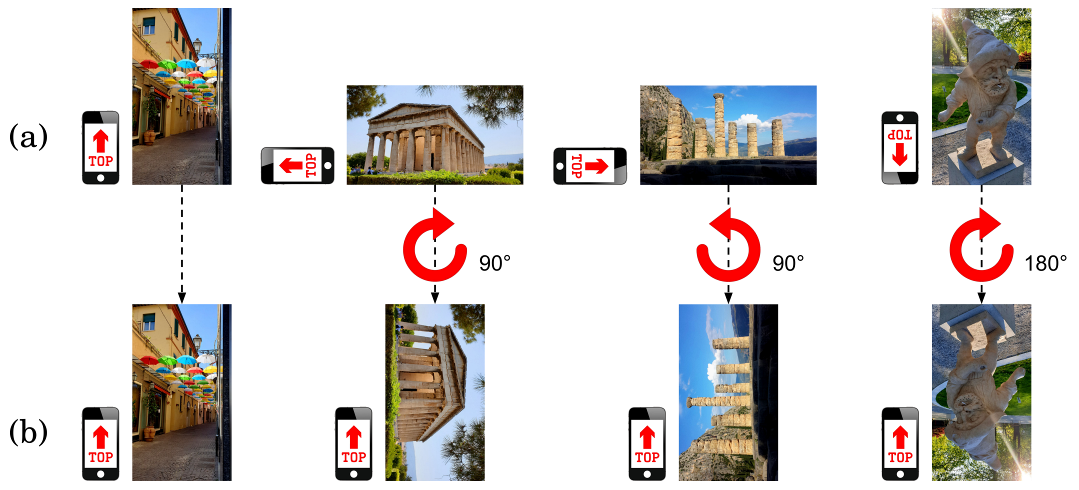


Figure 4.3: Normalization of images' orientation: (a) the smartphones' orientation along with the resulting images, and (b) the corresponding images after rotation.

tion information from the file header of images, due to the user privacy. However, for those images whose orientation information is lost there are still some ways to tackle the orientation issue. For example, once the NCC between RNs is calculated by (2.3), one of the RNs is rotated 180° and again the NCC is computed. Next, the highest value is considered as the similarity between the RNs. This approach is time consuming and is not applicable to large-scale datasets. In [96], a rotation-invariant binary representation of SPN was proposed, which reduces the computational cost, but it has the penalty of loosing information.

Channel. Digital images can be represented in different color spaces, commonly RGB and YCbCr. A color RGB image consists of three channels namely Red (R), Green (G), and Blue (B), while YCbCr represents color images as brightness/luma (Y) and two color difference signals, i.e., blue minus luma (Cb) and red minus luma (Cr). The luma component in YCbCr color space is essentially the grayscale copy of the image. Smartphones are equipped with RGB camera, however, the YCbCr representation can be obtained through a mathematical coordinate transformation from the associated RGB color space. The SPN extraction can be performed using all these different channels. To reduce memory usage and the computational cost, we use the Y channel (grayscale version) of images in this thesis, as it resulted in better effectiveness in the clustering task, in our previous work [13].

Since dark and saturated images do not provide trustworthy RNs [21], including these images makes the clustering task unreliable and computationally cumbersome [61]. Therefore, we recognize these images and exclude them from our experiments. The value of each pixel in a grayscale image can be in the range [0 - 255], where the values 0 and 255 represent black and white colors, respectively. Accordingly, the image whose 70% of pixel intensities are smaller than 50 or greater than 250 is considered as a dark or saturated image, respectively.

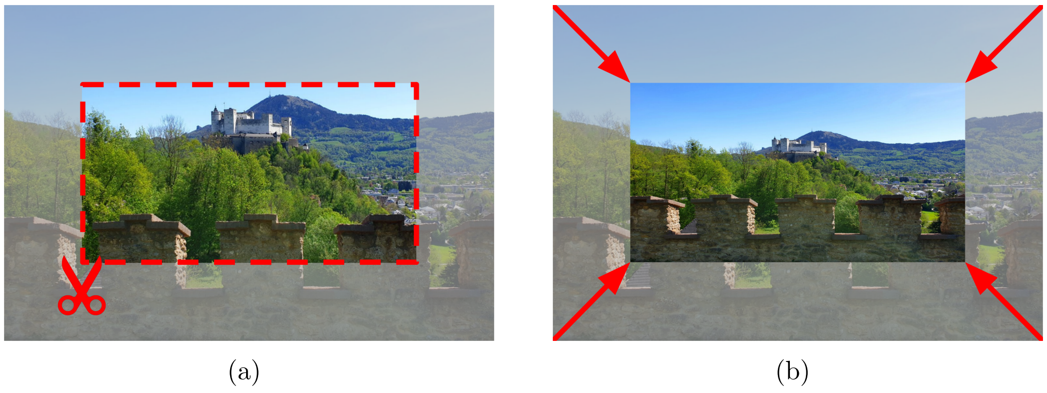


Figure 4.4: (a) Cropping versus (b) resizing.

4.3.1.2 RN resizing

After extracting the RNs from the full-size grayscale images by 2.1, unlike many works, which cropped RNs (often from the center), we resize them to a specific resolution based on bicubic interpolation [97]. Indeed, resizing is a flexible way to calculate the correlations between the RNs with different resolutions. Assume that images in a dataset were captured by 2 smartphones and have different resolutions such as 2560×1920 and 960×720 px. To calculate correlations, all the RNs are typically cropped to the lowest resolution, i.e., 960×720 px in this case. Consequently, a large segment of RNs with the highest resolution, i.e., 2560×1920 px, is discarded. Conversely, by resizing, we can flexibly upscale the lowest resolution or downscale the highest resolution to a specific size for more efficient use of available information, see Figure 4.4. Actually, zero-padding can be another option to handle the computation of correlations between RNs with different resolutions. It may give rise to other issues, e.g., memory usage, but it is worth further investigation.

4.3.2 Clustering-based SI and intra-layer UPL

As shown in Figure 4.1 (b), we apply clustering to group unlabeled images into a given number of clusters, which is the number of the smartphones generated the images. The clustering is performed based on the correlation matrix computed by (2.3). In hierarchical clustering, the objects are typically organized into a dendrogram (tree structure), where leaf nodes represent the individual data and the root is the whole dataset. The middle nodes represent merged groups of similar objects [98]. In partitional clustering such as k-means [99], and k-medoids the objects are divided into some partitions, each of which is considered as a cluster. The partitional clustering starts by initializing a set of k cluster centers. Then, each object is assigned to the cluster whose center is the nearest [93, 100]. K-medoids is an expensive method, but it is more reliable in the presence of noise and outliers compared to the other clustering methods [101].

We compare the hierarchical, k-means, and k-medoids techniques to cluster the "native images" and achieve *original-by-original SI.* This also allows us to select the best method for clustering smartphone camera fingerprints. Then, in a similar way, we cluster the "shared images" to achieve *intra-layer UPL.* Figure 4.5 shows the task of *original-by-original SI.*

Let I be a set of the "native images", and $S = \{S_1, S_2, ..., S_m\}$ be a set of m camera sources. We aim to cluster the images of I into the right sources of S, where each camera source S_i has its own set of images, i.e., $I_{\langle 1,i \rangle}, ..., I_{\langle j,i \rangle}, ..., I_{\langle n,i \rangle} \in S_i$. Thus, we have the full dataset $I = \bigcup I_{\langle i,j \rangle},\ \forall\ i = 1, ..., n$ and $\ j = 1, ..., m$, where n is the number of the collected images for each smartphone. Firstly, we extract the RNs of the "native images" such that $RN_{<i,j>}$ is the RN corresponding to i^{th} image taken by j^{th} smartphone.

We create a matrix $\mathcal{A}$ containing correlations between each pair of the ex-

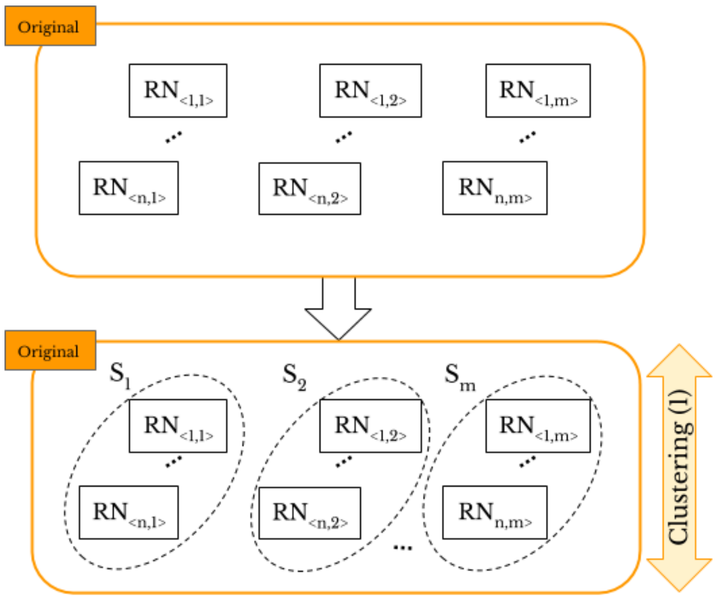


Figure 4.5: *Original-by-original SI* task based on clustering: the "native images" are clustered according to their sources.

tracted RNs by (2.3). Because of the varying qualities of SPNs of different cameras, the average correlation between the RNs from one camera is different from that of other camera [59]. This makes the clustering of SPNs more challenging. To address this problem, an alternative similarity measure is calculated [59], based on Shared $\mathcal{K}$-Nearest Neighbors (SNN) proposed in [60]:

$$\mathcal{W}(\rho_i, \rho_j) = |\mathbb{N}(\rho_i) \cap \mathbb{N}(\rho_j)| \tag{4.1}$$

where ρ_i and ρ_j are two elements in the full-pairwise correlation matrix $\mathcal{A}$, and $\mathbb{N}(\rho_i)$ and $\mathbb{N}(\rho_j)$ are the SNN of ρ_i and ρ_j, so $\mathcal{W}(\rho_i, \rho_j)$ results in the number of $\mathcal{K}$-nearest neighbours shared by ρ_i and ρ_j, where $\mathcal{K}$ is a predefined parameter. Then, we apply clustering to the resulted matrix $\mathcal{W}$ from SNN.

Smartphone identification deals with 1-to-m matching problem and determines which smartphone out of m took a given image. So, the stopping criterion in the hierarchical clustering and the parameter k in k-means and k-medoids are

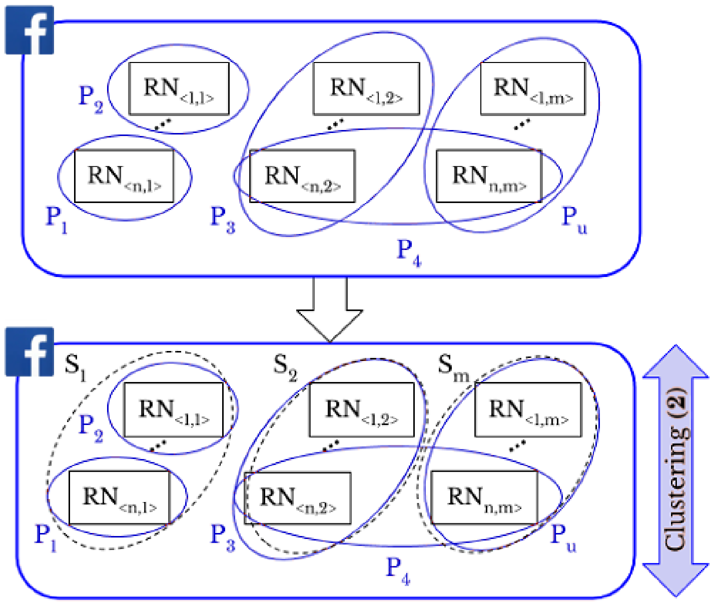


Figure 4.6: *Intra-layer UPL* task based on clustering: profiles P_1 and P_2 are linked since they share images taken from the same smartphone S_1.

set to the number of smartphones. All the clustering techniques label each RN with a cluster, representing the related source of the image.

Similarly, we address the *intra-layer UPL* task, as shown in Figure 4.6. Let $\mathcal{D}^x = \{d_1, d_2, ..., d_N\}$ be a set of images where $x \in$ {G, W, FH, FL, T}, corresponding to social networks *Google+* (G), *WhatsApp* (W), *Facebook High Resolution* (FH), *Facebook Low Resolution* (FL) and *Telegram* (T). Each image d_i in $\mathcal{D}^x$ has a specific profile tag P_i that represents the i^{th} user's profile on the social network x the image comes from. Like *original-by-original SI*, we exploit the full-pairwise correlation matrix of the extracted RNs to cluster $\mathcal{D}^x$ images into the right sources of S. Then, by using the resulted clusters and profile tags, we are able to link profiles. Moreover, we can determine whether a user uploaded images taken by one or more smartphones. In the first case, if within two different profiles there are images that are in the same cluster S_i, these profiles could be linked. For instance, in Figure 4.6, identification of smartphone

S_1 leads to a matching between the profiles P_1 and P_2. In the second case, if the images belonging to the same profile are grouped in different clusters, it means that the user uploaded the images from different smartphones. In Figure 4.6, the user of profile P_4 has shared images taken by two different smartphones, namely S_2 and S_m.

4.3.3 Classification-based SI and inter-layer UPL

Here, we exploit the obtained clusters, from *original-by-original SI* and *intra-layer UPL* tasks, as ground truths of the fingerprinted smartphones to classify "native" or "shared images" into m classes. Generally, ANNs, inspired by the biological form of the human neural system, have proven their effectiveness in classification tasks [102]. They are very flexible in learning features and can solve non-linear problems. Compared with the other classifiers such as support vector machine, extreme learning machine, and random forest, ANNs are more fault tolerant [103]. As a mathematical model, an ANN consists of a set of attached neurons called processing units. Neurons are organized in layers. The output of a neuron is stated as $f(\mathfrak{h})$, where $f()$ is the *activation function*, and $\mathfrak{h}$ is computed as follows:

$$\mathfrak{h} = \sum_{i=1}^{s} \mathfrak{w}_i \mathfrak{x} + \mathfrak{b} \tag{4.2}$$

$\mathfrak{x}_i$ and $\mathfrak{w}_i$ are the input and weight of the neuron, respectively; $\mathfrak{b}$ is the bias; and s is the total number of input connections of the neuron [104]. For a desirable classification, the weights of the ANN should be tuned. This process is called *training* or *learning* [105]. A Multi-Layer Perceptron (MLP) is a kind of ANN composed of one or several hidden layers of neurons [106]. An MLP is trained by using a Back Propagation (BP) algorithm such that it minimizes the

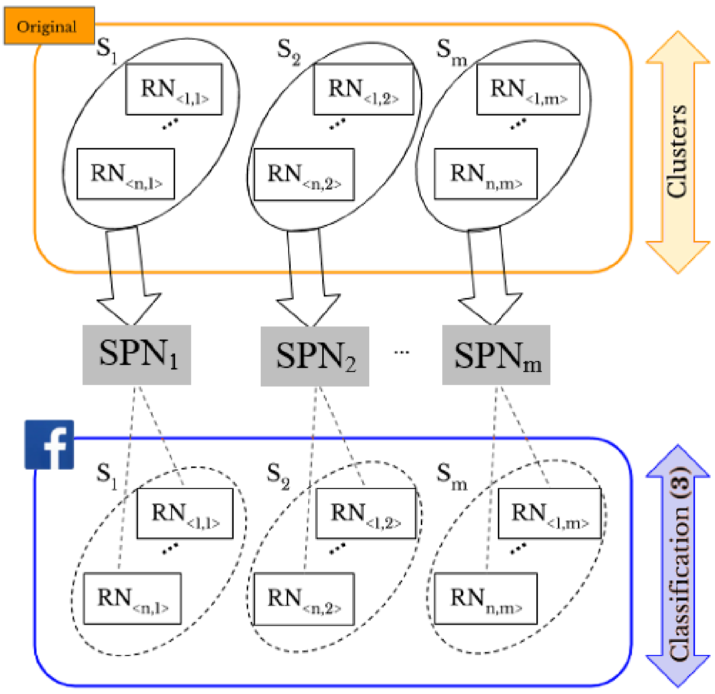


Figure 4.7: *Social-by-original SI* task based on classification: the clustered "native images" are used to train the ANN and classify the "shared images".

Mean Squared Error (MSE), which is formulated by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\mathfrak{T}_i - \mathfrak{O}_i)^2 \tag{4.3}$$

where $\mathfrak{O}$ and $\mathfrak{T}$ are matrices representing the labels predicted by ANN and the class labels of the inputs, respectively, and N is the number of samples in $\mathcal{D}^x$. We will use the clustered images that are the outcome of the previous task and ANN to perform both *social-by-original SI* and *inter-layer UPL*.

The *social-by-original SI* task is shown in Figure 4.7. We first define the fingerprint SPN$_i$ corresponding to the obtained clusters from the set I, such that SPN$_i$ transitively identifies the smartphone S_i. Then, by Equation (2.3), we calculate the correlation values between each pair of RNs extracted from the images in $\mathcal{D}^x$, and the obtained SPNs. For example, a correlation matrix of

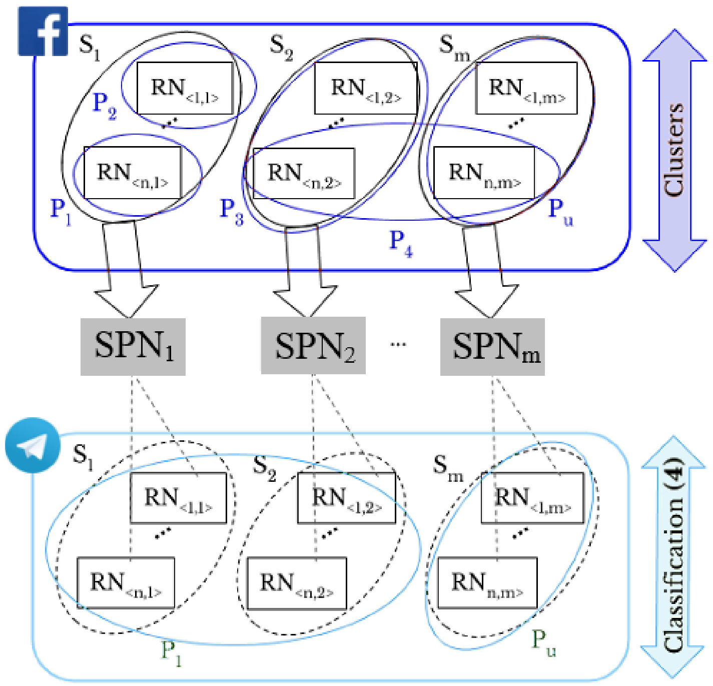


Figure 4.8: *Inter-layer UPL* task based on classification: to classify the "shared images" on a given social network (e.g., *WhatsApp*), the ANN is trained by using the obtained clusters of "shared images" on a different social network (e.g., *Google+*).

the size 900×18 is formed corresponding to 900 RNs in $\mathcal{L}^{\mathrm{G}}$ to be classified according to 18 smartphones in $\mathcal{L}^{\mathrm{NA}}$ which have already been identified in the clustering. The matrix is used for training and test the ANN through a 10-folds cross-validation model [107]. In particular, in each 10 iterations, the ANN is given 90% of the rows in the correlation matrix and corresponding class labels (smartphone labels by which the RNs in $\mathcal{L}^{\mathrm{G}}$ were generated) as the ground truth. In the test, the trained ANN is provided by 10% of the rows in the correlation matrix to classify each image in $\mathcal{L}^{\mathrm{G}}$, called *social-by-original SI*. By using the 10-fold technique, all the samples in the correlation matrix are tested as there is a swap between training and test in each iteration.

In *inter-layer UPL* task, as shown in Figure 4.8, the profile tag P_i, where i represents the i^{th} profile on a given social network, allows one to link user profiles across different social networks. The SPN_i is defined by using the clusters obtained from *Google+*, and the ANN is trained to classify the *WhatsApp* images. After the classification, the profile P_1 on *WhatsApp* is linked to the profiles P_1, P_2 and P_3 on *Google+* because they share images taken from the same smartphones S_1 and S_2. Similarly, the profile P_u on *WhatsApp* is linked to the profiles P_4 and P_u on *Google+*.

4.4 Experimental Setting and Results

In this section, some experimental setting for the proposed methods, i.e., SI and UPL, is performed and the results of the methods on different datasets are presented. To evaluate the proposed methods, we gathered a dataset that consists of 4500 "native images" captured by 18 different smartphones and call it "Lab Dataset", i.e., $\mathcal{L}$. The dataset was uploaded and downloaded on 4 of the most popular social networks, namely *Google+*, *Facebook* (high resolution), *WhatsApp*, and *Telegram*[1]. Corresponding to the "native" (NA) and "shared images", we have the datasets $\mathcal{L}^{\mathrm{NA}}$, $\mathcal{L}^{\mathrm{G}}$, $\mathcal{L}^{\mathrm{W}}$, $\mathcal{L}^{\mathrm{FH}}$, and $\mathcal{L}^{\mathrm{T}}$. The characteristics of the applied smartphones in $\mathcal{L}$ are listed in Table 4.1.

Also, we validate our proposed methods by the VISION image dataset [95], shown in Table 4.2, composed of images in both *native* format and *shared* version. More specifically, the dataset contains *flat* and *generic* images taken by 35 smartphones. The former is a set of images of walls and skies, while the latter is a set of images without limitations on orientation or scenario.

The images were shared through *WhatsApp* and *Facebook* (in both high and low resolutions). We use only the *generic* images in our experiments. We call

[1]The entire dataset including shared images on 16 social networks is available from: `http://smartdata.cs.unibo.it/datasets#images`

Phone ID	Brand	Model	Resolution	#images
S_1	LG	Nexus 4	3264×2448	50
S_2	Samsung	Galaxy S2	3264×2448	50
S_3	Apple	iPhone 6+	3264×2448	50
S_4	LG	Nexus 5	3264×2448	50
S_5	Huawei	Y550	2592×1944	50
S_6	Apple	iPhone 5	3264×2448	50
S_7	Motorola	Moto G	2592×1456	50
S_8	Samsung	Galaxy S4	4128×3096	50
S_9	LG	G3	4160×3120	50
S_{10}	LG	Nexus 5	3264×2448	50
S_{11}	Sony	Xperia Z3	5248×3936	50
S_{12}	Samsung	Samsung S3	3264×2448	50
S_{13}	HTC	One S	3264×2448	50
S_{14}	LG	Nexus 5	3264×2448	50
S_{15}	Apple	iPhone 6	3264×2448	50
S_{16}	Samsung	Galaxy S2	3264×2448	50
S_{17}	Nokia	Lumia 625	2592×1456	50
S_{18}	Apple	iPhone 5S	3264×2448	50

Table 4.1: Smartphone's characteristics in Dataset in $\mathcal{L}$.

the datasets $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$, and $\mathcal{V}^{\mathrm{FL}}$ corresponding to the *native*, *WhatsApp* and *Facebook* with high and low resolutions images. In Table 4.3, the lowest and the highest resolutions of images in both datasets for each social network are presented.

In the following subsections, the results of *original-by-original SI*, *social-by-original SI*, *intra-layer UPL*, and *inter-layer UPL* are presented, respectively.

4.4.1 Original-by-original SI results

In this experiment, we use "native images" to identify their related smartphones, which is called the *original-by-original SI* task. As shown in Table 4.3, these images have a high resolution, so the results can be considered as a benchmark for the capability of the clustering in the best case. Also, we exploit this experiment to select a specific resolution for RNs cropping or resizing. In particular, in

ID	Brand	Model	Resolution	#images
S_1	Apple	iPhone 4S	3264×2448	178
S_2	Apple	iPhone 4S	3264×2448	200
S_3	Apple	iPhone 5	3264×2448	203
S_4	Apple	iPhone 5	3264×2448	223
S_5	Apple	iPhone 5c	3264×2448	201
S_6	Apple	iPhone 5c	3264×2448	206
S_7	Apple	iPhone 5c	3264×2448	333
S_8	Apple	iPhone 6	3264×2448	129
S_9	Apple	iPhone 6	3264×2448	227
S_{10}	Samsung	Galaxy S III Mini GT-I8190	2560×1920	150
S_{11}	Samsung	Galaxy S III Mini GT-I8190N	2560×1920	200
S_{12}	Apple	iPad2	960×720	170
S_{13}	Apple	iPad mini G	2592×1936	157
S_{14}	Apple	iPhone 4	2592×1936	217
S_{15}	Apple	iPhone 6 Plus	3264×2448	256
S_{16}	Asus	Zenfone	3264×1836	208
S_{17}	Huawei	Ascend G6-U10	3264×2448	153
S_{18}	Huawei	Honor 5C	4160×3120	271
S_{19}	Huawei	P8 GRA-L09	4160×2336	265
S_{20}	Huawei	P9 EVA-L09	3968×2976	237
S_{21}	Huawei	P9 Lite VNS-L31	4160×3120	234
S_{22}	Lenovo	P70-A	4784×2704	216
S_{23}	LG	D290	3264×2448	224
S_{24}	Microsoft	Lumia 640 LTE	3264×2448	180
S_{25}	OnePlus	A3000	4640×3480	284
S_{26}	OnePlus	A3003	4640×3480	236
S_{27}	Samsung	Galaxy S3 GT-I9300	3264×2448	207
S_{28}	Samsung	Galaxy S4 Mini GT-I9195	3264×1836	208
S_{29}	Samsung	Galaxy S5 SM-G900F	5312×2988	254
S_{30}	Samsung	Galaxy Tab 3 GT-P5210	2048×1536	166
S_{31}	Samsung	Galaxy Tab A SM-T555	2592×1944	154
S_{32}	Samsung	Galaxy Trend Plus GT-S7580	2560×1920	163
S_{33}	Sony	Xperia Z1 Compact D5503	5248×3936	216
S_{34}	Wiko	Ridge 4G	3264×2448	249
S_{35}	Xiaomi	Redmi Note 3	4608×2592	305

Table 4.2: Smartphone's characteristics in Dataset $\mathcal{V}$.

the setting, we use the k-medoids method, where k is set to the number of smartphones, i.e., $k = 18$ and $k = 35$ for the datasets $\mathcal{L}$ and $\mathcal{V}$, respectively, see Tables 4.1 and 4.2.

[1]The highest resolution for cropping RNs is 960×544 px corresponding to the highest image resolutions in $\mathcal{L}^{\text{NA}}$ in Table 4.3.

Dataset	Lowest resolution	Highest resolution
$\mathcal{L}^{\mathrm{NA}}$	960×544	5248×3936
$\mathcal{L}^{\mathrm{G}}$	960×544	5248×3936
$\mathcal{L}^{\mathrm{W}}$	960×544	1600×1200
$\mathcal{L}^{\mathrm{FH}}$	960×544	2048×1536
$\mathcal{L}^{\mathrm{T}}$	960×544	1280×960
$\mathcal{V}^{\mathrm{NA}}$	960×720	5248×3936
$\mathcal{V}^{\mathrm{W}}$	960×720	1280×960
$\mathcal{V}^{\mathrm{FH}}$	960×720	2048×1536
$\mathcal{V}^{\mathrm{FL}}$	1040×584	1312×984

Table 4.3: The lowest and highest image resolution in different datasets.

Size	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
960×544[1]	0.89	0.91	0.90	0.824	0.89	0.95	0.00
512×512	0.79	0.85	0.82	0.692	0.81	0.89	0.02
256×256	0.75	0.76	0.76	0.598	0.74	0.87	0.02
128×128	0.42	0.44	0.43	0.208	0.39	0.66	0.04

Table 4.4: Results (%) of *original-by-original SI* on $\mathcal{L}^{\mathrm{NA}}$, by cropping the RNs with different image resolution.

Size	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
1536×1536	0.85	**0.91**	0.88	0.787	0.88	0.95	**0.00**
1280×1024	**0.90**	0.89	0.86	0.819	0.85	0.94	**0.00**
1024×1024	**0.90**	**0.91**	**0.91**	**0.834**	**0.90**	**0.96**	**0.00**
960×544	0.86	0.90	0.88	0.789	0.87	0.95	**0.00**
512×512	0.86	0.90	0.88	0.789	0.87	0.94	0.01
256×256	0.55	0.58	0.57	0.348	0.55	0.75	0.03
128×128	0.16	0.18	0.17	0.034	0.13	0.38	0.06

Table 4.5: Results (%) of *original-by-original SI* on $\mathcal{L}^{\mathrm{NA}}$, by resizing the RNs with different image resolution.

To select a specific resolution resulting in the best quality of the clustering, we consider a variety of image resolutions. For example, for the dataset $\mathcal{L}^{\mathrm{NA}}$, we consider the resolutions 128×128, 256×256, 512×512, 960×544, 1024×1024, and 1280×1024 and 1536×1536 px. While for cropping, we crop each image from the center to the resolutions including 128×128, 256×256, 512×512, and 960×544 px, which is the lowest resolution of the *native* images in the dataset,

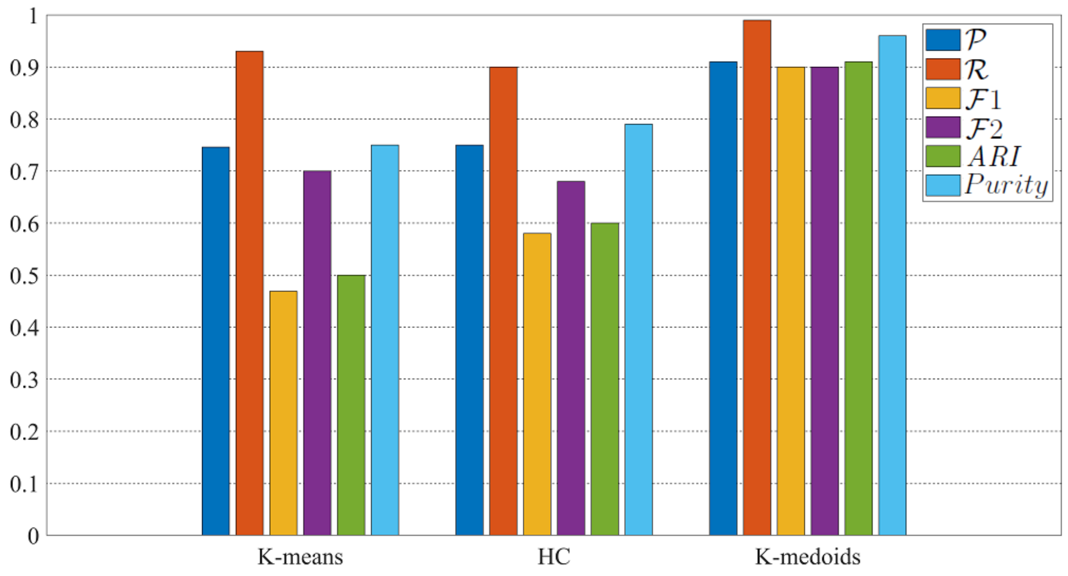


Figure 4.9: Results (%) of original-by-original SI by using different clustering methods on $\mathcal{L}^{\mathrm{NA}}$.

see Table 4.3. Based on the obtained results shown in Tables 4.4 and 4.5, it can be seen that resizing the RNs to the size 1024×1024 results in the best values of all the measures, i.e., $\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}1$, $\mathcal{F}2$, ARI, $Purity$ and FPR compared with the other resolutions. The best value of each measure is highlighted in bold, in Tables 4.4 and 4.5. Hence, in the following experiments, RNs are resized to the resolution 1024×1024 for both datasets $\mathcal{L}^{\mathrm{NA}}$ and $\mathcal{V}^{\mathrm{NA}}$. The comparison among k-means, hierarchical clustering and k-medoids techniques applied to $\mathcal{L}^{\mathrm{NA}}$ is shown in Figure 4.9. The results confirm that the k-medoids is the best for clustering RNs, among the other basic clustering techniques, even for clustering the images from identical models of smartphones.

Figure 4.10 shows the impact of SNN on the pairwise correlation matrices of the datasets $\mathcal{L}^{\mathrm{NA}}$ and $\mathcal{V}^{\mathrm{NA}}$. Comparing the sub-figures (c) and (d) with (a) and (b), it can be seen that the average of intra-camera correlations, shown in the diagonal parts, has increased, while the average of the inter-camera correlations has decreased. This improvement in the correlations between RNs produces better results in the k-medoids clustering. The value of $\mathcal{K}$ in SNN for each dataset

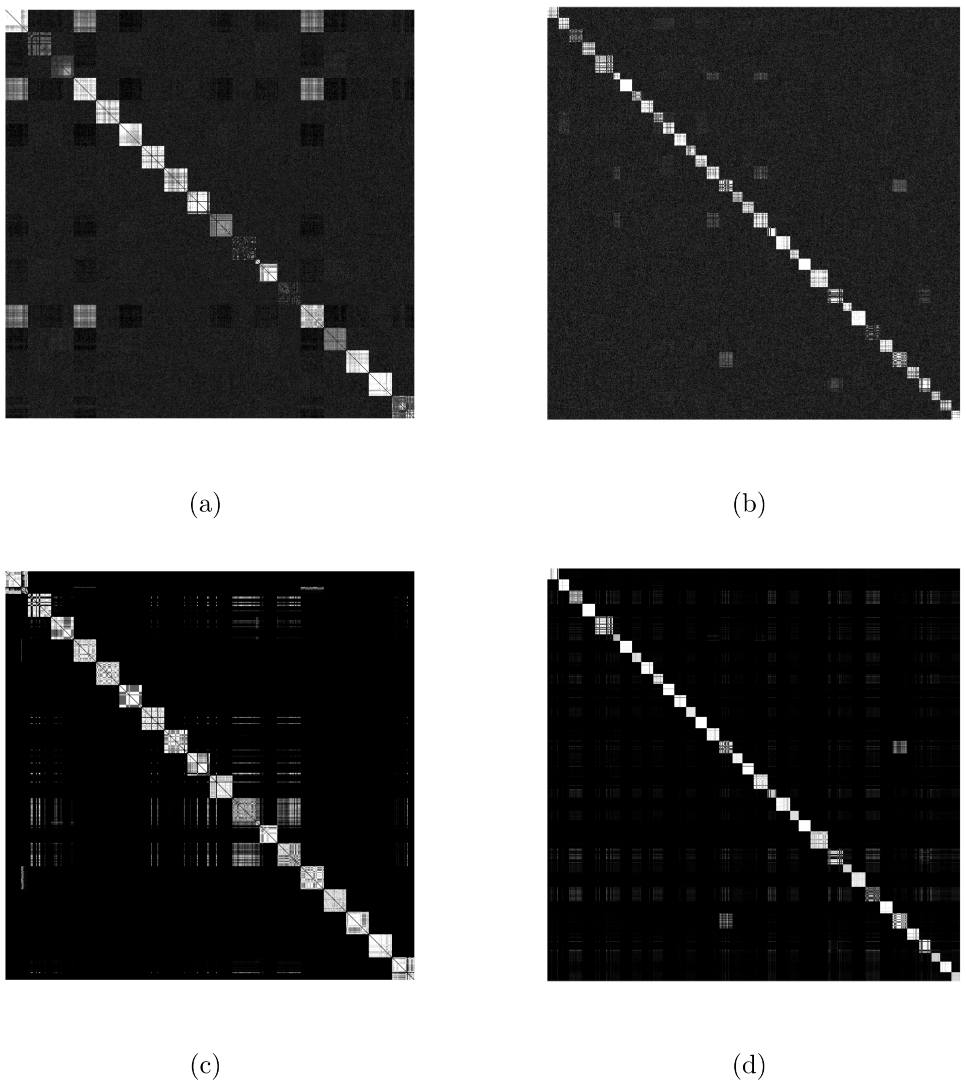


Figure 4.10: Pairwise similarities of RNs: (a) and (b) without and (c) and (d) with using shared $\mathcal{K}$-nearest neighbour, respectively from left to right for $\mathcal{L}^{\text{NA}}$, $\mathcal{K}$=20, and $\mathcal{V}^{\text{NA}}$, $\mathcal{K}$=70.

was experimentally determined. Different values were tested and $\mathcal{K} = 20$ and $\mathcal{K} = 70$ generated the best results in the clustering for $\mathcal{L}^{\text{NA}}$ and $\mathcal{V}^{\text{NA}}$, respectively.

Table 4.6, presents the results of *original-by-original SI* on different datasets based on k-medoids clustering.

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
$\mathcal{L}^{\mathrm{NA}}$	0.901	0.911	0.915	0.835	0.900	0.964	0.000
$\mathcal{V}^{\mathrm{NA}}$	0.827	0.834	0.831	0.713	0.825	0.894	0.005

Table 4.6: Results (%) of *original-by-original SI* on different datasets.

4.4.2 Social-by-original SI results

In this test, we use both "native" and "shared images" to present *social-by-original SI*. We apply the obtained clusters of $\mathcal{L}^{\mathrm{NA}}$ from the previous test to classify the shared images on social networks based on ANNs.

As described before, we evaluate the effectiveness of the ANN in the classification stage as well as its generalization capability by using 10-fold cross-validation. Firstly, a matrix including the correlations between the extracted RNs and the obtained SPNs are calculated based on Equation (2.3). The i^{th} row of the matrix includes the similarities between the i^{th} RN and all the resulted SPNs from the clustering. The rows related to the same smartphone are shuffled to have an order-independent evaluation. Then, they are divided into 10 folds so that each of them includes an equal number of samples for each smartphone. In each of 10 iterations of the cross-validation, nine folds and one independent fold are used respectively for "training set" and "test set". For example, in $\mathcal{L}^{\mathrm{NA}}$ we have 50 images for each smartphone, so we use 850 and 50 rows, respectively, in training and test at each iteration. The 10-fold cross-validation process is repeated 10 times by swapping between training and test samples. Finally, the average values obtained from the measures in (2.4)-(2.11) for all the iterations are considered as the ANN results.

To set up the architecture of the applied ANNs, we use the obtained clusters of the $\mathcal{L}^{\mathrm{NA}}$ to classify the images in $\mathcal{L}^{\mathrm{G}}$, as *Google+* images provide the highest resolution. Accordingly, the test could also be considered as a benchmark for the ANNs used for the other social networks. We tested different topology in terms of training method, activation function, and number of hidden layers. As a result, an appropriate effectiveness of *social-by-original SI* and *inter-layer UPL* is achieved by the simple ANN's architecture shown in Table 4.7. In particular, we use *trainscg* as the training function that updates weight and bias values based on the *scaled conjugate gradient training* algorithm, and the *logistic sigmoid* as activation function that provides an appropriate convergence in the training. In particular, the applied activation function is defined as follows:

$$f(\mathfrak{h}) = \frac{1}{1+\mathfrak{e}^{-\mathfrak{h}}} \tag{4.4}$$

where $\mathfrak{h}$ is obtained by (4.2).

Type	Multi-Layer Perceptron (MLP)
Number of layers	2
Neurons in input layer	for $\mathcal{L}$, 850 (training) and 50 (test); for $\mathcal{V}$, 6732 (training) and 748 (test)
Neurons in hidden layer	35
Neurons in output layer	18 for $\mathcal{L}$; 35 for $\mathcal{V}$
Learning rule	Back Propagation (BP)
Training function	trainscg
Activation function	logsig
Error	Mean Squared Error (MSE)

Table 4.7: ANN's architecture.

Based on Figure 4.11, by systematically increasing the number of neurons, the classification results are improved in terms of $\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}1$, $\mathcal{F}2$, *ARI* and *Purity*. After the cardinality of 5, the trend of the results is almost plateau. However, to be sure to get the best results for all the datasets, we consider the cardinality 35 resulting in the highest values in the plateau part of the graph. The tuning

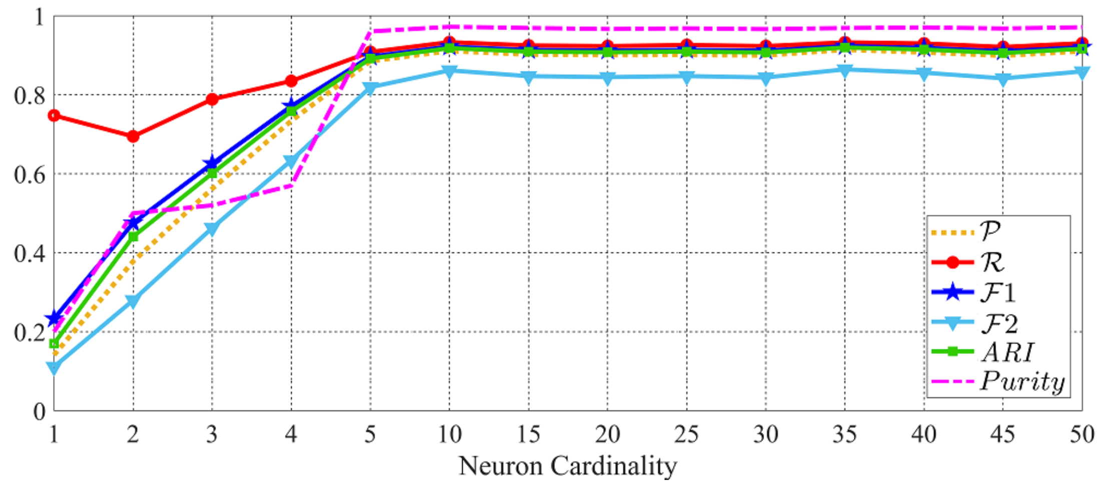


Figure 4.11: Results (%) of *social-by-original SI* for systematically increasing the number of neurons in the hidden layer of ANN. Images in $\mathcal{L}^{\text{G}}$ are classified by the obtained clusters of images in $\mathcal{L}^{\text{NA}}$ and the trained ANN.

phase of the ANNs can also be used as a benchmark for the capability of the classification in the best case because the "native images" and *Google+* images have the highest resolution in the dataset. The results of *social-by-original SI* for both datasets $\mathcal{L}$ and $\mathcal{V}$ are shown in Table 4.8. The *social-by-original SI* enables identification of smartphones in spite of the fact that the pictures get downgraded during the uploading and downloading process.

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR
$\mathcal{L}^{\text{G}} - \mathcal{L}^{\text{NA}}$	0.933	0.954	0.943	0.898	0.940	0.980	0.003
$\mathcal{L}^{\text{W}} - \mathcal{L}^{\text{NA}}$	0.815	0.853	0.833	0.716	0.825	0.923	0.009
$\mathcal{L}^{\text{FH}} - \mathcal{L}^{\text{NA}}$	0.833	0.859	0.846	0.736	0.839	0.926	0.008
$\mathcal{L}^{\text{T}} - \mathcal{L}^{\text{NA}}$	0.828	0.866	0.846	0.736	0.839	0.930	0.008
$\mathcal{V}^{\text{W}} - \mathcal{V}^{\text{NA}}$	0.882	0.836	0.829	0.762	0.824	0.910	0.005
$\mathcal{V}^{\text{FH}} - \mathcal{V}^{\text{NA}}$	0.795	0.821	0.808	0.676	0.802	0.900	0.006
$\mathcal{V}^{\text{FL}} - \mathcal{L}^{\text{NA}}$	0.730	0.774	0.752	0.591	0.744	0.883	0.008

Table 4.8: Results (%) of *social-by-original SI* on different datasets.

4.4.3 Intra-layer UPL results

In this section, we discuss the results of *intra-layer UPL*. In particular, this test exploits "shared images" to determine whether a given set of user profiles within the same social network are linked. Table 4.9 shows the results on the "shared images" in both $\mathcal{L}$ and $\mathcal{V}$. The best results are related to $\mathcal{L}^{\mathrm{G}}$. The reason is that *Google+* images have the same resolution as the "native images" confirming that the compression algorithm on this social network results in less elimination of image details, (see Table 4.3). Although the other social networks compress the images more than *Google+*, the method has returned good results confirming the effectiveness of the method in the task of *intra-layer UPL*.

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
$\mathcal{L}^{\mathrm{G}}$	0.884	0.897	0.890	0.809	0.884	0.942	0.006
$\mathcal{L}^{\mathrm{W}}$	0.845	0.878	0.861	0.760	0.853	0.938	0.009
$\mathcal{L}^{\mathrm{FH}}$	0.848	0.862	0.855	0.751	0.846	0.921	0.008
$\mathcal{L}^{\mathrm{T}}$	0.848	0.862	0.855	0.751	0.846	0.921	0.008
$\mathcal{V}^{\mathrm{W}}$	0.742	0.751	0.746	0.586	0.738	0.839	0.007
$\mathcal{V}^{\mathrm{FH}}$	0.700	0.729	0.714	0.539	0.705	0.793	0.009
$\mathcal{V}^{\mathrm{FL}}$	0.412	0.424	0.418	0.197	0.400	0.573	0.018

Table 4.9: Results (%) of *intra-layer UPL* on different datasets.

4.4.4 Inter-layer UPL results

This last test is the most challenging. We demonstrate that the proposed method is able to link a restricted set of user profiles across different social networks. In other words, we verify whether two sets of images from different user profiles on different social networks are linked, that is *inter-layer UPL*. The strengths of our method include the possibility to exploit images from different social networks, not only the "native images", but also the robustness in spite of the fact that some social networks degrade the resolution of the images more than others. We consider all the different combinations of the selected social networks for each dataset, as shown in Figure 4.2. The results for all the possible pairs of social networks are presented in Tables 4.10 and 4.11.

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
$\mathcal{L}^{\mathrm{W}} - \mathcal{L}^{\mathrm{G}}$	0.890	0.911	0.900	0.825	0.896	0.005	0.958
$\mathcal{L}^{\mathrm{FH}} - \mathcal{L}^{\mathrm{G}}$	0.880	0.905	0.892	0.811	0.887	0.958	0.005
$\mathcal{L}^{\mathrm{T}} - \mathcal{L}^{\mathrm{G}}$	0.887	0.909	0.898	0.821	0.893	0.005	0.957
$\mathcal{L}^{\mathrm{G}} - \mathcal{L}^{\mathrm{W}}$	0.871	0.900	0.885	0.799	0.880	0.959	0.006
$\mathcal{L}^{\mathrm{FH}} - \mathcal{L}^{\mathrm{W}}$	0.815	0.860	0.836	0.721	0.829	0.941	0.009
$\mathcal{L}^{\mathrm{T}} - \mathcal{L}^{\mathrm{W}}$	0.839	0.872	0.855	0.750	0.848	0.942	0.007
$\mathcal{L}^{\mathrm{G}} - \mathcal{L}^{\mathrm{FH}}$	0.864	0.897	0.880	0.791	0.875	0.953	0.006
$\mathcal{L}^{\mathrm{W}} - \mathcal{L}^{\mathrm{FH}}$	0.815	0.864	0.838	0.723	0.830	0.941	0.009
$\mathcal{L}^{\mathrm{T}} - \mathcal{L}^{\mathrm{FH}}$	0.854	0.885	0.869	0.773	0.863	0.946	0.007
$\mathcal{L}^{\mathrm{G}} - \mathcal{L}^{\mathrm{T}}$	0.900	0.921	0.910	0.842	0.906	0.965	0.004
$\mathcal{L}^{\mathrm{W}} - \mathcal{L}^{\mathrm{T}}$	0.88	0.902	0.891	0.809	0.886	0.953	0.005
$\mathcal{L}^{\mathrm{FH}} - \mathcal{L}^{\mathrm{T}}$	0.822	0.881	0.850	0.741	0.843	0.944	0.009

Table 4.10: Results (%) of *inter-layer UPL* on $\mathcal{L}$.

It is worth mentioning that the images in $\mathcal{L}$ used for experiments of *inter-layer UPL* on different social networks are not from the same scenes, making more similar real-life situation. As it is shown in Table 4.10, using *Google+* images to classify the images on the other social network datasets, i.e., $\mathcal{L}^{\mathrm{W}}$, $\mathcal{L}^{\mathrm{FH}}$ and $\mathcal{L}^{\mathrm{T}}$ produces the highest values of $\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}1$, $\mathcal{F}2$, *ARI*, *Purity*, and *FPR*, as shown in the first rows of Table 4.10. From the results in Table 4.11, it is implied that using images in $\mathcal{V}^{\mathrm{W}}$ to classify the images in the other datasets i.e., $\mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}^{\mathrm{FL}}$ obtained the best results. It is interesting that the classification of the images in $\mathcal{V}^{\mathrm{FL}}$ in inter-layer UPL compared with the clustering the images in intra-layer UPL generates better results. Given the results, it is proven that the proposed methods are reliable enough to link user profiles on the selected social networks.

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*
$\mathcal{V}^{\mathrm{FH}} - \mathcal{V}^{\mathrm{W}}$	0.755	0.785	0.772	0.619	0.762	0.878	0.007
$\mathcal{V}^{\mathrm{FL}} - \mathcal{V}^{\mathrm{W}}$	0.755	0.782	0.775	0.617	0.760	0.878	0.007
$\mathcal{V}^{\mathrm{W}} - \mathcal{V}^{\mathrm{FH}}$	0.755	0.781	0.772	0.616	0.761	0.877	0.007
$\mathcal{V}^{\mathrm{FL}} - \mathcal{V}^{\mathrm{FH}}$	0.754	0.776	0.760	0.612	0.756	0.871	0.007
$\mathcal{V}^{\mathrm{W}} - \mathcal{V}^{\mathrm{FL}}$	0.589	0.610	0.591	0.389	0.582	0.723	0.013
$\mathcal{V}^{\mathrm{FH}} - \mathcal{V}^{\mathrm{FL}}$	0.585	0.611	0.600	0.387	0.586	0.736	0.012

Table 4.11: Results (%) of *inter-layer UPL* on $\mathcal{V}$.

4.5 Conclusion

In this Chapter, we have proposed clustering and classification based methods to achieve SI and UPL. The methods can help to detect evidence references in data analysis, when a set of images captured by a given number of smartphones and shared on a set of user profiles are provided. We have evaluated our methods on different datasets, i.e., our dataset and VISON dataset. The results confirm the effectiveness of the methods, even with the same models of smartphones. The methods are applicable to images compressed on social networks, and there is no need to hack user's smartphone for fingerprinting. An important outcome of our work is presenting the *inter-layer UPL* task, which is more desirable in shared image analysis because it links user profiles on different social networks. The methods will become even more powerful when considering other types of information such as GPS, users' e-mail addresses, and login information, to name a few. Through the methods, the number of the smartphones has to be provided. In the following Chapter, we propose SPN-based image clustering methods for clustering different types of shared images, whether images taken and shared by user smartphones or images taken from the other sources like the Web, without prior knowledge of the number of the camera sources generated the images.

CHAPTER 5

Shared Image Clustering and User Profile Linking

The traditional clustering algorithms like k-means, k-medoids and hierarchical clustering has to be provided by the initial information about the number of camera sources. In the cases without the initial information, usually combining different clustering algorithms is more effective. In this Chapter, firstly, we present a **H**ybrid M**a**rkov C**l**ustering (HAL) method capable of clustering the images captured and shared by users on social networks, without prior knowledge about the types and number of the camera sources. The HAL method exploits batch partitioning, image resizing, hierarchical and graph-based clustering techniques to cluster the images. Using Markov clustering, the hierarchical clustering is conducted in such a way that the representative clusters with a higher probability of belonging to the same camera are selected for merging, which accelerates the clustering. For merging the clusters, an adaptive threshold, which is updated iteratively through the clustering process, is used, resulting in more precise clusters even for images from identical smartphones.

The shared images by users can be categorized into two groups. The first one is a set of images representing the SPN characteristics of their camera sources sufficiently and can be applied to the smartphone camera fingerprinting. The second one is a set cropped images, images from the Web, or single images from different sources, not representing the SPN characteristics of their camera sources sufficiently. So, they cannot be applied to smartphone camera fingerprinting. We consider the second group as outliers and propose **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO) method to detect and remove the outliers from the clustering process. Particularly, in the HALO method, the outlier detection is performed based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique.

The results on the VISION image dataset, including both "native" and "shared images", prove the effectiveness and efficiency of the HAL and HALO methods in comparison with the state-of-the-art SPN-based image clustering algorithms.

5.1 Problem Statement

For real-life applications dealing with a large number of images, the SPN-based image clustering algorithm needs to be precise, scalable and feasible. In other words, it is desired to cluster the shared images on user profiles into the right groups, to be applicable on any given dataset, and have a reasonable running time. Developing an algorithm meeting all these requirements has some challenges as follows:

- To extract the right SPN, the correct orientation of images has to be obtained. Some smartphone settings and social network platforms remove the orientation information from metadata of the image file, which makes the clustering task more challenging.
- The extracted RN from an image can be severely contaminated by other interference. Besides, the process of content compression performed by social networks causes loss of image details and weakens the SPN.

- To compute the similarity among the RNs, they have to have the same spatial resolution. The typical way is to crop the central block of RNs which may cause loss of critical data.
- Though it is desirable to apply high resolutions of RNs, for having better quality of clustering, the heavy overhead on data storage and computation costs their usage.
- Images captured by different smartphones of the same model undergo the same imaging pipeline which may introduce similar artifacts in the SPNs.
- Calculating the full-pairwise correlation matrix is a cumbersome and sometimes infeasible task especially for large-scale datasets.
- In a more challenging situation, the users may share images from various sources, not representing the SPN characteristics of their cameras. This results in the generation of a large number of singleton clusters which is not computationally cost effective.

With these challenges in the SPN-based image clustering, many works have been done as it was mentioned in Chapter 3. However, the literature lacks a precise, scalable, and feasible algorithm designed particularly for clustering the *shared images* on social network platforms.

5.2 Contribution

We present a **H**ybrid M**a**rkov C**l**ustering (HAL) method by combining the hierarchical and Markov clustering techniques. The HAL method is capable of clustering the images captured and shared through social networks without prior knowledge about the types and number of the related smartphones, and it tackles most of the mentioned challenges. This makes it applicable to the real-life cases:

- Unlike most studies presented in the literature, to get better characteristics of SPN, we exploit resizing rather than cropping the RNs, which is more useful in the clustering, especially for *shared images* having low resolutions.

- To decrease the computational costs in terms of memory usage, the dataset is partitioned into small batches, whose sizes fit the size of the available RAM.

- By using Markov clustering, the hierarchical clustering is conducted in such a way that the representative clusters with a higher probability of belonging to the same camera are selected for a merging.

- To merge the candidate clusters, an adaptive threshold is used. The threshold generally increases as the size and quality of a cluster increase. This prevents wrong merging of clusters, especially the clusters from the same models of smartphones.

- Partitioning the dataset, exploiting the inherent sparseness of the correlation matrix, and checking only the representative clusters in the merging result in the calculation of a small portion of the full-pairwise correlation matrix. This accelerates the clustering, which is particularly helpful for large-scale datasets.

- Considering the images, which do not sufficiently represent the SPN characteristics of their sources, as outliers, a developed version of the HAL method based on DBSCAN algorithm is presented, called **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO). The HALO method detects outliers and purifies each batch, resulting in more efficient and effective clustering.

5.3 Hybrid Markov Clustering Method

The HAL method mainly consists of preparation, hybrid clustering, and post-processing phases, as shown in Figure 5.1. We will look into each of the three

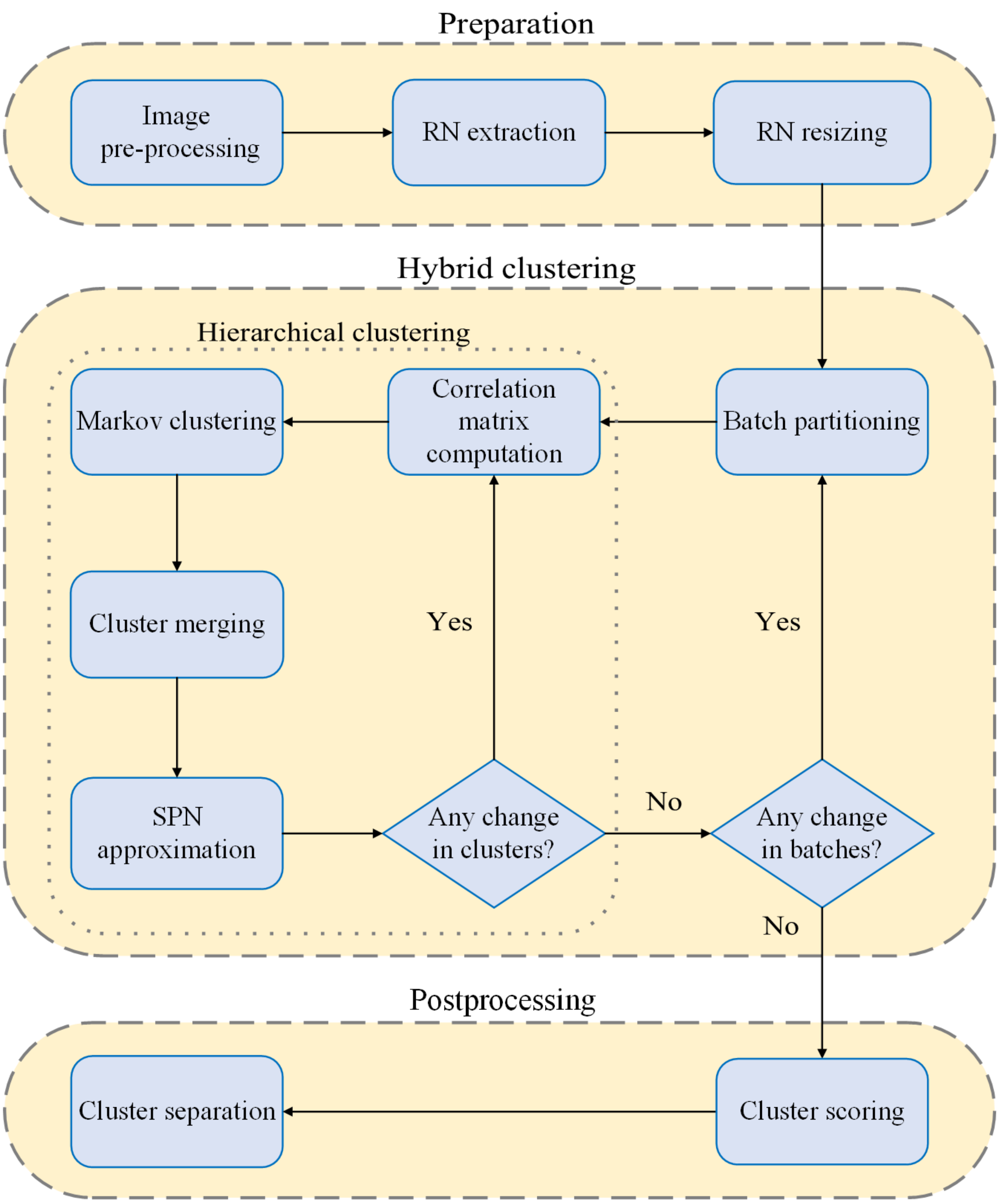


Figure 5.1: Flowchart of HAL method.

phases to give an overview of the proposed method:

1. **Preparation:** Firstly, images are aligned to the same orientation, depending on the availability of meta-data of the image files. Next, the grayscale version of the images is obtained and dark and saturated images are excluded. Then, RNs are extracted from the pre-processed images, and they are resized to a specific resolution, i.e., 1024×1024 px.

2. **Hybrid clustering:** the clustering starts with randomly partitioning the dataset into small batches. For each batch, a pairwise correlation matrix is calculated. Markov clustering algorithm is applied to the correlation matrix. By using the probability matrix, as the output of the Markov clustering, and nearest neighboring, the candidate clusters are selected, and subsequently an adaptive threshold is computed for merging the clusters. The SPN is updated for the merged clusters and similar process is hierarchically performed on the merged clusters. The clustering stops once no new cluster is found.

3. **Post-processing:** in this phase, the resulted clusters are scored based on their sizes, and the coarse clusters, i.e., the clusters with a notable number of RNs sharing the same SPN characteristics, are stored as the final result.

5.3.1 Preparation

The applied pre-processing step was mentioned previously, in Section 4.3. We apply similar pre-processing step, but there is only a difference in setting the orientation of images. For some images without EXIF data, we only rotate the images to either portrait or landscape orientation based on the spatial resolution

[95]. This may not completely resolve the orientation problem, but it can be alleviated. Anyhow, the clustering quality would be affected.

5.3.2 Hybrid clustering

The proposed hybrid clustering groups the RNs based on the combination of hierarchical and Markov clustering algorithms and an adaptive threshold which is iteratively generated according to the quality of the resulted clusters. First, we explain batch partitioning and how the cluster similarities are computed. Then, in the following subsections, we describe different steps in the hybrid clustering in the following subsections.

To reduce the use of RAM and to make the algorithm scalable, we follow the approach presented in [61]. Let N be the total number of RNs in the dataset. The pre-processed RNs are randomly partitioned into t batches, i.e., $B = \{b_1, b_2, ..., b_t\}$, where $t = \lceil \frac{N}{q} \rceil$ and q is the batch size. The parameter q is determined with respect to the available size of RAM. For each batch, a correlation matrix $\mathcal{A}$ is constructed, with each element $\mathcal{A}(i, j)$ being NCC similarity between any two SPNs in the batch, calculated by (2.3).

5.3.2.1 Hierarchical clustering

The clustering for each batch starts by considering each RN as a singleton cluster, and it is performed in an agglomerative hierarchical way by iteratively merging the similar clusters. At the end of each iteration of the hierarchical clustering, the camera fingerprints corresponding to the merged clusters are updated according to (2.2). Then, the obtained clusters from all the batches are grouped, and they are hierarchically partitioned and clustered until no new cluster is found, see

Figure 5.1. The hierarchical clustering has some drawbacks. Once a merging is done, this cannot be undone, and a wrong assignment may propagate the error to the following iterations in the clustering. Moreover, its computational burden is high because it has to investigate all the pairs of clusters for merging [98]. In our proposed clustering algorithm, we handle the mentioned drawbacks of the hierarchical clustering by combining it with the Markov clustering algorithm and a cluster merging step based on an adaptive threshold to achieve precise and fast clustering.

5.3.2.2 Markov clustering

Markov clustering is a fast and scalable unsupervised algorithm proposed in [108]. It has successfully been applied in different fields of science. It considers the objects as the vertices of a graph, e.g., Q, and groups them with respect to the weights of the edges, i.e., the similarities between the objects [109]. The Markov matrix $\mathcal{M}$ associated with the graph Q is defined by normalizing all the columns of the similarity matrix. A random walk is simulated over the vertices of the graph to increase and decrease the flow in strong and weak currents in Q, respectively [110]. More specifically, the random walk can be modeled as a Markov chain on the graph Q. Starting from a vertex, a random walk is more likely to arrive at the vertices within the same cluster than those in different clusters. The vertices of Q are considered as a set of states $S = \{s_1, s_2, ..., s_n\}$, and the graph edges are associated with the transition probabilities represented in $\mathcal{M} = [\mathfrak{p}(i,j)] \in \mathbb{R}^{n \times n}$, where each element at index (i,j) is the transition probability from vertex i to vertex j and

$$\sum_{i=1}^{n} \mathfrak{p}(i,j) = 1,\ 0 \leq \mathfrak{p}(i,j) \leq 1 \tag{5.1}$$

By alternatively applying expansion and inflation operators to $\mathcal{M}$, the clusters can be interpreted from the resulting transition matrix at the converged state. The expansion operator performed based on matrix multiplication simulates a random walk on the graph Q by:

$$\mathcal{M}_{exp} = \mathcal{M}^{e} \tag{5.2}$$

where e is the expansion parameter. The j^{th} column of $\mathcal{M}_{exp}$ can be interpreted as the probability distribution of the j^{th} of random walk.

Subsequently, the inflation operator is performed on each element of the matrix $\mathcal{M}_{exp}$ as follows:

$$\mathcal{M}_{inf}(i,j) = \frac{\mathcal{M}_{exp}(i,j)^{\eta}}{\sum_{k=1}^{n} \mathcal{M}_{exp}(k,j)^{\eta}} \tag{5.3}$$

By the inflation operator, the elements of the matrix $\mathcal{M}_{exp}$ are raised to the power of the inflation parameter η, and then the columns are normalized. In each column, the elements which have very small values (less than a predefined value ς) are removed, and the remaining elements are rescaled, to make the sum of each column equal to 1. This is called pruning defined as follows:

$$\mathcal{M}_{pru}(i,j) = \begin{cases} 0, & \mathcal{M}_{inf}(i,j) < \varsigma \\ \mathcal{M}_{inf}(i,j), & \text{otherwise} \end{cases} \tag{5.4}$$

The pruning reduces the number of non-zero elements in the matrix $\mathcal{M}_{inf}$, which decrease the memory usage and accelerates the clustering [111]. A global chaos $\mathcal{G}$ denotes the rate of the changes in the probability values in every two consecutive iterations. The algorithm stops when the global chaos approximately equals zero. The value of $\mathcal{G}$ is calculated based on the maximum value of chaos denoted as $\mathcal{C}_j$

on every column j of the matrix $\mathcal{M}_{pru}$ [112].

$$\mathcal{C}_j = \frac{\max\limits_{i=1,2,..,n} \mathcal{M}_{pru}(i,j)}{\sum_{i=1}^{n} \mathcal{M}_{pru}(i,j)^2} \tag{5.5}$$

$$\mathcal{G} = \max_{j=1,2,..,n} \mathcal{C}_j \tag{5.6}$$

The details of the Markov clustering as a part of the proposed HAL method are presented in Algorithm 1. The Markov clustering receives the matrix $\mathcal{A}$, computed by (2.3), including the similarities of the fingerprints in one batch, as an input, and it produces the matrix $\mathcal{M}$, such that each entry of the matrix represents the degree of the similarities between a pair of RNs in the batch. The addition of self-loops to the input matrix $\mathcal{A}$ prevents the dependency of the flow distribution on the length of the random walk, ensuring at least one

Algorithm 1: Markov clustering algorithm.

```
input: Pairwise correlation matrix, $\mathcal{A}$
output: Probabilities matrix, $\mathcal{M}$
- expansion parameter: $e$
- inflation parameter: $\eta$
- global chaos: $\mathcal{G}$
- prune parameter: $\varsigma$
- threshold for global chaos: $\xi$
- add self-loops to the graph $\mathcal{A}$, $\mathcal{A} = \mathcal{A} + \mathcal{I}$
- create the diagonal degree matrix of $\mathcal{A}$, $\mathcal{D}$
- create Markov matrix, $\mathcal{M} = \mathcal{A}\mathcal{D}^{-1}$
while $\mathcal{G} > \xi$ do
    - expansion on $\mathcal{M}$, based on (5.2)
    - inflation on $\mathcal{M}_{exp}$, based on (5.3)
    - pruning on $\mathcal{M}_{inf}$, based on (5.4)
    - update $\mathcal{G}$ based on (5.5) and (5.6)
    - $\mathcal{M}$=$\mathcal{M}_{pru}$
return $\mathcal{M}$
```

non-zero entry per column [111]. Given two clusters c_i and c_j corresponding to the vertices v_i and v_j in Q, if they share the same or very similar fingerprint characteristics of a camera, the element $\mathcal{M}(i,j)$ is assigned to a non-zero value. Otherwise, it means that the clusters are from different cameras and $\mathcal{M}(i,j)$ is set to 0.

During the hybrid clustering, in every iteration of the hierarchical clustering, the Markov clustering is performed on each batch, see Figure 5.1. By applying nearest neighboring to the columns of the resulted probability matrix, small cluster granularities can be obtained. We consider these representative and precise clusters as the candidate clusters, which are more likely to be from the same camera sources, and we merge them iteratively to discover larger clusters.

5.3.2.3 Cluster merging

Since the RNs were randomly partitioned into batches, the matrix $\mathcal{M}$ of a batch may contain many sparse columns, which are populated with many zero values. Hence, only the clusters corresponding to non-sparse columns are kept, and the remaining clusters are passed to the next iterations to get a better chance for a merging, as the clusters are being evolved. In the implementation, we consider a column as non-sparse if the number of its non-zero elements is less than 20. For each non-sparse column corresponding to the cluster c_i, its nearest neighbor cluster, i.e., c_j is found based on the highest probability value existing in the column. Then, the clusters c_i and c_j are selected as the candidate clusters for a merging. Usually, the RNs from the same model of smartphones present high correlations, and correspondingly high probabilities in $\mathcal{M}$ are produced. Accordingly, it is probable that they are selected as the candidate clusters. Therefore, to make the HAL method more precise, in the cluster merging, in addition to using the candidate clusters, we use an adaptive threshold.

The adaptive threshold is updated based on the quality of the obtained clusters in each iteration of the hierarchical clustering. It exploits the idea that the more images from a given smartphone are precisely clustered, the better the quality of SPN can be estimated [50]. As the cluster size grows, the inter-camera and intra-camera correlation distributions are normally more separable. Therefore, adaptively increasing the threshold can effectively prevent wrong merging of clusters, especially those from the same model of smartphones. The adaptive threshold $\mathcal{T}$ is defined as follows [61]:

$$\mathcal{T} = \max(\tau, \frac{\psi\sqrt{n_{c_i} n_{c_j} \mu_{c_i}^2 \mu_{c_j}^2}}{\sqrt{[(n_{c_i}-1)\mu_{c_i}^2+1][(n_{c_j}-1)\mu_{c_j}^2+1]}}) \tag{5.7}$$

where τ is a minimum threshold working as a trust boundary of $\mathcal{T}$. The terms n_{c_i} and n_{c_j} denote the number of the RNs in the two clusters c_i and c_j, respectively, and ψ is a predefined scaling factor. The quality of the cluster c_i, that is μ_{c_i}, is defined as the mean of the correlation values between all the pairs of RNs in the cluster. Given two candidate clusters c_i and c_j, if the correlation calculated by (2.3) between their corresponding fingerprints f_i and f_j is greater than the adaptive threshold, i.e., $\mathcal{A}(f_i, f_j) > \mathcal{T}$, the clusters are merged. Otherwise, they are kept separately and passed to the next iteration of the algorithm.

5.3.2.4 Computational complexity

Given the explanations above, we present the pseudo code of the HAL method in Algorithm 2. To compute the time complexity of the HAL algorithm, we first compute the complexity for one batch and then generalize it over all the t

Algorithm 2: Hybrid Markov Clustering.

```
input: pre-processed RNs
output: list of clusters, $C$
- number of RNs, $N$
- scaling factor, $\psi$ in (5.7)
- minimum threshold, $\tau$ in (5.7)
- size of batches, $q$
- clustering initialization, $C_{old} = \{\}$
- considering a set of single clusters corresponding to the RNs, $C_{new} = \{c_1, c_2, ..., c_N\}$
- initializing a set of camera fingerprints with the RNs corresponding to the clusters,
  $F = \{f_1, f_2, ..., f_N\}$
- partitioning initialization, $B_{old} = \{\}$
- $t = \lceil \frac{N}{q} \rceil$
- randomly partition $C_{new}$ into $t$ batches with size $q$, $B_{new} = \{b_1, b_2, ..., b_t\}$
while $|B_{new}| \neq |B_{old}|$ do
    for $k = 1 : t$ do
        while $|C_{new}| \neq |C_{old}|$ do
            - compute correlation matrix $\mathcal{A}$ by (2.3)
            - apply Markov clustering to $\mathcal{A}$ and generate the probability matrix $\mathcal{M}$ by
              Algorithm 1
            - put non-sparse column's indices in the list $L$
            for $i = 1 : |L|$ do
                - find the nearest cluster $c_j$ to the cluster $c_i$ from the list $L$
                - compute the adaptive threshold $\mathcal{T}$ by (5.7)
                if $\mathcal{A}(f_i, f_j) > \mathcal{T}$ then
                    - merge clusters $c_i$ and $c_j$
                else
                    - continue
            - put the obtained clusters in $C_{new}$
            - update the camera fingerprints in $F$ for the merged clusters by (2.2)
            - $C_{old} = C_{new}$
    - consider all the obtained clusters from batches as a new cluster $C_{new}$
    - $B_{old} = B_{new}$
    - $N = |C_{new}|$
    - update $t$, $t = \lceil \frac{N}{q} \rceil$
    - partition the clusters in $C_{new}$ into $t$ batches with size $q$, and form $B_{new}$
- $C = C_{new}$
return $C$
```

batches. For each batch, different steps such as correlation matrix computation, Markov clustering, finding non-sparse columns and nearest neighboring for cluster merging are applied. The correlation matrix computation for each batch with the size of q has the complexity $\mathcal{O}(q^2)$. Then, Markov clustering is applied to the correlation matrix $\mathcal{A}$ with the complexity $\mathcal{O}(qz^2)$, where z is the maximum number of non-zero elements in each column in $\mathcal{A}$. Finding non-sparse columns in $\mathcal{M}$, has the complexity $\mathcal{O}(q)$. In addition, selecting the nearest neighbors of non-sparse columns in $\mathcal{M}$ requires the complexity $\mathcal{O}(q^2 \log q)$ in the worst case, where $\mathcal{M}$ has no sparse columns. Totally, the complexity of the clustering for each batch is $\mathcal{O}([q^2+qz^2+q+q^2 \log q])$. The agglomerative hierarchical clustering has the cost $\mathcal{O}(C'^2 \log C')$ where C' is the number of the obtained clusters after the first iteration. Eventually, the total time complexity of clustering t batches is approximated as $\mathcal{O}(t[q^2 + qz^2 + q + q^2 \log q] + C'^2 \log C') \approx \mathcal{O}(q^2)$. With respect to q, the complexity of the HAL method is almost quadratic which is promising for SPN-based image clustering.

5.3.3 Post-processing

As the final phase of the HAL method, post-processing is applied to the resulted clusters. The HAL method generates both fine and coarse clusters. While coarse clusters include a notable number of RNs sharing the same camera fingerprints characteristics, the fine clusters include few RNs which do not share sufficient similarities with the obtained camera fingerprints during the clustering. Due to the nature of the noise-like camera fingerprints [61], and especially the low resolution of the *shared images* on social networks, the presence of the fine clusters are almost unavoidable. For the datasets with a variety of images coming from the same or different smartphone models and brands, merging the fine clusters into the coarse ones may cause a drop in the quality of the clustering. Accordingly, to present a more precise clustering tool for shared image analysis, we remove the fine clusters and preserve the coarse ones. To distinguish the coarse clusters from the fine ones, we introduce a size-based score ζ_i specified for each cluster as follows:

$$\zeta_i = \frac{|c_i|.|C|}{N} \tag{5.8}$$

where N is the number of RNs, and c_i is the i^{th} cluster in the resulted set of clusters, i.e., C, from the HAL method. If $\zeta_i \leq 1$, the cluster c_i is considered as a fine cluster, and it is excluded from C. Otherwise, we keep it as a coarse cluster.

5.3.4 Experimental results

In this section, firstly we explain how we do pre-processing on the RNs and parameter setting for the HAL method. Next, we present the results of the method on the different datasets. Finally, the HAL method is compared with other state-of-the-art SPN-based clustering algorithms. For the datasets covering a variety of smartphone models and brands, it is difficult to achieve the best values for all the mentioned measures in Section 2.5. For example, merging the RNs of different cameras into the same cluster increases FPR, which can propagate the error to the following iterations in the clustering. As a result, $\mathcal{P}$ and $Purity$ decrease, although $\mathcal{R}$ increases [61]. We prefer to have the clusters with high values of $\mathcal{P}$, $Purity$, a low value of FPR, and accurate values of $\mathcal{N}_C$.

5.3.4.1 Experimental setting

To see the functionality of the proposed algorithm on the images from the identical or completely different models of smartphones, we divide the entire dataset $\mathcal{V}$ into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$.

ID	Brand	Model	Original resolution	#images
S_1	Apple	iPhone 4S	3264×2448	178
S_2	Apple	iPhone 4S	3264×2448	200
S_3	Apple	iPhone 5	3264×2448	203
S_4	Apple	iPhone 5	3264×2448	223
S_5	Apple	iPhone 5c	3264×2448	201
S_6	Apple	iPhone 5c	3264×2448	206
S_7	Apple	iPhone 5c	3264×2448	333
S_8	Apple	iPhone 6	3264×2448	129
S_9	Apple	iPhone 6	3264×2448	227
S_{10}	Samsung	Galaxy S III Mini GT-I8190	2560×1920	150
S_{11}	Samsung	Galaxy S III Mini GT-I8190N	2560×1920	200

Table 5.1: Smartphone's characteristics in Dataset $\mathcal{V}_1$.

ID	Brand	Model	Original resolution	#images
S_1	Apple	iPad2	960×720	170
S_2	Apple	iPad mini G	2592×1936	157
S_3	Apple	iPhone 4	2592×1936	217
S_4	Apple	iPhone 6 Plus	3264×2448	256
S_5	Asus	Zenfone	3264×1836	208
S_6	Huawei	Ascend G6-U10	3264×2448	153
S_7	Huawei	Honor 5C	4160×3120	271
S_8	Huawei	P8 GRA-L09	4160×2336	265
S_9	Huawei	P9 EVA-L09	3968×2976	237
S_{10}	Huawei	P9 Lite VNS-L31	4160×3120	234
S_{11}	Lenovo	P70-A	4784×2704	216
S_{12}	LG	D290	3264×2448	224
S_{13}	Microsoft	Lumia 640 LTE	3264×2448	180
S_{14}	OnePlus	A3000	4640×3480	284
S_{15}	OnePlus	A3003	4640×3480	236
S_{16}	Samsung	Galaxy S3 GT-I9300	3264×2448	207
S_{17}	Samsung	Galaxy S4 Mini GT-I9195	3264×1836	208
S_{18}	Samsung	Galaxy S5 SM-G900F	5312×2988	254
S_{19}	Samsung	Galaxy Tab 3 GT-P5210	2048×1536	166
S_{20}	Samsung	Galaxy Tab A SM-T555	2592×1944	154
S_{21}	Samsung	Galaxy Trend Plus GT-S7580	2560×1920	163
S_{22}	Sony	Xperia Z1 Compact D5503	5248×3936	216
S_{23}	Wiko	Ridge 4G	3264×2448	249
S_{24}	Xiaomi	Redmi Note 3	4608×2592	305

Table 5.2: Smartphone's characteristics in Dataset $\mathcal{V}_2$.

After excluding dark and saturated images, $\mathcal{V}_1$ includes 2250 images from 11 smartphones with 5 different models, while $\mathcal{V}_2$ covers 5230 images from 24 smartphones with completely different models. The characteristics of the applied smartphones and their corresponding number of images in our experiments are listed in Tables 5.1 and 5.2. For each dataset $\mathcal{V}_i$, both types of images *native* and *shared* are considered, so we have $\mathcal{V}_i^{\mathrm{NA}}$, $\mathcal{V}_i^{\mathrm{W}}$, $\mathcal{V}_i^{\mathrm{FH}}$ and $\mathcal{V}_i^{\mathrm{FL}}$ corresponding to *Native*, *WhatsApp*, *Facebook high-resolution* and *Facebook low-resolution* images, respectively. Finally, to test the generalization of the proposed algorithm, we run it on the whole dataset, i.e., $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$. In particular, we have $\mathcal{V}^{\mathrm{NA}} = \mathcal{V}_1^{\mathrm{NA}} \cup \mathcal{V}_2^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}} = \mathcal{V}_1^{\mathrm{W}} \cup \mathcal{V}_2^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}} = \mathcal{V}_1^{\mathrm{FH}} \cup \mathcal{V}_2^{\mathrm{FH}}$, and $\mathcal{V}^{\mathrm{FL}} = \mathcal{V}_1^{\mathrm{FL}} \cup \mathcal{V}_2^{\mathrm{FL}}$, each of them includes 7480 images. In Table 5.3, the lowest and the highest resolutions of the images for each dataset are listed.

Dataset	Lowest resolution	Highest resolution
$\mathcal{V}_1^{\mathrm{NA}}$	2560×1920	3264×2448
$\mathcal{V}_1^{\mathrm{W}}$	1280×960	1280×960
$\mathcal{V}_1^{\mathrm{FH}}$	2048×1536	2048×1536
$\mathcal{V}_1^{\mathrm{FL}}$	960×720	1224×918
$\mathcal{V}_2^{\mathrm{NA}}$	960×720	4608×2592
$\mathcal{V}_2^{\mathrm{W}}$	960×720	2048×1536
$\mathcal{V}_2^{\mathrm{FH}}$	960×720	1280×960
$\mathcal{V}_2^{\mathrm{FL}}$	1040×584	1312×984
$\mathcal{V}^{\mathrm{NA}}$	960×720	5248×3936
$\mathcal{V}^{\mathrm{W}}$	960×720	1280×960
$\mathcal{V}^{\mathrm{FH}}$	960×720	2048×1536
$\mathcal{V}^{\mathrm{FL}}$	1040×584	1312×984

Table 5.3: Lowest and highest image resolution in different datasets.

To empirically set a specific resolution for resizing the RNs and the required parameters for the the HAL method, we consider the sample datasets $\mathcal{V}_0^{\mathrm{NA}} \subseteq \mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}_0^{\mathrm{W}} \subseteq \mathcal{V}^{\mathrm{W}}$, $\mathcal{V}_0^{\mathrm{FH}} \subseteq \mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}_0^{\mathrm{FL}} \subseteq \mathcal{V}^{\mathrm{FL}}$. From each of 35 smartphones, 100 images are randomly selected, so each dataset includes 3500 images. The sample datasets make the setting facilitative and they still include images from a variety of smartphone models and brands. Hence, the obtained values from the setting for each dataset can be generalized to the entire datasets, i.e., $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}^{\mathrm{FL}}$.

The impact of resizing versus cropping is evaluated based on both correlation gain and clustering quality. The related results are presented in Figure 5.2 in terms of Receiver Operating Characteristics (ROC) showing the relationship between *TPR* and *FPR* measures. Each ROC curve is obtained by selecting a threshold from the range [-1,1] and comparing it with the values in the full-pairwise correlation matrix, representing the similarities between the resized/cropped RNs. The results of the HAL method by using the correlation matrices obtained from cropping and resizing RNs in $\mathcal{V}_0^{\mathrm{NA}}$, with different resolutions, are also presented in Tables 5.4 and 5.5. The best value of each measure is

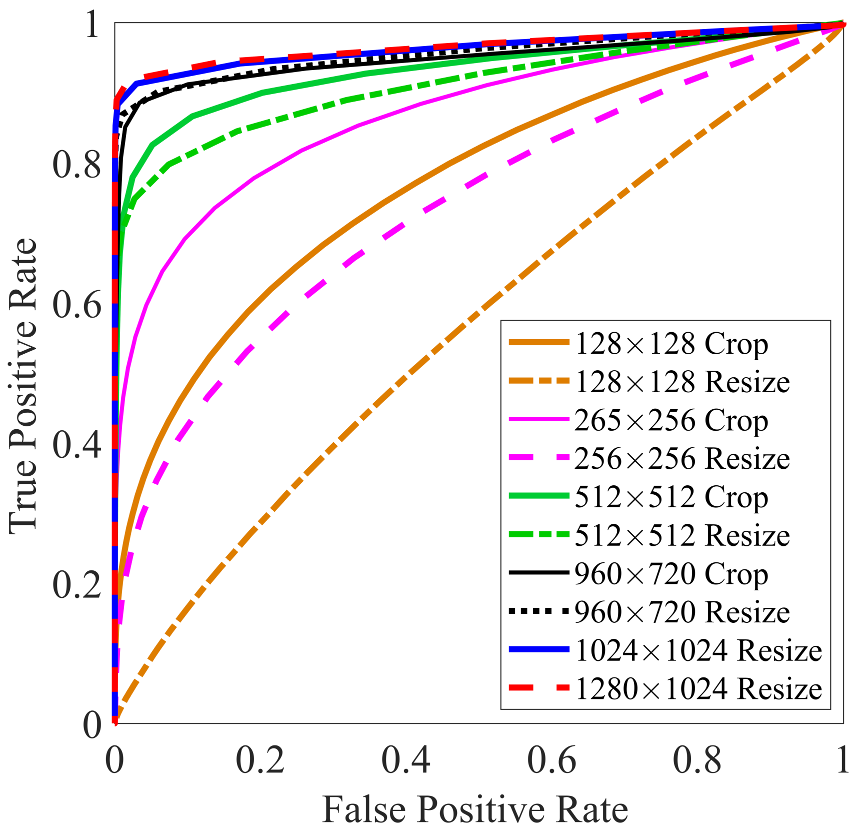


Figure 5.2: ROC curves obtained from resizing versus cropping with different resolution of RNs in $\mathcal{V}_0^{\mathrm{NA}}$.

highlighted in bold. The results in Figure 5.2 and Tables 5.4 and 5.5, show that for the resolutions smaller than 960 × 720, cropping generates better results. In contrast, for the resolutions equal to or greater than 960 × 720, resizing delivers far better clustering results compared with cropping. As it can be easily seen from Table 5.5, resizing the RNs to the resolution 1024 × 1024 concludes the best values of $\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}1$, $\mathcal{F}2$, ARI, FPR and $\mathcal{N}_C$. Accordingly, in the experiments, we resize all the RNs to the resolution 1024 × 1024 px.

Size	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
960 × 720[1]	0.69	0.61	0.65	0.456	0.64	0.96	0.00	31/35	95/3500
512 × 512	0.67	0.49	0.57	0.365	0.56	0.96	0.00	37/35	129/3500
256 × 256	0.65	0.30	0.41	0.226	0.40	0.88	0.00	60/35	158/3500
128 × 128	0.48	0.13	0.21	0.075	0.20	0.59	0.00	159/35	257/3500

Table 5.4: Results (%) of HAL method on $\mathcal{V}_0^{\mathrm{NA}}$, by cropping the RNs with different image resolutions.

Size	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
1280 × 1024	**0.99**	0.75	0.86	0.781	0.85	**0.99**	**0.00**	**35/35**	121/3500
1024 × 1024	**0.99**	**0.76**	**0.86**	**0.790**	**0.86**	0.99	**0.00**	**35/35**	138/3500
960 × 720	0.98	0.72	0.83	0.747	0.83	0.99	**0.00**	34/35	134/3500
512 × 512	0.95	0.44	0.60	0.470	0.59	0.96	**0.00**	48/35	115/3500
256 × 256	0.50	0.02	0.05	0.012	0.04	0.59	**0.00**	280/35	305/3500
128 × 128	0.031	0.14	0.05	0.005	0.00	0.17	0.13	26/35	**98/3500**

Table 5.5: Results (%) of HAL method on $\mathcal{V}_0^{\mathrm{NA}}$, by resizing the RNs with different image resolutions.

There are few parameters that should be set for the hybrid algorithm. Using the sample datasets $\mathcal{V}_0^{\mathrm{NA}}$, $\mathcal{V}_0^{\mathrm{W}}$, $\mathcal{V}_0^{\mathrm{FH}}$, and $\mathcal{V}_0^{\mathrm{FL}}$, we set the parameters empirically to the values resulting in the highest quality of the clustering. As an example of the setting process, the results of setting the inflation parameter η in (5.3) and ψ for the adaptive threshold in (5.7), for $\mathcal{V}_0^{\mathrm{NA}}$, are presented in Figure 5.3. Based

[1]The highest resolution for cropping RNs is 960 × 720 px, based on Table 4.3.

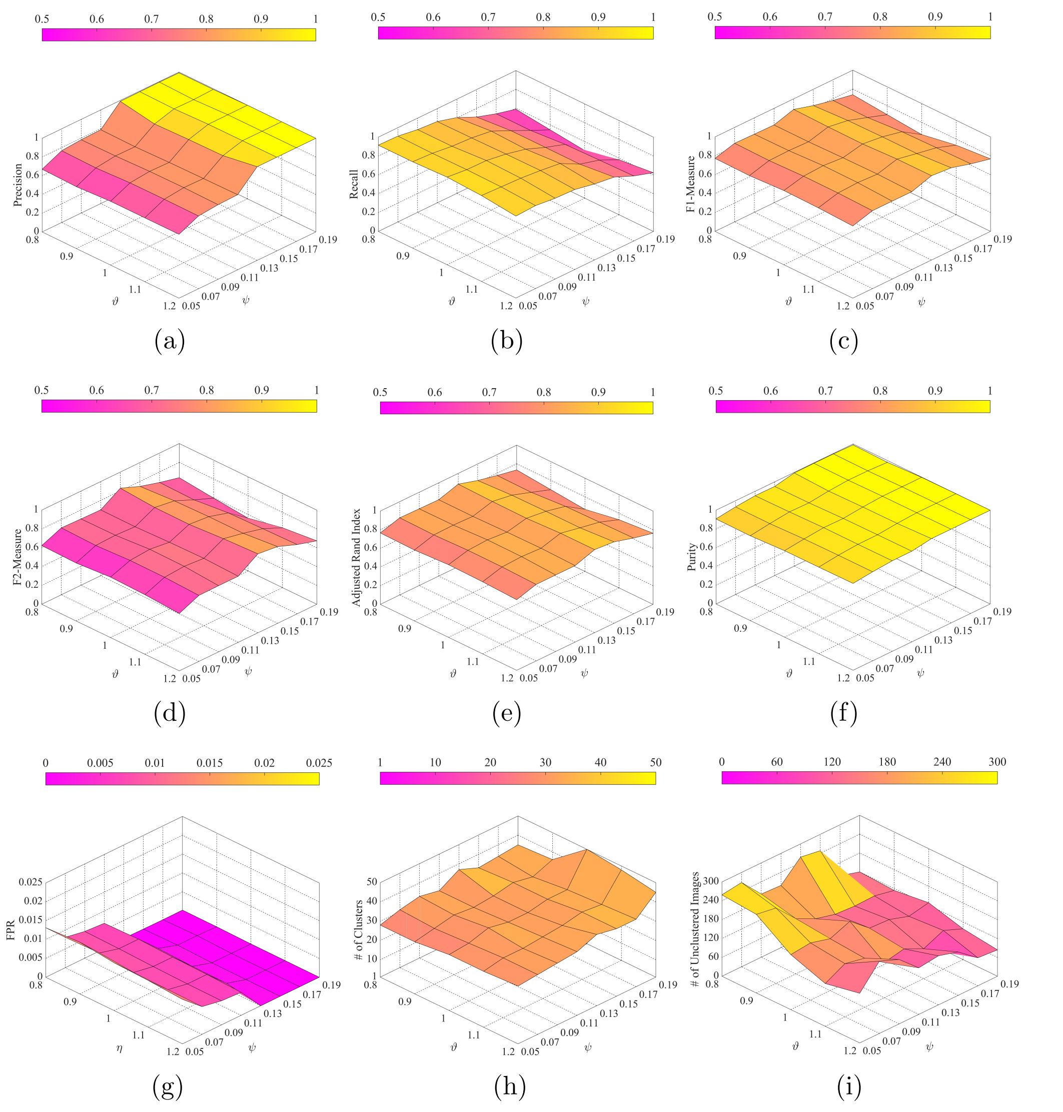


Figure 5.3: The inflation parameter η and threshold ψ affect the clustering based on HAL method on $\mathcal{V}_0^{\mathrm{NA}}$, (a) *Precision*, (b) *Recall*, (c) *F1-Measure*, (e) *F2-Measure*, (f) *Adjusted Rand Index*, (f) *Purity*, (h) *False Positive Rate*, (h) number of the obtained clusters and (i) number of unclustered images.

on the graphs shown in Figure 5.3, setting ψ to a value in the range $[0.11, 0.17]$ and $\eta = 1$ generates reasonable clustering results. With $\eta = 1$, if ψ is set to a high value, e.g., 0.19, the clustering generates a large number of fine clusters up to 50. Accordingly, to have the best values of $\mathcal{P}$, *purity*, *FPR* and $\mathcal{N}_C$ we set $\psi = 0.15$ for *native images*. Similar parameter setting is performed for the *shared*

Notation	Value	Description
q	1500	batch size for partitioning dataset
e	2	expansion parameter in (5.2)
η	1	inflation parameter in (5.3)
ς	0.005	prune parameter in (5.4)
$\mathcal{G}$	2	initial value of global chaos in (5.6) and Algorithm 1
ξ	0.3	threshold for global chaos in (5.6)
ψ	0.15	scaling factor in (5.7) for $\mathcal{V}_1^{\text{NA}}$,$\mathcal{V}_2^{\text{NA}}$ and $\mathcal{V}^{\text{NA}}$
	0.09	scaling factor in (5.7) for $\mathcal{V}_1^{\text{W}}$,$\mathcal{V}_2^{\text{W}}$ and $\mathcal{V}^{\text{W}}$
	0.07	scaling factor in (5.7) for $\mathcal{V}_1^{\text{FH}}$,$\mathcal{V}_2^{\text{FH}}$ and $\mathcal{V}^{\text{FH}}$
	0.03	scaling factor in (5.7) for $\mathcal{V}_1^{\text{FL}}$,$\mathcal{V}_2^{\text{FL}}$ and $\mathcal{V}^{\text{FL}}$
τ	0.004	minimum threshold in (5.7) for adaptive threshold

Table 5.6: Parameter values for the HAL method.

images. In Table 5.6, all the required parameters and their set values are listed. It can be seen that while ψ depends on the quality of the images and it needs to be set to different values for *shared* and *native images*, the values of other parameters are set equally for all the images in the datasets. So, the appropriate values of ψ for the images of *Native*, *WhatsApp*, *Facebook high-resolution* and *Facebook low-resolution* datasets are set to 0.09, 0.07, and 0.03, respectively.

5.3.4.2 Hybrid Markov Clustering

In this section, we illustrate the obtained results of the hybrid clustering on different datasets $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}$, covering 11, 24, and 35 smartphones, respectively, for both *native* and *shared images.* We try to figure out how the proposed algorithm is capable of clustering images in different datasets, even if the datasets include images from identical smartphones. Since the algorithm randomly partitions the RNs into some batches, the results from multiple running of the algorithm on a given dataset may vary. Thereby, for each dataset, we run the algorithm 10 times and report the average results for the different measures. The results for the dataset $\mathcal{V}_1$ which includes smartphones from the same models are shown in Table 5.7. Due to the high resolution of *native images* in $\mathcal{V}_1^{\text{NA}}$, as shown in Table

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
$\mathcal{V}_1^{\text{NA}}$	1.000	0.826	0.905	0.855	0.896	1.000	0.000	11/11	134/2250
$\mathcal{V}_1^{\text{W}}$	0.975	0.775	0.863	0.791	0.850	0.993	0.002	13/11	252/2250
$\mathcal{V}_1^{\text{FH}}$	0.994	0.720	0.835	0.758	0.821	0.994	0.001	12/11	268/2250
$\mathcal{V}_1^{\text{FL}}$	0.866	0.601	0.705	0.565	0.680	0.914	0.009	9/11	548/2250

Table 5.7: Results (%) of clustering based on HAL method on $\mathcal{V}_1$.

4.3, the clustering results in the best values of $\mathcal{P}$, $\mathcal{R}$, $\mathcal{F}1$, $\mathcal{F}2$, ARI, $Purity$, FPR, $\mathcal{N}_C$ and $\mathcal{N}_U$. Interestingly, the algorithm can detect all the 11 clusters corresponding to the smartphones in $\mathcal{V}_1^{\text{NA}}$. For the datasets $\mathcal{V}_1^{\text{W}}$ and $\mathcal{V}_1^{\text{FH}}$, the results are better than those for $\mathcal{V}_1^{\text{FL}}$. It is because of low-resolution images in $\mathcal{V}_1^{\text{FL}}$, see Table 5.3, but the algorithm is still capable to cluster the images with $\mathcal{P} = 0.866$, $Purity = 0.914$, $FPR = 0.009$, and $\mathcal{N}_C = 9/11$. In Table 5.8, similar

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
$\mathcal{V}_2^{\text{NA}}$	0.992	0.672	0.801	0.713	0.794	0.992	0.000	27/24	813/5230
$\mathcal{V}_2^{\text{W}}$	0.970	0.610	0.750	0.641	0.741	0.972	0.000	24/24	985/5230
$\mathcal{V}_2^{\text{FH}}$	0.958	0.627	0.758	0.649	0.749	0.966	0.001	25/24	1053/5230
$\mathcal{V}_2^{\text{FL}}$	0.798	0.543	0.647	0.476	0.634	0.876	0.006	29/24	1258/5230

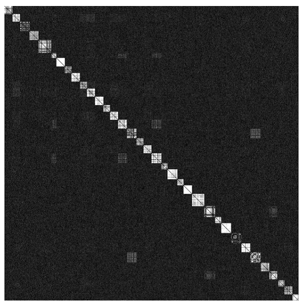
Table 5.8: Results (%) of clustering based on HAL method on $\mathcal{V}_2$.

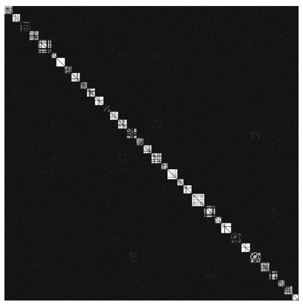
trends can be observed for $\mathcal{V}_2$, with the best and worst results for $\mathcal{V}_2^{\text{NA}}$ and $\mathcal{V}_2^{\text{FL}}$, respectively. However, clustering the images of $\mathcal{V}_2$ generates lower quality than that of $\mathcal{V}_1$. Obviously, the reason is that $\mathcal{V}_2$ covers images from completely different smartphone models and brands as well as various image resolutions. In Table 5.9, the results for *native* and *shared images* taken by all the 35 smartphones are shown. We consider it as the most challenging test for the proposed algorithm. Comparing the results in Tables 5.8 and 5.9, we can see that the algorithm can effectively cluster the images even in the larger datasets, i.e., $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}^{\mathrm{FL}}$.

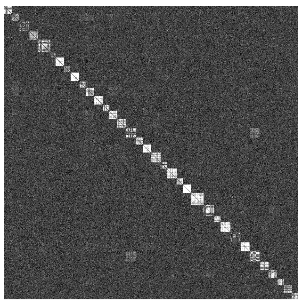
Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
$\mathcal{V}^{\mathrm{NA}}$	0.992	0.720	0.834	0.756	0.830	0.994	0.000	37/35	725/7480
$\mathcal{V}^{\mathrm{W}}$	0.964	0.600	0.733	0.628	0.727	0.975	0.000	33/35	1601/7480
$\mathcal{V}^{\mathrm{FH}}$	0.962	0.610	0.746	0.636	0.740	0.975	0.000	33/35	1624/7480
$\mathcal{V}^{\mathrm{FL}}$	0.750	0.513	0.609	0.426	0.599	0.847	0.005	33/35	1950/7480

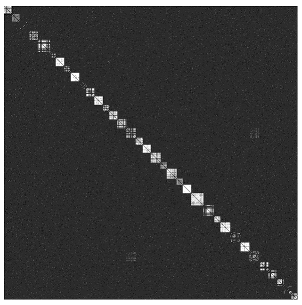
Table 5.9: Results (%) of clustering based on HAL method on $\mathcal{V}$.

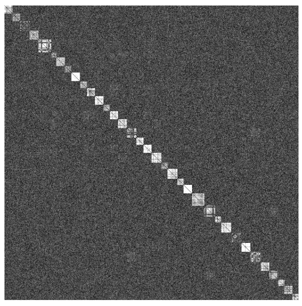
To see why the proposed clustering is effective in calculating only a portion of the full-pairwise correlation matrix, in Figure 5.4, sub-figures (a)-(h), we present both full-pairwise correlation matrices and the correlation matrices calculated by the algorithm for each dataset. Indeed, the HAL method exploits the sparseness of the full-pairwise correlation matrix and applies the Markov clustering, through which the trivial correlation values between the clusters in each batch are pruned. Subsequently, only the candidate clusters, which are selected based on the obtained transition matrix from the Markov clustering and the nearest neighboring, are checked in the merging. Therefore, to produce the adaptive threshold in (5.7), intra-cluster correlations and inter-cluster correlations of the candidate clusters need to be calculated, besides a few correlations between the clusters randomly partitioned in batches. Accordingly, for the datasets $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$, and $\mathcal{V}^{\mathrm{FL}}$, the algorithm calculates 14.75%, 14.52%, 14.66%, and 14.59% of the corresponding full-pairwise correlation matrices, respectively. In sub-figures (a)-(h) of Figure 5.5, the probability distributions of the corresponding correlation matrices in Figure 5.4 are shown. The more the inter-camera and intra-camera correlation distributions are separable, the better the quality of the clustering is obtained. Comparing the graphs presented in Figure 5.5 in the sub-figures (b), (d), (f) and (h) with those presented in the sub-figures (a), (c), (e) and (g), it can be seen

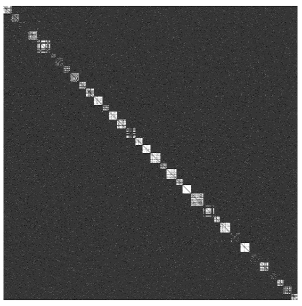

(a)

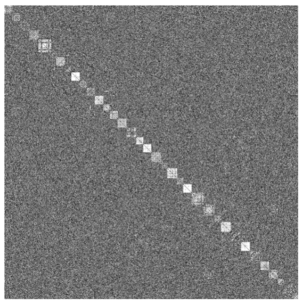

(b)


(c)


(d)


(e)


(f)


(g)

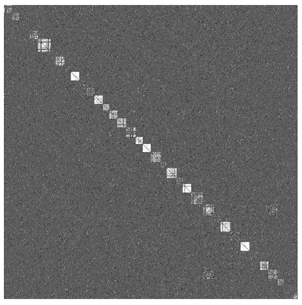

(h)

Figure 5.4: Effectiveness of the HAL method in calculating correlation matrix: (a), (c), (e) and (g) are full-pairwise correlation matrices of RNs of the images in $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}^{\mathrm{FL}}$, respectively, and (b), (d), (f) and (h) are the corresponding correlation matrices calculated by the method.

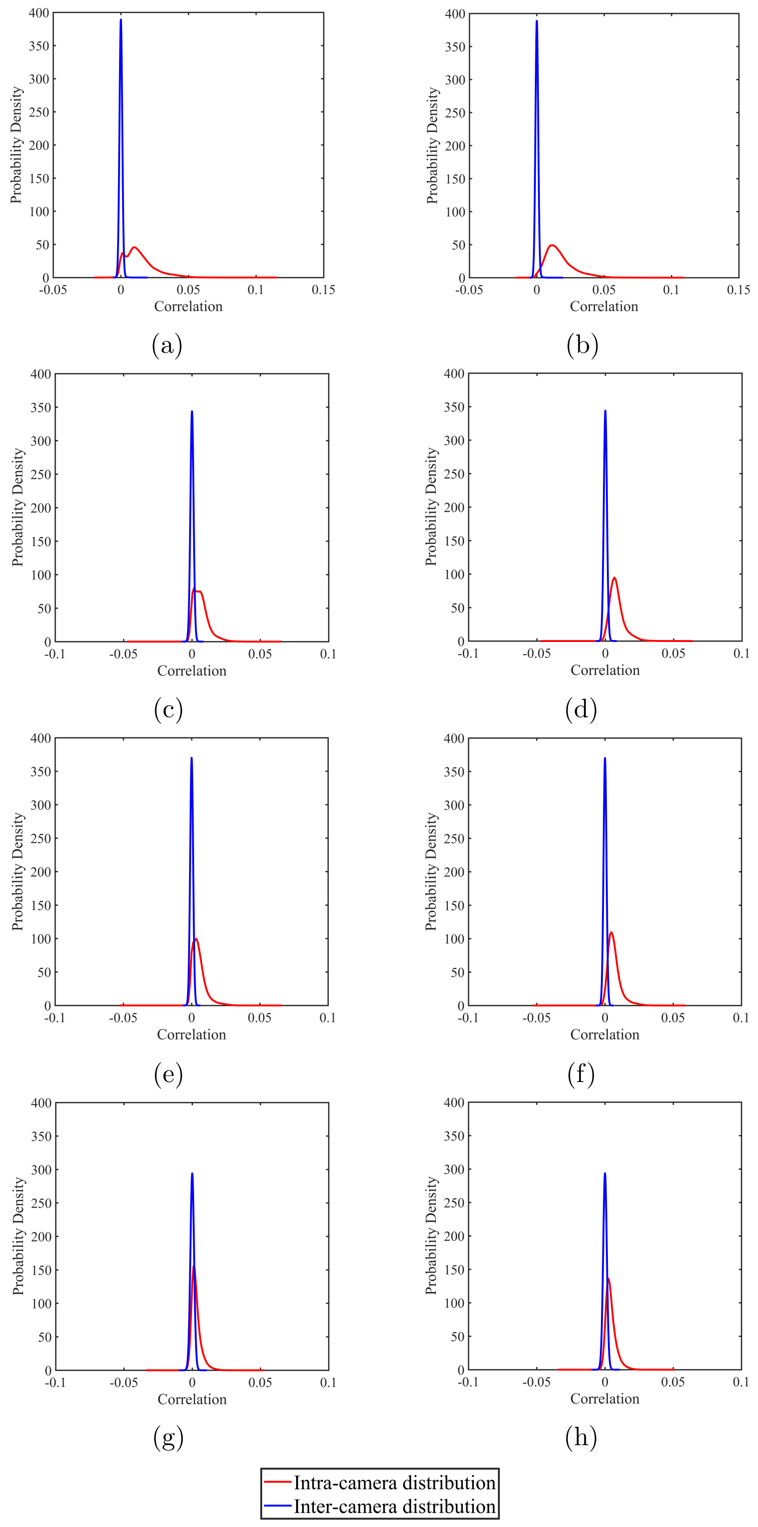


Figure 5.5: Effectiveness of the HAL method in calculating correlation matrix: (a), (b), (c), (d), (e), (f), (g) and (h) are probability distributions corresponding to the correlation matrices in Figure 5.4.

that the calculated correlation matrices by the algorithm provide more separation between the probability distributions. In other words, the decrease in the overlapping areas of the probability distributions prevents wrong merging in the clustering, so more precise clusters are achieved.

5.3.4.3 Comparison with other clustering algorithms

The proposed HAL is compared with other SPN-based image clustering algorithms. All the clustering algorithms including the proposed HAL, Fast Clustering (FC) [59], and Hierarchical Clustering (HC) [53] and Correlation Clustering (CC) [58] are performed on the datasets $\mathcal{V}^{\text{NA}}$, $\mathcal{V}^{\text{W}}$, $\mathcal{V}^{\text{FH}}$, and $\mathcal{V}^{\text{FL}}$, and the results are analysed based on both clustering quality and running time. The comparison results in Tables 5.10, 5.11, 5.12 and 5.13 show that the proposed algorithm achieves the outstanding values of $\mathcal{P}$, *Purity*, *FPR* and $\mathcal{N}_C$, apart from only *Purity* for $\mathcal{V}^{\text{FL}}$. The values of $\mathcal{F}$ and *ARI* obtained by CC, and $\mathcal{R}$ by HC are the highest for the datasets $\mathcal{V}^{\text{NA}}$, $\mathcal{V}^{\text{W}}$ and $\mathcal{V}^{\text{FH}}$. Although FC does not show any improvement for the mentioned datasets, it generates the highest values of $\mathcal{F}1$, $\mathcal{F}2$, *ARI* and *Purity* for $\mathcal{V}^{\text{FL}}$. Also, for all the datasets, HC concludes the highest $\mathcal{R}$. Compared with the other methods, the HAL method has achieved the best values of $\mathcal{P}$, *Purity*, *FPR* and $\mathcal{N}_C$.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*	$\mathcal{N}_C$	$\mathcal{N}_U$
HAL	**0.992**	0.720	0.834	0.756	0.830	**0.994**	**0.000**	**37/35**	725/7480
FC [59]	0.952	0.759	0.845	0.759	0.841	0.982	0.001	63/35	**0/7480**
HC [53]	0.205	**0.949**	0.338	0.196	0.304	0.828	0.112	23/35	**0/7480**
CC [58]	0.987	0.863	**0.921**	**0.875**	**0.919**	0.915	**0.000**	46/35	**0/7480**

Table 5.10: Comparison of clustering methods on $\mathcal{V}^{\text{NA}}$.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
HAL	**0.964**	0.600	0.733	0.628	0.727	**0.975**	**0.000**	**33/35**	1601/7480
FC [59]	0.919	0.722	0.809	0.702	0.804	0.974	0.001	65/35	**0/7480**
HC [53]	0.245	**0.863**	0.382	0.217	0.352	0.813	0.081	32/35	**0/7480**
CC [58]	0.952	0.787	**0.862**	**0.782**	**0.858**	0.856	0.001	56/35	**0/7480**

Table 5.11: Comparison of clustering methods on $\mathcal{V}^{\mathrm{W}}$.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
HAL	**0.962**	0.610	0.746	0.636	0.740	**0.975**	**0.000**	**33/35**	1624/7480
FC [59]	0.913	0.758	0.828	0.727	0.824	0.974	0.002	64/35	**0/7480**
HC [53]	0.475	**0.823**	0.602	0.405	0.587	0.793	0.027	30/35	**0/7480**
CC [58]	0.955	0.793	**0.866**	**0.790**	**0.863**	0.841	0.001	58/35	**0/7480**

Table 5.12: Comparison of clustering methods on $\mathcal{V}^{\mathrm{FH}}$.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_U$
HAL	**0.750**	0.513	0.609	0.426	0.599	0.847	**0.005**	**33/35**	1950/7480
FC [59]	0.665	0.690	**0.717**	**0.489**	**0.680**	**0.915**	0.011	52/35	**0/7480**
HC [53]	0.031	**0.999**	0.061	0.309	0.003	0.520	0.941	2/35	**0/7480**
CC [58]	0.712	0.632	0.669	0.485	0.660	0.776	0.007	42/35	**0/7480**

Table 5.13: Comparison of clustering methods on $\mathcal{V}^{\mathrm{FL}}$.

In addition to the quality of clustering, the running time of a clustering algorithm is another important factor that should be taken into account for real-life applications. Hence, we compare the running time of the algorithms on the datasets $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$, and $\mathcal{V}^{\mathrm{FL}}$ as shown in Figure 5.6. The time for each algorithm is separately depicted by I/O, correlation calculation, and clustering operations. In terms of the overall running time, FC and CC are the fastest and the slowest algorithms, respectively. FC, HC, and CC have to be provided with the full-pairwise correlation matrix, so they have equal I/O and correlation calculation time. Despite the shorter clustering time of FC, HC and CC, they cannot be applicable to large-scale datasets due to the unavoidable time needed for the computation of full-pairwise correlation matrix. In contrast,

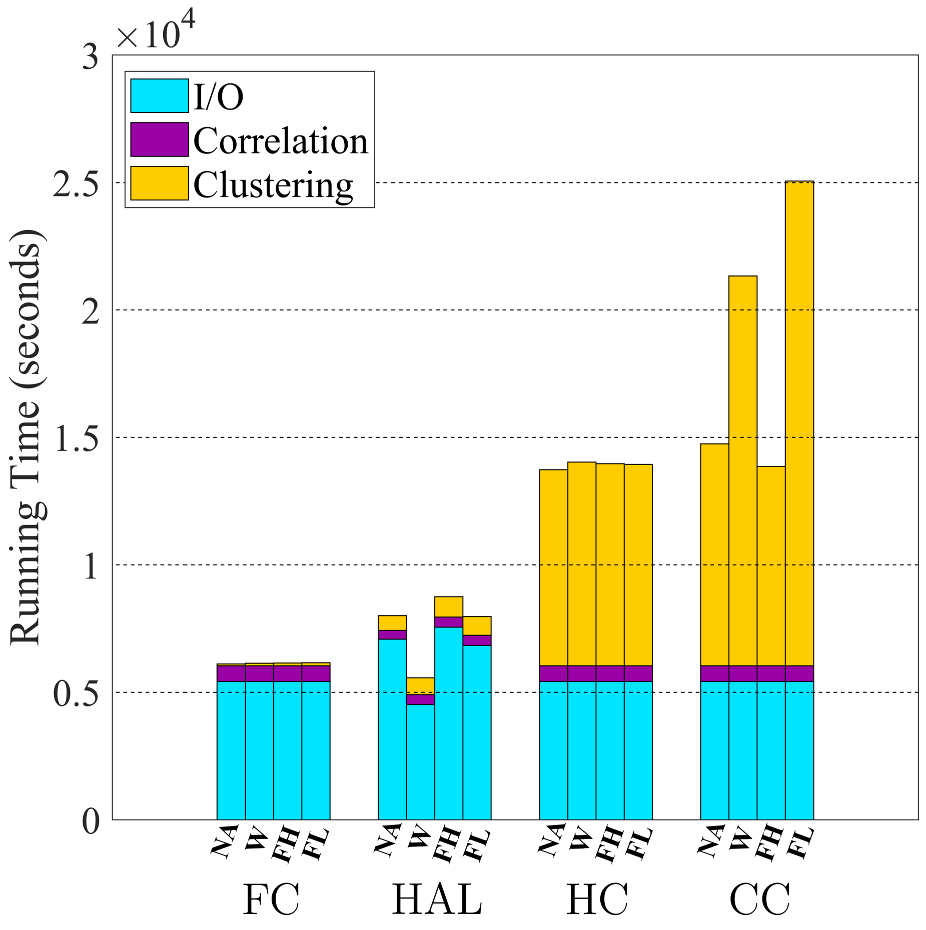


Figure 5.6: Running time of different clustering algorithms on $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$, and $\mathcal{V}^{\mathrm{FL}}$ with image resolution 1024×1024.

by the proposed HAL, a small portion of the correlation matrix is calculated. This accelerates the clustering and also results in more precise clusters.

5.4 Hybrid Markov Clustering with Outliers Method

In the HAL algorithm, the RNs which do not share sufficient similarities with the obtained clusters are removed in the post-processing phase, i.e., Subsection 5.3.3. However, when the dataset is perturbed by a large number of outliers, the HAL is not cost-effective in terms of the running time and the memory usage, as the ouliers have to be kept until the final step of the clustering. Hence, an effective and efficient clustering algorithm is needed for outlier detection and removal such that the clustering of the other images is not affected negatively. We present the HALO method to detect and remove the outliers, performed in the first steps of the clustering, and cluster the remaining images based on the camera sources. The flowchart of the HALO method is presented in Figure 5.7. The difference between Figure 5.1 and Figure 5.7 is the outlier detection step which purifies each batch with detecting and removing the outliers from the clustering process. Particularly, the outliers are detected by a Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. We show that the HALO method is stable against the the number of the outliers and can cluster the images coming from unknown number of sources (smartphones in our case).

5.4.1 Outlier removal based on DBSCAN

Outlier detection is a pre-clustering step that is usually performed in many density based clustering algorithms. Through a density-based approach, the clusters of arbitrary shapes can be found, i.e., It can even find a cluster completely surrounded by (but not connected to) a different cluster. DBSCAN is a density-based clustering non-parametric algorithm proposed in [80]. DBSCAN does not require to be provided with the number of clusters in the data a priori, unlike k-means, k-medoids and hierarchical clustering.

Giving some objects, DBSCAN groups together the objects and marks these which lie alone in low-density regions as the ouliers. It finds the object's neighbors by density ϑ on an n-dimensional sphere with radius ϵ. The parameter ϑ is defined as the minimum number (a threshold) of the objects huddled together

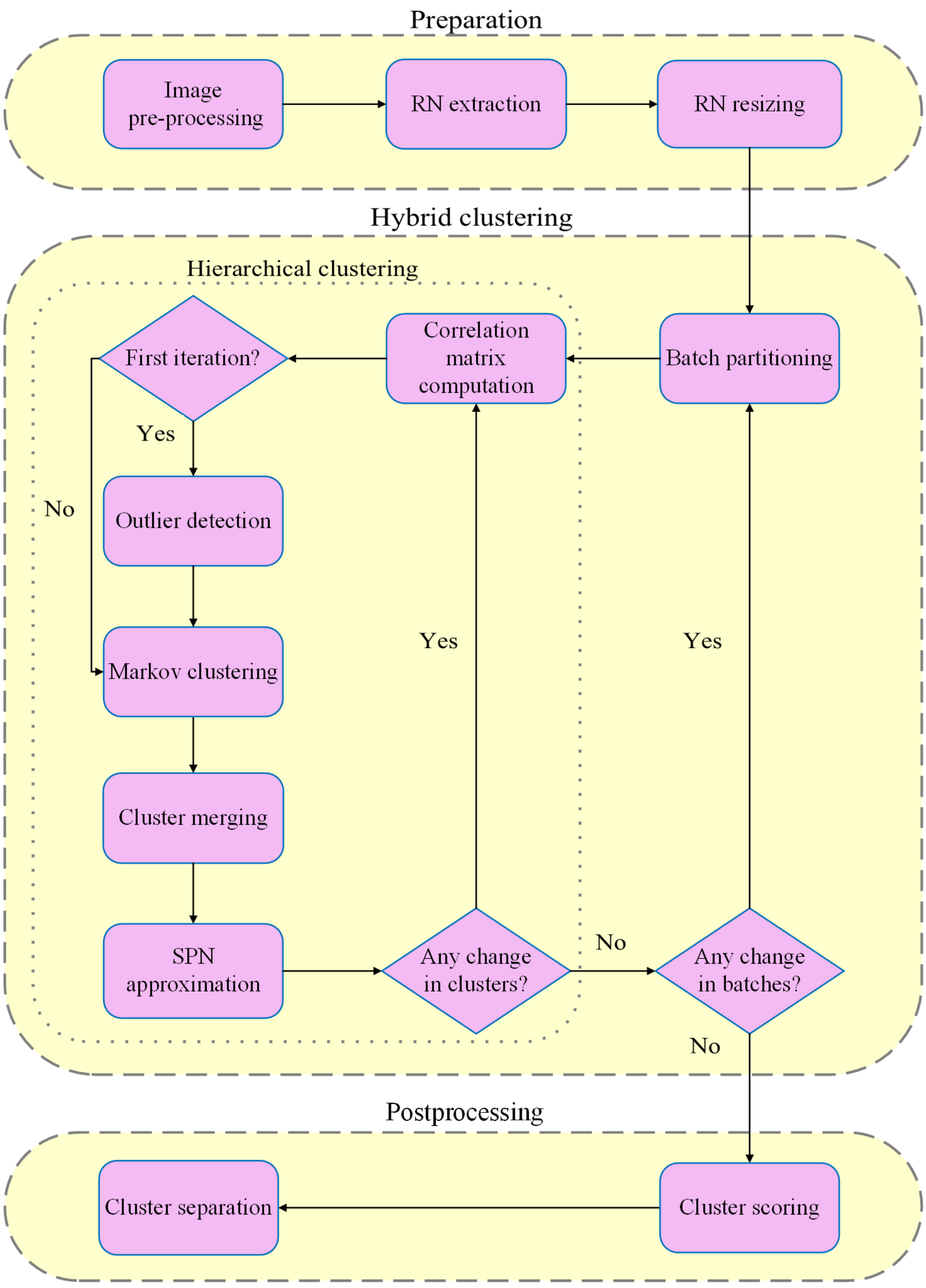


Figure 5.7: Flowchart of HALO method.

for a region to be considered dense, and ϵ is a parameter specifying the radius of the neighborhood. A cluster can be defined as the maximal set of density connected objects in the space.

DBSCAN defines different classes of objects such as core object, reachable object and outlier, as follows:

- A is a core object if at least ϑ objects are within distance ϵ of it (including A).
- An object B is directly reachable from A if the object B is within distance ϵ from core object A. Objects are only said to be directly reachable from core objects.
- An object C is reachable from B if there is a path $p_1, ..., p_n$ with $p_1 = A$ and $p_n = C$ where each p_{i+1} is directly reachable from p_i. It means that all objects on the path must be core objects, with the possible exception of C.
- O is an outlier object that lies in no cluster and it is not reachable from any other object. So, this object will have its own cluster.

If A is a core object, then it forms a cluster together with all objects, whether core or non-core, that are reachable from it. Each cluster includes at least one core object. Non-core objects can be part of a cluster, forming the edge of cluster, because they cannot be used to reach more objects.

Reachability in the DBSCAN is not a symmetric relation, based on the above definitions, no object may be reachable from a non-core object, regardless of distance. Thus, a non-core point may be reachable, but nothing can be reached from it. Accordingly, a notion of connectedness is needed to define the extent of

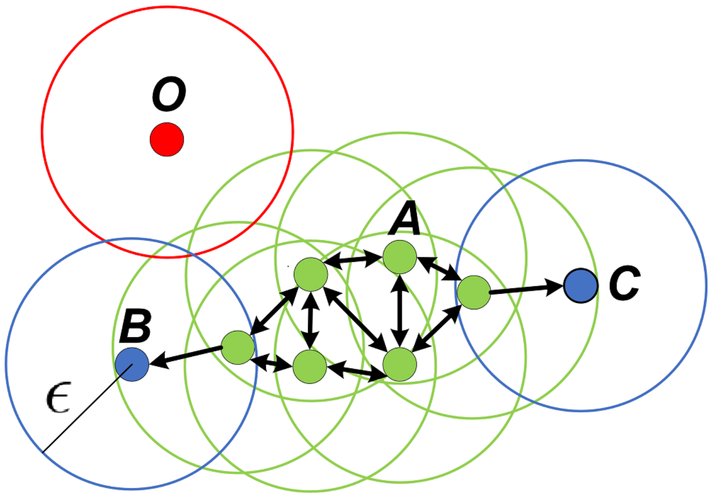


Figure 5.8: An example of DBSCAN algorithm with $\vartheta = 4$: object A and the other green points are core objects, objects B and C are reachable from A via other core objects and Object O is an outlier object.

the clusters found by DBSCAN. Two points p and q are density-connected if there is a point r such that both p and q are reachable from r. Density-connectedness is symmetric. Subsequently, a cluster satisfies two properties:

- All objects within the cluster are mutually density-connected.
- If an object is density-reachable from any object of the cluster, it is part of the cluster as well.

To give a visual example, consider a set of objects in some space to be clustered, see Figure 5.8. Object A and the other green objects depict core objects, as the area surrounding these objects in an ϵ radius contain at least 4 objects, including the object itself. Since the green objects are all reachable from one another, they are grouped in the same cluster. Objects B and C are not core objects, but they are reachable from A via other core objects. Accordingly, they belong to the cluster as well. Object O is an outlier object that is neither a core object nor reachable.

DBSCAN is a strong and effective method when the distribution of values in the feature space cannot be assumed. Hence, it can handle the process of clustering the high dimensional RNs specified by a large feature vector e.g., with the size 1024×1024, and detect outliers. DBSCAN is robust to outliers. We apply DBSCAN to the clustering just before the Markov clustering is performed and for the first iteration of the processing of each batch, see Figure 5.7. Once the batch is purified by removing the discovered outliers, it is passed to the following stages to cluster the remained RNs into unknown number of clusters, each of them including RNs coming from the same samrtphone. It is worth mentioning that the larger the size is considered for each batch the better the effectiveness of the outlier detection is concluded, as the RNs are compared in a larger scale.

The algorithms of the applied DBSCAN and the proposed HALO methods are presented in Algorithms 3 and 4, respectively.

Algorithm 3: DBSCAN algorithm.

```
input: distance matrix $\hat{\mathcal{A}} = 1 - \mathcal{A}$ including distances between RNs
output: two list of clusters, outliers $C_O$ and normal RNs $C_N$
- threshold for neighborhood of each RN, $\epsilon$
- threshold for number of RNs to form a dense region, $\vartheta$
- corresponding to all RNs, consider a list $lb$
while all RNs in lb are labeled do
    - pick an arbitrary data point $RN$ as the first point
    - mark $RN$ in the list $lb$ as the visited sample
    - find all RNs present in its neighborhood (upto $\epsilon$ distance from the
      point), and call it a set $nb$
    if $|nb| >= \vartheta$ then
        - consider the $RN$ as an inlier and add it to $C_N$
        for $id = 1 : |nb|$ do
            - consider all points within $\epsilon$ distance (members of $nb$) as inliers
    else
        - add $RN$ to $C_O$ as an outlier
return $C_O$ and $C$
```

Algorithm 4: Hybrid Markov clustering with outliers.

```
input: pre-processed RNs
output: list of clusters, outliers $C_O$ and $C$ normal images
- number of RNs, $N$
- scaling factor, $\psi$ in (5.7)
- minimum threshold, $\tau$ in (5.7)
- size of batches, $q$
- clustering initialization, $C_{old} = \{\}$
- considering a set of single clusters corresponding to the RNs, $C_{new} = \{c_1, c_2, ..., c_N\}$
- initializing a set of camera fingerprints with the RNs corresponding to the clusters,
  $F = \{f_1, f_2, ..., f_N\}$
- partitioning initialization, $B_{old} = \{\}$
- $t = \lceil \frac{N}{q} \rceil$
- randomly partition $C_{new}$ into $t$ batches with size $q$, $B_{new} = \{b_1, b_2, ..., b_t\}$
- parameter for determining first iteration to apply DBSCAN, $flag = 1$
while $|B_{new}| \neq |B_{old}|$ do
    for $k = 1 : t$ do
        while $|C_{new}| \neq |C_{old}|$ do
            - compute correlation matrix $\mathcal{A}$ by (2.3)
            if $flag$ then
                - apply DBSCAN to $\mathcal{A}$ by Algorithm 3
                - put the found outliers in the cluster $C_O$
                - extract the correlation matrix of normal RNs, i.e., $\mathcal{A}_N$
                - $\mathcal{A} = \mathcal{A}_N$
            - apply Markov clustering to $\mathcal{A}$ and generate the probability matrix $\mathcal{M}$ by
              Algorithm 1
            - put non-sparse column's indices in the list $L$
            for $i = 1 : |L|$ do
                - find the nearest cluster $c_j$ to the cluster $c_i$ from the list $L$
                - compute the adaptive threshold $\mathcal{T}$ by (5.7)
                if $\mathcal{A}(f_i, f_j) > \mathcal{T}$ then
                    - merge clusters $c_i$ and $c_j$
                else
                    - continue
            - put the obtained clusters in $C_{new}$
            - update the camera fingerprints in $F$ for the merged clusters by (2.2)
            - $C_{old} = C_{new}$
    - consider all the obtained clusters from batches as a new cluster $C_{new}$
    - $B_{old} = B_{new}$
    - $N = |C_{new}|$
    - update $t$, $t = \lceil \frac{N}{q} \rceil$
    - partition the clusters in $C_{new}$ into $t$ batches with size $q$, and form $B_{new}$
    - $flag = 0$
- $C = C_{new}$
return $C_O$ and $C$
```

5.4.2 Experimental results

To validate the HALO method, we apply the VISION image dataset. Firstly, some parameter setting is performed for the DBSCAN algorithm. Then, the results of the HALO method are presented and discussed. We evaluate the HALO method by all the mentioned measures in (2.4)-(2.14).

5.4.2.1 Experimental setting

We use all the set parameter values mentioned in Table 5.6 for the HALO method and need to only set the parameters ϵ and ϑ for the applied DBSCAN algorithm. The parameters should be set to the values resulting in the best quality of the outlier detection and the clustering. So, we consider different ranges for the parameters ϵ and ϑ, which are respectively $[0.991, 0.993, 0.995, 0.997, 0.999]$ and $[10, 15, 20, 25, 30]$. We perform the parameter setting on the sample dataset $\mathcal{V}_0^{\mathrm{NA}}$ and perturb the dataset with 500 outliers. We evaluate the clustering, which is performed by selecting different values of the parameters, based on different measures defined by (2.4) to (2.14). From Figure 5.9, it can be seen if the values of ϵ and ϑ are set to 0.995 and 20, respectively, the best effectiveness for both clustering the images and detecting the outliers can be obtained. Setting ϵ with the values smaller than 0.995 results in the removal of more RNs as outliers.

5.4.2.2 Hybrid Markov Clustering with Outliers

In the literature of the outlier detection, e.g., in [26], it has been mentioned that the number of outlier samples is much less than the number of other samples. However, in shared image analysis, it cannot always be true as users may share more single or cropped images than those applicable to camera fingerprinting.

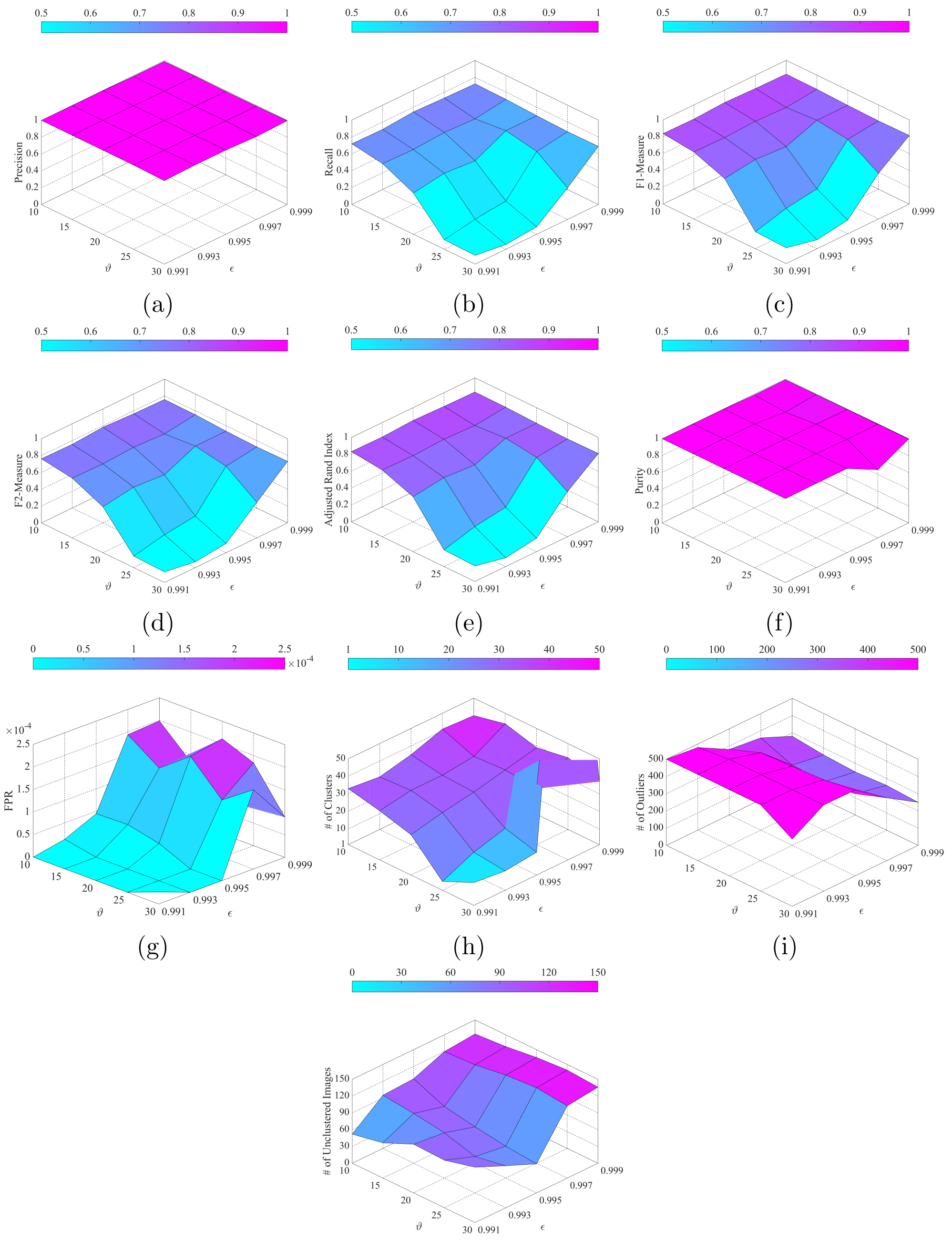


Figure 5.9: The parameters ϵ and ϑ of DBSCAN affect quality of outlier detection and clustering based on the HALO method on $\mathcal{V}_0^{\mathrm{NA}}$, perturbed by 500 outliers, (a) *Precision*, (b) *Recall*, (c) *F1-Measure*, (d) *F2-Measure*, (e) *Adjusted Rand Index*, (f) *Purity*, (g) *False Positive Rate*, (h) number of the obtained clusters, (i) number of the discovered outliers and (j) number of unclustered images.

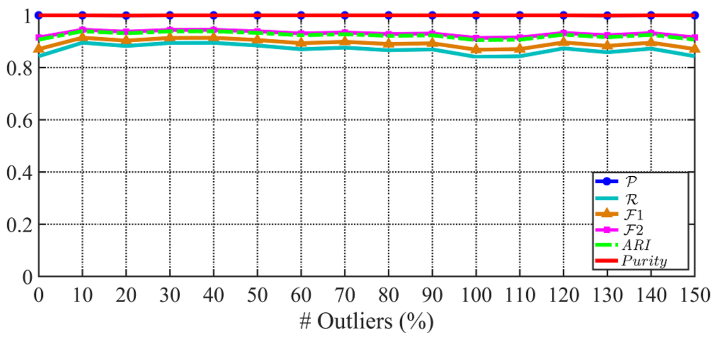
So, in this challenging case, stability of the clustering algorithm is important.

To test stability, we increase the number of outliers gradually and perform clustering based on the HALO method. To simulate the case where users upload outlier images, we select some images from $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, $\mathcal{V}^{\mathrm{FH}}$ and $\mathcal{V}^{\mathrm{FL}}$, and crop blocks with different sizes at different locations from each image, all are performed randomly. As the SPN is location-based, such a simple and efficient way of image cropping simulates the process of generating images acquired by different sensors. We apply the datasets $\mathcal{V}_1^{\mathrm{NA}}$, $\mathcal{V}_1^{\mathrm{W}}$, $\mathcal{V}_1^{\mathrm{FH}}$ and $\mathcal{V}_1^{\mathrm{FL}}$, each of them including 2250 images from identical smartphones. The datasets are perturbed by the generated outliers. More specifically, the outliers are increasingly added with the order 10%, 20%, 30%, ..., 150% of the 2250 images in the datasets. The results of the clustering on the perturbed datasets are shown in Figure 5.10. With increasing the number of outliers, the clustering is not affected, meaning that it is stable against the outliers even for the increase of 150%. The fine fluctuations in the graphs are related to the random selection of RNs to fill in each batch.

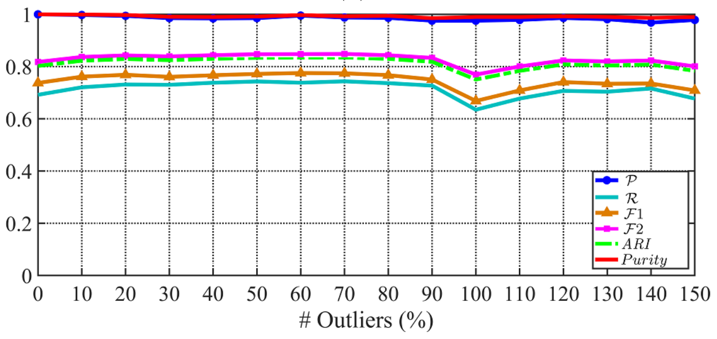
Table 5.14 presents the results of the HALO method on the datasets $\mathcal{V}_{\mathrm{NA}}$, $\mathcal{V}_{\mathrm{W}}$, $\mathcal{V}_{\mathrm{FH}}$ and $\mathcal{V}_{\mathrm{FL}}$ in terms of both clustering quality and outlier detection. We have set the number of outliers to 3000. The algorithm removed successfully the outliers with $\mathcal{N}_O$>90% and also clustered the remained RNs with high quality of different measures. To see the impact of the HALO method on non perturbed

Dataset	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	ARI	$Purity$	FPR	$\mathcal{N}_C$	$\mathcal{N}_O$	$\mathcal{N}_U$
$\mathcal{V}^{\mathrm{NA}}$	0.996	0.754	0.858	0.789	0.854	0.996	0.000	37/35	2836/3000	595/10480
$\mathcal{V}^{\mathrm{W}}$	0.907	0.613	0.732	0.602	0.725	0.970	0.001	30/35	2686/3000	783/10480
$\mathcal{V}^{\mathrm{FH}}$	0.981	0.601	0.738	0.640	0.732	0.989	0.000	30/35	2834/3000	407/10480
$\mathcal{V}^{\mathrm{FL}}$	0.796	0.301	0.433	0.277	0.423	0.876	0.002	14/35	2608/3000	569/10480

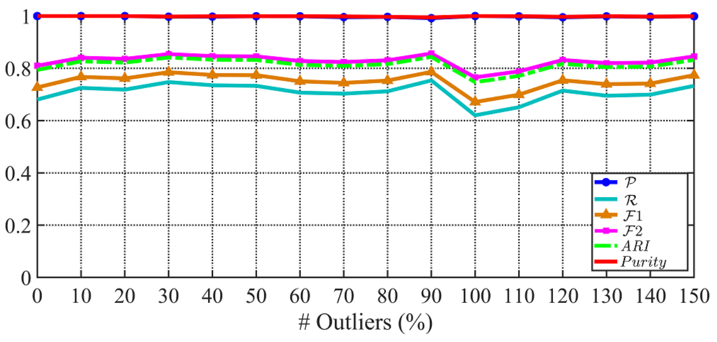
Table 5.14: Results (%) of HALO method on different datasets perturbed by 3000 outliers.



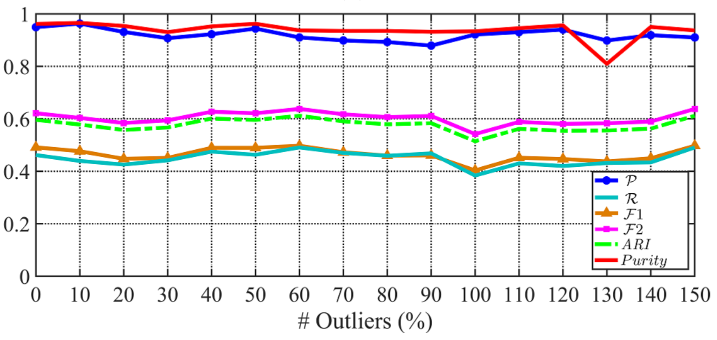
(a)



(b)



(c)



(d)

Figure 5.10: HALO method is stable with increasing the number of outliers in different datasets: (a) $\mathcal{V}_1^{\text{NA}}$, (b) $\mathcal{V}_1^{\text{W}}$, (c) $\mathcal{V}_1^{\text{FH}}$ and (d) $\mathcal{V}_1^{\text{FL}}$.

datasets and compare it with the HAL method, in Tables 5.15, 5.16, 5.17 and 5.18, the results of the clustering methods performed on different datasets are shown. From the obtained results, it can be implied that the HALO method is more effective when it is applied to high resolution datasets such as $\mathcal{V}^{\mathrm{NA}}$, $\mathcal{V}^{\mathrm{W}}$, and $\mathcal{V}^{\mathrm{FH}}$, because the outlier removal step purifies each batch and prevents from wrong assignment of contaminated or low resolution RNs to the obtained clusters through the clustering process. By removing the RNs, they are not computationally an overhead until the last iteration of the algorithm. In contrast, the HAL method is better for clustering low resolution datasets such as $\mathcal{V}^{\mathrm{FL}}$, since it does not remove the RNs and it gives them an extra chance to be clustered through the clustering process.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*	$\mathcal{N}_C$	$\mathcal{N}_O$	$\mathcal{N}_U$
HAL	0.992	0.720	0.834	0.756	0.830	0.994	**0.000**	**37/35**	—	725/7480
HALO	**0.999**	**0.759**	**0.863**	**0.796**	**0.859**	**0.999**	**0.000**	**37/35**	—	**423/7480**

Table 5.15: Comparison of HAL and HALO methods on $\mathcal{V}^{\mathrm{NA}}$ without perturbation.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*	$\mathcal{N}_C$	$\mathcal{N}_O$	$\mathcal{N}_U$
HAL	**0.964**	0.600	0.733	0.628	0.727	0.975	**0.000**	33/35	—	1601/7480
HALO	0.951	**0.666**	**0.783**	**0.678**	**0.778**	**0.985**	0.001	**34/35**	—	**470/7480**

Table 5.16: Comparison of HAL and HALO methods on $\mathcal{V}^{\mathrm{W}}$ without perturbation.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*	$\mathcal{N}_C$	$\mathcal{N}_O$	$\mathcal{N}_U$
HAL	0.962	**0.610**	**0.746**	0.636	**0.740**	0.975	**0.000**	**33/35**	—	1624/7480
HALO	**0.981**	0.605	0.742	**0.644**	0.733	**0.996**	0.001	32/35	—	**723/7480**

Table 5.17: Comparison of HAL and HALO methods on $\mathcal{V}^{\mathrm{FH}}$ without perturbation.

Method	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{F}2$	*ARI*	*Purity*	*FPR*	$\mathcal{N}_C$	$\mathcal{N}_O$	$\mathcal{N}_U$
HAL	0.750	**0.513**	**0.609**	**0.426**	**0.599**	0.847	0.005	**33/35**	—	1950/7480
HALO	**0.847**	0.342	0.488	0.333	0.478	**0.902**	**0.001**	19/35	—	**406/7480**

Table 5.18: Comparison of HAL and HALO methods on $\mathcal{V}^{\text{FL}}$ without perturbation.

5.5 Conclusion

In this Chapter, firstly, we have presented an SPN-based image clustering algorithm, called HAL to cluster shared images by users, without prior knowledge about the types and number of the related smartphones. Particularly, the HAL method exploits image resizing, hierarchical and graph-based clustering algorithms, and an adaptive threshold to cluster the images. We have shown that resizing the RNs to a specific resolution can tackle the problem of low resolution of the *shared images* compressed by social network platforms. Using Markov clustering, the hierarchical clustering is conducted in such a way that the representative clusters with a higher probability of belonging to the same camera are selected for merging. The adaptive threshold for merging the representative clusters depends on the quality of the obtained clusters. The threshold that is updated during the hierarchical algorithm can produce more precise clusters even for images from the same model of smartphones. The scalability of the HAL method by partitioning dataset into batches decrease the computational costs, in terms of memory usage. Partitioning dataset, exploiting the inherent sparseness of the correlation matrix, and checking only the representative clusters in the merging result in the calculation of a small portion (about 15%) of the full-pairwise correlation matrix, accelerating the clustering. Also, we have presented the HALO method which detects and clusters the perturbed datasets with a large number of outliers, i.e., single images from different camera sources or cropped images not applicable to fingerprinting their sources. The outlier detection is performed based on DBSCAN technique. We have shown that the HALO method is stable against the outliers. The results of the proposed methods on the VISION image dataset confirm their effectiveness and efficiency in comparison with the state-of-the-art clustering methods.

Chapter 6

Conclusions

The work presented in this thesis has concerned on shared image analysis based on camera Sensor Pattern Noise (SPN). Since SPN has been proven to be an effective and robust form of any device fingerprint, it has attracted considerable attention, due to its significant characteristics such as uniqueness to individual devices, stability over environmental conditions, and robustness against common image processing operations. It has successfully been applied to different data analysis tasks such as identifying the source device, linking devices, and linking profiles on social networks.

In Chapter 4, we have proposed clustering and classification based methods to achieve Smartphone Identification (SI) and User Profile Linking (UPL) tasks, given a set of images captured by a known number of smartphones and shared on a set of user profiles. The important outcome of the proposed methods is UPL across different social networks, where the clustered images from one social network are applied to classify shared images on another social network to fingerprint their sources. By the classified images and user profile tags, the shared images from the same source are linked and inter-layer UPL is achieved.

In Chapter 5, we have presented a **H**ybrid M**a**rkov C**l**ustering (HAL) method to cluster the images captured and shared by the users of social network platforms, without prior knowledge about the types and number of the related smartphones. Particularly, the HAL method exploits image resizing, hierarchical and

Markov clustering algorithms, and an adaptive threshold to cluster the images. We have shown that resizing the RNs to a specific resolution can tackle the problem of low resolution of the shared images compressed by social network platforms. Using Markov clustering, the hierarchical clustering is conducted in such a way that the representative clusters with a higher probability of belonging to the same camera are selected for merging. The adaptive threshold for merging the representative clusters depends on the quality of the obtained clusters. The threshold that is updated during the hierarchical algorithm can produce more precise clusters even for images from the same model of smartphones. The scalability of the algorithm by partitioning dataset into batches decreases the computational costs in terms of memory usage. Particularly, partitioning the dataset, exploiting the inherent sparseness of the correlation matrix, and checking only the representative clusters in the merging result in the calculation of a small portion of the full-pairwise correlation matrix, accelerating the clustering. Subsequently in Chapter 5, we have developed the HAL method and presented **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO) method. The HALO method detects and clusters the perturbed datasets with a large number of outliers, i.e., single images from different camera sources or cropped images not applicable to fingerprinting their sources. The outlier detection is performed based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique. Based on the obtained results, it has been confirmed that the HALO method is stable against the outliers.

Appendix A
Source Codes Description

Implementation of the proposed methods in this thesis was done in MATLAB®, version R2019a, on a laptop with the following characteristics: Intel Core i7-6500U (2.93 GHz), 16 GB of RAM, and Windows 10 operating system. The proposed methods have been evaluated on our image dataset[1] and the public benchmarking VISION image dataset[2]. Also, the source codes of the methods are available on GitHub[3]. The structure of the source codes is as follows:

- Residual Noise (RN) Extraction. The code is structured in 3 modules including:
  Module 1: image pre-processing
  Module 2: Block-Matching and 3D (BM3D) denoising
  Module 3: RN resizing
- Smartphone Identification (SI) and User Profile Linking (UPL) Method. The code is generally structured in 6 modules including:
  Module 1: correlation computation
  Module 2: Shared $\mathcal{K}$-Nearest Neighbors (SNN) computation
  Module 3: k-medoids Clustering
  Module 4: sensor pattern noise computation
  Module 5: Artificial Neural Network (ANN) classification

[1]The dataset is available from: `http://smartdata.cs.unibo.it/datasets#images`
[2]The dataset is available from: `https://lesc.dinfo.unifi.it/en/datasets`
[3]`https://tinyurl.com/y2zw8owy`

Module 6: evaluation

- **H**ybrid M**a**rkov C**l**ustering (HAL) Method (see Algorithm 2 in Subsection 5.3.2.4). The code is generally structured in 7 modules including:
  Module 1: batch partitioning
  Module 2: correlation computation
  Module 3: hierarchical clustering
  Module 4: Markov clustering
  Module 5: adaptive threshold computation and cluster merging
  Module 6: sensor pattern noise computation
  Module 7: evaluation

- **H**ybrid M**a**rkov C**l**ustering with **O**utliers (HALO) Method, (see Algorithm 4 in Subsection 5.4.1). The code is generally structured in 8 modules including:
  Module 1: batch partitioning
  Module 2: correlation computation
  Module 3: Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
  Module 4: hierarchical clustering
  Module 5: Markov clustering
  Module 6: adaptive threshold computation and cluster merging
  Module 7: sensor pattern noise computation
  Module 8: evaluation

Bibliography

[1] F. Norouzizadeh Dezfouli, A. Dehghantanha, B. Eterovic-Soric, and K.-K. R. Choo, “Investigating social networking applications on smartphones detecting facebook, twitter, linkedin and google+ artefacts on android and ios platforms,” *Australian Journal of Forensic Sciences*, vol. 48, no. 4, pp. 469–488, 2016. 1, 25

[2] Q. Liu, X. Li, L. Chen, H. Cho, P. Cooper, Z. Chen, M. Qiao, and A. Sung, “Identification of smartphone-image source and manipulation,” *Advanced Research in Applied Artificial Intelligence*, pp. 262–271, 2012. 1

[3] N. Huang, J. He, N. Zhu, X. Xuan, G. Liu, and C. Chang, “Identification of the source camera of images based on convolutional neural network,” *Digital Investigation*, vol. 26, pp. 72–80, 2018. 1

[4] I. J. Cox, M. L. Miller, J. A. Bloom, and C. Honsinger, *Digital watermarking*, vol. 53. Springer, 2002. 2

[5] J. A. Redi, W. Taktak, and J.-L. Dugelay, “Digital image forensics: a booklet for beginners,” *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 133–162, 2011. 2

[6] J. Lukas, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006. 2, 10, 11

[7] X. Lin, *Digital Image Forensics Based on Sensor Pattern Noise.* PhD thesis, University of Warwick, 2016. 3, 11

[8] N. Yager and A. Amin, "Fingerprint classification: a review," *Pattern Analysis and Applications*, vol. 7, no. 1, pp. 77–93, 2004. 3

[9] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli, "Linking accounts across social networks: the case of stackoverflow, github and twitter.," in *KDWeb*, pp. 41–52, 2015. 5

[10] F. Bertini, R. Sharma, A. Iannì, and D. Montesi, "Smartphone verification and user profiles linking across social networks by camera fingerprinting," in *International Conference on Digital Forensics and Cyber Crime*, pp. 176–186, Springer, 2015. 6

[11] F. Bertini, R. Sharma, A. Iannì, and D. Montesi, "Profile resolution across multilayer networks through smartphone camera fingerprint," in *Proceedings of the 19th International Database Engineering & Applications Symposium*, pp. 23–32, ACM, 2015. 6, 26

[12] F. Bertini, R. Sharma, A. Iannì, and D. Montesi, "Social media investigations using shared photos," in *The International Conference on Computing Technology, Information Security and Risk Management (CTISRM2016)*, p. 47, 2016. 6

[13] R. Rouhi, F. Bertini, and D. Montesi, "A cluster-based approach of smartphone camera fingerprint for user profiles resolution within social network," in *Proceedings of the 22nd International Database Engineering & Applications Symposium*, pp. 287–291, ACM, 2018. 6, 26, 32

[14] R. Rouhi, F. Bertini, D. Montesi, and C.-T. Li, "Social network forensics through smartphones and shared images," in *2019 7th International Workshop on Biometrics and Forensics (IWBF)*, pp. 1–6, IEEE, 2019. 6, 26

[15] T. S. Madhulatha, “Comparison between k-means and k-medoids clustering algorithms,” in *Advances in Computing and Information Technology*, pp. 472–481, Springer, 2011. 8

[16] M. Kirchner and R. Böhme, “Synthesis of color filter array pattern in digital images,” in *Media Forensics and Security*, vol. 7254, p. 72540K, International Society for Optics and Photonics, 2009. 10

[17] X. Lin and C.-T. Li, “Preprocessing reference sensor pattern noise via spectrum equalization,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 126–140, 2016. 10, 12

[18] S. Taspinar, M. Mohanty, and N. Memon, “Prnu-based camera attribution from multiple seam-carved images,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3065–3080, 2017. 10

[19] J. Lukáš, J. Fridrich, and M. Goljan, “Determining digital image origin using sensor imperfections,” in *Proc. SPIE Electronic Imaging, Image and Video Communication and Processing*, vol. 5685, pp. 249–260, 2005. 11, 12

[20] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-d transform-domain collaborative filtering,” *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007. 11

[21] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008. 12, 32

[22] G. Chierchia, S. Parrilli, G. Poggi, C. Sansone, and L. Verdoliva, “On the influence of denoising in prnu based forgery detection,” in *Proceedings of the 2nd ACM workshop on Multimedia in Forensics, Security and Intelligence*, pp. 117–122, ACM, 2010. 12

[23] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014. 12

[24] F. Schwenker and E. Trentin, "Pattern classification and clustering: A review of partially supervised learning approaches," *Pattern Recognition Letters*, vol. 37, pp. 4–14, 2014. 13

[25] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969. 13

[26] A. Taha and A. S. Hadi, "Anomaly detection methods for categorical data: A review," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, p. 38, 2019. 13, 86

[27] H. Liu, J. Li, Y. Wu, and Y. Fu, "Clustering with outlier removal," *arXiv preprint arXiv:1801.01899*, 2018. 13, 23, 24

[28] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, Dec 1985. 15

[29] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable.," in *NDSS*, 2014. 17

[30] O. Willers, C. Huth, J. Guajardo, and H. Seidel, "Mems-based gyroscopes as physical unclonable functions.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 261, 2016. 17

[31] R. Jin, L. Shi, K. Zeng, A. Pande, and P. Mohapatra, "Magpairing: Pairing smartphones in close proximity using magnetometers," *IEEE Transactions*

on Information Forensics and Security, vol. 11, no. 6, pp. 1306–1320, 2016. 17

[32] I. Amerini, R. Becarelli, R. Caldelli, A. Melani, and M. Niccolai, "Smartphone fingerprinting combining features of on-board sensors," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 2457–2466, 2017. 17

[33] E. J. Alles, Z. J. Geradts, and C. J. Veenman, "Source camera identification for low resolution heavily compressed images," in *Computational Sciences and Its Applications, 2008. ICCSA'08. International Conference on*, pp. 557–567, IEEE, 2008. 17

[34] A. Das, N. Borisov, and M. Caesar, "Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 441–452, ACM, 2014. 17

[35] T. Van Lanh, K.-S. Chong, S. Emmanuel, and M. S. Kankanhalli, "A survey on digital camera image forensic methods," in *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 16–19, IEEE, 2007. 17, 19

[36] K. San Choi, E. Y. Lam, and K. K. Wong, "Source camera identification using footprints from lens aberration.," in *Digital Photography*, p. 60690J, 2006. 17

[37] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on cfa interpolation," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3, pp. III–69, IEEE, 2005. 18

[38] O. Celiktutan, I. Avcibas, B. Sankur, and N. Memon, "Source cell-phone identification," *IEEE Signal Processing and Communications Applications*, pp. 1–3, 2005. 18

[39] J. Lukáš, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006. 18

[40] L.-H. Chan, N.-F. Law, and W.-C. Siu, "A confidence map and pixel-based weighted correlation for prnu-based camera identification," *Digital Investigation*, vol. 10, no. 3, pp. 215–225, 2013. 18

[41] F. Gisolf, A. Malgoezar, T. Baar, and Z. Geradts, "Improving source camera identification using a simplified total variation based noise removal algorithm," *Digital Investigation*, vol. 10, no. 3, pp. 207–214, 2013. 18

[42] G. Cattaneo, U. F. Petrillo, M. Nappi, F. Narducci, and G. Roscigno, "An efficient implementation of the algorithm by lukáš et al. on hadoop," in *International Conference on Green, Pervasive, and Cloud Computing*, pp. 475–489, Springer, 2017. 19

[43] M. Kharrazi, H. T. Sencar, and N. Memon, "Blind source camera identification," in *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 1, pp. 709–712, IEEE, 2004. 19

[44] O. Çeliktutan, B. Sankur, and I. Avcibas, "Blind identification of source cell-phone model.," *IEEE Trans. Information Forensics and Security*, vol. 3, no. 3, pp. 553–566, 2008. 19

[45] A. S. Orozco, D. A. González, J. R. Corripio, L. G. Villalba, and J. C. Hernandez-Castro, "Source identification for mobile devices, based

on wavelet transforms combined with sensor imperfections," *Computing*, vol. 96, no. 9, pp. 829–841, 2014. 19

[46] M.-J. Tsai and G.-H. Wu, "Using image features to identify camera sources," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2, pp. II–II, IEEE, 2006. 19

[47] L. Baroffio, L. Bondi, P. Bestagini, and S. Tubaro, "Camera identification with deep convolutional networks," *arXiv preprint arXiv:1603.01068*, 2016. 19, 20

[48] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *2016 IEEE International workshop on information forensics and security (WIFS)*, pp. 1–6, IEEE, 2016. 19, 20

[49] D. Freire-Obregón, F. Narducci, S. Barra, and M. Castrillón-Santana, "Deep learning for source camera identification on mobile devices," *Pattern Recognition Letters*, vol. 126, pp. 86–91, 2019. 20

[50] G. J. Bloy, "Blind camera fingerprinting and image clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 532–534, 2008. 20, 63

[51] C.-T. Li, "Unsupervised classification of digital images using enhanced sensor pattern noise," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3429–3432, IEEE, 2010. 20

[52] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti, "Fast image clustering of unknown source images," in *2010 IEEE International Workshop on Information Forensics and Security*, pp. 1–5, IEEE, 2010. 21

[53] L. J. G. Villalba, A. L. S. Orozco, R. R. López, and J. H. Castro, "Identification of smartphone brand and model via forensic video analysis," *Expert Systems with Applications*, vol. 55, pp. 59–69, 2016. 21, 77, 78

[54] B.-b. Liu, H.-K. Lee, Y. Hu, and C.-H. Choi, "On classification of source cameras: A graph based approach," in *2010 IEEE International Workshop on Information Forensics and Security*, pp. 1–5, IEEE, 2010. 21

[55] I. Amerini, R. Caldelli, P. Crescenzi, A. Del Mastio, and A. Marino, "Blind image clustering based on the normalized cuts criterion for camera identification," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 831–843, 2014. 21

[56] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. 21

[57] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "Correlation clustering for prnu-based blind image source identification," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, 2016. 22

[58] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "Blind prnu-based image clustering for source identification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2197–2211, 2017. 22, 77, 78

[59] C.-T. Li and X. Lin, "A fast source-oriented image clustering method for digital forensics," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 69, 2017. 22, 35, 77, 78

[60] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the*

2003 SIAM International Conference on Data Mining, pp. 47–58, SIAM, 2003. 22, 35

[61] X. Lin and C.-T. Li, “Large-scale image clustering based on camera fingerprints,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 793–808, 2017. 22, 23, 32, 58, 63, 65, 66

[62] Q.-T. Phan, G. Boato, and F. G. De Natale, “Accurate and scalable image clustering based on sparse representation of camera fingerprint,” *IEEE Transactions on Information Forensics and Security*, 2018. 22, 24

[63] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” in *ACM sigmod record*, vol. 29, pp. 93–104, ACM, 2000. 23

[64] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, “Enhancing effectiveness of outlier detections for low density patterns,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 535–548, Springer, 2002. 23

[65] K. Zhang, M. Hutter, and H. Jin, “A new local distance-based outlier detection approach for scattered real-world data,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 813–822, Springer, 2009. 23

[66] Z. He, X. Xu, J. Z. Huang, and S. Deng, “Fp-outlier: Frequent pattern based outlier detection.,” *Comput. Sci. Inf. Syst.*, vol. 2, no. 1, pp. 103–118, 2005. 23

[67] H.-P. Kriegel, M. Schubert, and A. Zimek, “Angle-based outlier detection in high-dimensional data,” in *Proceedings of the 14th ACM SIGKDD inter-*

national conference on Knowledge discovery and data mining, pp. 444–452, ACM, 2008. 23

[68] N. Pham and R. Pagh, "A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 877–885, ACM, 2012. 23

[69] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008. 23

[70] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 7, pp. 1460–1470, 2012. 23

[71] R. Kannan, H. Woo, C. C. Aggarwal, and H. Park, "Outlier detection for text data," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 489–497, SIAM, 2017. 23

[72] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International Conference on Machine Learning*, pp. 4393–4402, 2018. 23

[73] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*, pp. 146–157, Springer, 2017. 23

[74] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient gan-based anomaly detection," *arXiv preprint arXiv:1802.06222*, 2018. 23

[75] D. Li, D. Chen, J. Goh, and S.-k. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *arXiv preprint arXiv:1809.04758*, 2018. 23

[76] S. Chawla and A. Gionis, "k-means–: A unified approach to clustering and outlier detection," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 189–197, SIAM, 2013. 24

[77] L. Ott, L. Pang, F. T. Ramos, and S. Chawla, "On integrated clustering and outlier detection," in *Advances in neural information processing systems*, pp. 1359–1367, 2014. 24

[78] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, "Algorithms for facility location problems with outliers," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pp. 642–651, Society for Industrial and Applied Mathematics, 2001. 24

[79] K. Chen, "A constant factor approximation algorithm for k-median clustering with outliers," in *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 826–835, Citeseer, 2008. 24

[80] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996. 24, 80

[81] F. Saki and N. Kehtarnavaz, "Online frame-based clustering with unknown number of clusters," *Pattern Recognition*, vol. 57, pp. 70–83, 2016. 24

[82] N. Al Mutawa, I. Baggili, and A. Marrington, "Forensic analysis of social networking applications on mobile devices," *Digital Investigation*, vol. 9, pp. S24–S33, 2012. 25

[83] L. Jamjuntra, P. Chartsuwan, P. Wonglimsamut, K. Porkaew, and U. Supasitthimethee, "Social network user identification," in *Knowledge and Smart Technology (KST), 2017 9th International Conference on*, pp. 132–137, IEEE, 2017. 25

[84] W. E. Winkler, "String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.," 1990. 25

[85] T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff, "Identifying users across social tagging systems.," in *ICWSM*, 2011. 25

[86] S. Bartunov, A. Korshunov, S.-T. Park, W. Ryu, and H. Lee, "Joint link-attribute user identity resolution in online social networks," in *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis*, ACM, 2012. 25

[87] E. Raad, R. Chbeir, and A. Dipanda, "User profile matching in social networks," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pp. 297–304, IEEE, 2010. 25

[88] S. Gupta and M. Rogers, "Using computer behavior profiles to differentiate between users in a digital investigation," 2016. 25

[89] R. Zafarani, L. Tang, and H. Liu, "User identification across social media," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 2, p. 16, 2015. 25, 26

[90] F. M. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli, "Where you are is who you are: User identification by matching statistics.," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 2, pp. 358–372, 2016. 25

[91] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *Acm Sigkdd Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017. 26

[92] R. Rouhi, F. Bertini, D. Montesi, X. Lin, Y. Quan, and C.-T. Li, "Hybrid clustering of shared images on social networks for digital forensics," *IEEE Access*, vol. 7, pp. 87288–87302, 2019. 26

[93] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009. 29, 34

[94] J. Mander, "Gwi social summary," *Global Web Index*, 2017. 30

[95] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva, "Vision: a video and image dataset for source identification," *EURASIP Journal on Information Security*, vol. 2017, no. 1, p. 15, 2017. 31, 40, 58

[96] X. Lin and C.-T. Li, "Rotation-invariant binary representation of sensor pattern noise for source-oriented image and video clustering," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, IEEE, 2018. 32

[97] R. E. Carlson and F. N. Fritsch, "An algorithm for monotone piecewise bicubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 26, no. 1, pp. 230–238, 1989. 33

[98] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: a review," in *International Conference on Computational Science and Its Applications*, pp. 707–720, Springer, 2014. 34, 59

[99] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982. 34

[100] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967. 34

[101] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009. 34

[102] S. Chauhan and S. Dhingra, "Pattern recognition system using mlp neural networks," *Pattern Recognition*, vol. 4, no. 9, pp. 43–46, 2012. 37

[103] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018. 37

[104] J. A. Freeman and D. M. Skapura, *Algorithms, Applications, and Programming Techniques.* Addison-Wesley Publishing Company, USA, 1991. 37

[105] S. Haykin, *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994. 37

[106] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009. 37

[107] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross validation, encyclopedia of database systems (edbs)," *Arizona State University, Springer*, p. 6, 2009. 39

[108] S. Van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 121–141, 2008. 59

[109] U. Mesa, "Regularized markov clustering algorithm on protein-protein interaction networks using ellpack-r sparse data format," *Undergraduated thesis, University of Indonesia*, 2012. 59

[110] S. Varia, *Regularized Markov Clustering in MPI and Map Reduce.* PhD thesis, The Ohio State University, 2013. 59

[111] V. M. Satuluri, *Scalable clustering of modern networks.* PhD thesis, The Ohio State University, 2012. 60, 62

[112] A. Bustamam, K. Burrage, and N. A. Hamilton, "Fast parallel markov clustering in bioinformatics using massively parallel computing on gpu with cuda and ellpack-r sparse format," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 9, no. 3, pp. 679–692, 2012. 61