A Classification Engine for Image Ballistics of Social Data

Oliver Giudice*†, Antonino Paratore†, Marco Moltisanti* and Sebastiano Battiato*†

*Dipartimento di Matematica e Informatica, University of Catania,

Email: giudice,moltisanti,battiato@dmi.unict.it

†iCTLab s.r.l. - Spinoff University of Catania

mail: antonino.paratore@ictlab.srl

Abstract—**Image Forensics has already achieved great results for the source camera identification task on images. Standard approaches for data coming from Social Network Platforms cannot be applied due to different processes involved (e.g., scaling, compression, etc.). Over 1 billion images are shared each day on the Internet and obtaining information about their history from the moment they were acquired could be exploited for investigation purposes. In this paper, a classification engine for the reconstruction of the history of an image, is presented. Specifically, exploiting K-NN and decision trees classifiers and a-priori knowledge acquired through image analysis, we propose an automatic approach that can understand which Social Network Platform has processed an image and the software application used to perform the image upload. The engine makes use of proper alterations introduced by each platform as features. Results, in terms of global accuracy on a dataset of 2720 images, confirm the effectiveness of the proposed strategy.**

Index Terms—**Social networks, Image Forensics, JPEG, Digital Ballistics.**

I. INTRODUCTION

Image Forensics traditionally refers to a number of different tasks on digital images aiming at producing evidence on the authenticity and integrity of data (e.g., forgery detection) and on the identification of the acquisition device (camera identification) [1],[2]. Digital images are continuously altered starting from the moment they were acquired. Most of the time, these alterations are made by users with precise malicious intents. Typical tamperings are the removal or the insertion of an object in an image, the cropping of an undesired portion of a picture, or the application of particular filters to modify or mask sensible parts (e.g., faces in pedo-pornographic photos). When the tampering is not clearly visible, the problem of detecting it becomes obviously challenging. To solve the forgery detection task, some approaches stand above the others: a group of them looks at the structure of the file (e.g., JPEG blocking artifacts analysis [3], [4], hash functions [5], JPEG headers analysis [6], thumbnails [7] and EXIF analysis [8], etc.); others try to identify the device that acquired the image by making use of PRNU patterns ([9],[10]), or focus on statistical analysis of the DCT coefficients ([11], [12], [13]). A voting approach has been used for the same purpose in [5]. Another important task for Image Forensics is finding the camera device that acquired the image. Some in-depth studies ([14] [15]) showed that it is possible to coarsely solve the camera identification task, using the DCT coefficients as a feature. All these works make clear the importance of the JPEG pipeline in retrieving information about the history of an image.

Nowadays Social Networks allow their users to upload and share large amounts of images: just on *Facebook* about 1 billion images are shared every day. What happens when a picture is shared on a social platform? How does the upload process affect the JPEG elements of the image? A Social Network is yet but another piece of software that alters images for bandwidth, storage and layout reasons. These alterations have been proved to make state-of-the-art approaches for camera identification less precise and reliable. Recent studies ([16],[17]) have shown that, although the platform heavily modifies an image, this processing leaves a sort of finger-print on the image itself. This evidence can be exploited to understand if the image has been actually uploaded to the *Facebook* platform. State of the art studies, regarding Social Platforms, focus on the analysis of alterations on images and do not propose a method to solve the image ballistics task. Moreover, they focus on too few Social Network platforms making their works not general enough. Hence, enlarging the analysis to further Social Network Services (SNSs), the reconstruction of the history of an image becomes a difficult task. To understand how SNSs process images, a dataset of images from different camera devices was collected, under controlled conditions. We selected ten SNSs through which we processed the collected images by mean of an upload and download process. By doing this, a dataset of images has been obtained, in order to identify any alterations on JPEG elements. The main discovery of our study was that alterations observed are platform dependent (server-side) but also related to the application carrying out the upload (client-side). This evidence can be fundamental for investigation purposes to understand not only the provenience of an image, but also if it has been uploaded from a given device (e.g., Android, iOS). All the observed alterations allowed to build an automatic classifier, based on two K-NN classifiers and a decision tree fitted on the built dataset. Starting from an input image, the proposed approach can predict the SNS that processed the image and the client application through which the image has been uploaded. The remainder of the paper is structured as follows: in Sec. II, we describe how the dataset has been built, which social platforms have been considered and what kind of upload methods have been used; in Sec.

III, an in-depth analysis on dataset images is reported in order to find alterations that can be coded into a fingerprint for a SNS processing; in Sec. IV, our approach for image ballistics on social image data is presented with the obtained classification results. Finally, conclusions and reasoning about possible future works on the topic are discussed.

II. A Dataset of Social Imagery

The alterations introduced on images by SNS can be thought as a unique fingerprint left by the SNS. The aim of our study is to discover those fingerprints by analyzing the behavior of the most popular SNSs that allow image sharing. Hence, 10 platforms have been selected. First of all, *Facebook* (http://www.facebook.com) and *Google+* (http://plus.google.com) were taken into account as being the two most popular platforms where users can share their statuses and multimedia content to a network of friends. *Twitter* (http://www.twitter.com) and *Tumblr* (http://www.tumblr.com) were considered as being representative of the micro-blogging concept. We included also *Flickr* (https://www.flickr.com) and *Instagram* (https://www.instagram.com) as platforms focused on sharing high quality artistic photos with capabilities of image editing and filtering. *Imgur* (http://www.imgur.com) and *Tinypic* (htto://www.tinypic.com) were also taken into consideration even if they are not properly SNSs but are very popular platforms for image sharing: users usually link images hosted on them from forums and web sites all over the Internet. Finally *WhatsApp* (http://www.whatsapp.com) and *Telegram* (http://www.telegram.org) were also selected as being the two most popular mobile messaging platforms that, by allowing users to create chat groups, are another big place for image sharing on the Internet. Specifically, the last two services are often involved in forensic investigations.

To discover how SNSs process images, we collected a set of photos with the camera devices listed in Table I. Images were acquired representing three different types of scenes: outdoor scenes with buildings (artificial environment), outdoor scenes without buildings (natural environment) and indoor scenes. When taking a picture, we captured two versions: a High Quality (HQ) photo at the maximum resolution allowed by the device, and a Low Quality (LQ) photo (see also Table I). Capturing images in this way, a dataset with a good variability in terms of contents and resolutions was obtained.

Images collected so far were uploaded to each of the considered platforms with two different methods: with a web browser, and with iOS and Android native apps. No further discrimination is needed for web browsers because we observed that alterations are not browser-dependent. Each download was performed by searching for the image file URL in the HTML code of the page showing the image itself. At the end of this phase 2400 images were properly collected.

The second upload method was carried out with iOS and Android native apps of each social platform, except for *Tinypic* that do not possess an official app in stores. Moreover, the upload has been done by choosing images in two ways: by searching in the gallery for a previously acquired image (images from local gallery) and by acquiring the image with the camera app embedded in the app itself (embedded camera app). After uploading all images as described above, all of them were downloaded through the "URL searching technique" previously described. 320 more images processed through 8 platforms were thus obtained. All uploads were performed with default settings.

The overall dataset consists of 2720 images in JPEG format and it is available at the following web address http://iplab.dmi.unict.it/DigitalForensics/social_image_forensics/.

III. Dataset Analysis

The main aim of our work is to find a fingerprint left by SNSs on JPEG structure elements, after an upload/download process, in order to build a classifier for image ballistics. To achieve this goal, all information contained in the JPEG file specification has been analyzed: image filename, image size, meta-data and JPEG compression information. We observed that each upload/download process through the considered SNSs produces different alterations among the above-mentioned elements that could be taken into account as fingerprints of the process itself. Details of these alterations will be described in the following Subsections.

A. Image Filename Alterations

The analysis of the filename of an image and the comparison with known patterns during an investigation on storage devices can provide information about the platform from which it could be downloaded and the date when it was uploaded. For this reason, we first evaluated if and how each platform modifies the file name. We observed that all platforms except *Google+* do a rename.

As an example, in Table II the new names for an uploaded file with name "IMG_2641.jpg" are reported. The column "image lookup" describes the presence into the new filename of an ID useful to reconstruct an URL that points to the web location where the image file is stored.

Facebook, *Flickr*, *Tumblr* and *Instagram* use the image ID and the platform public API (e.g., Graph for *Facebook*) to build the corresponding URL. *Twitter* and *Imgur* allow finding the image on the respective platform by navigating to the URL:

- https://pbs.twimg.com/media/<IMAGE ID> for *Twitter*;
- http://imgur.com/<IMAGE ID> for *Imgur*;

The other platforms do not present an image ID.

B. Image Size Alterations

A stronger evidence than file naming is the resize of the uploaded images on each platform. A fine-grained test was performed by using synthetic images derived from our dataset and resized at different scales.

On most platforms, resizing is applied if and only if the input image matches certain conditions. This condition is linked to the length in pixels of the longest side M of the original image, where $M = max(width, height)$. If M is greater than a threshold, a resizing algorithm is applied and the resulting image has its longest size equal to the threshold. In

Model	Device Type	Low Resolution	High Resolution
Canon Eos 650D	Dedicated device	720x480	5184x3456
QUMOX SJ4000	Dedicated device	640x480	4032x3024
Sony Powershot A2300	Dedicated device	640x480	4608x3456
Samsung Galaxy Note 3 Neo	Android 4 Phone	640x480	3264x2448
HTC Desire 526g	Android 5 Phone	640x480	3264x2448
Huawei G Play Mini	Android 6 Phone	640x480	4208x3120
iPhone 5	iOS 6 Phone	640x480	2448x3264
iPad mini 2	iOS 8 Pad	640x480	800x600

Social	Rename (image ID in bold)	Image Lookup	Other information
Facebook	11008414_**746657488782610**_8508378989307666639_n.jpg	YES	Upload resolution
Flickr	26742193671_**8a63f10c85**_h.jpg	YES	Download resolution (h=1600)
Tumblr	tumblr_**o3q9ghRCRh1vnf44lo9**_1280.jpg	YES	Download resolution (1280)
Imgur	04 - **Dw0KXG2**.jpg	YES	
Twitter	**CdqCPQ-WAAAzrHI.jpg**	YES	
WhatsApp	IMG-20160314-WA0038.jpg	NO	Receiving Date (2016-03-14)
Tinypic	1zqdirm.jpg	NO	
Instagram	1689555_**169215806798447**_744040439_n.jpg	YES	Upload Resolution
Telegram	422114602_5593965449613038107.jpg	NO	

Social	EXIF		File Size		JPEG Compression	
	Camera Data	Other Data	Resize	Resize Condition	Re-Compression	Re-Compression Condition
Facebook	Delete	Delete	Yes	LQ: $M > 960$ HQ: $M > 2048$	Yes	Always
Google+	Maintain	Maintain/Edit	Yes	$M > 2048$	Yes	$M > 2048$
Flickr	Delete	Maintain/Edit	Yes	Depends on options	Yes	Depends on options
Tumblr	Maintain	Maintain/Edit	Yes	$M > 1280$	Yes	$M > 1280$
Imgur	Delete	Delete	No	Never	Yes	Image Size (MB) > 5.45 MB
Twitter	Delete	Delete	Yes	$M > 2048$	Yes	Always
whatsApp	Delete	Delete	Yes	$M > 1600$	Yes	Always
Tinypic	Maintain	Maintain/Edit	Yes	$M > 1600$	Yes	$M > 1600$
Instagram	Delete	Delete	Yes	$M > 1080$	Yes	Always
Telegram	Delete	Delete	Yes	$M > 2560$	Yes	Always

Table III, such conditions and the corresponding thresholds for each platform are reported. *Tumblr* does not rescale uploaded images, while in *Flickr* the threshold is set by the user. When the images are resized, the longest side will be set to a fixed value that identifies, in some sense, the platform that made the operation (see Table III). Let note that, some of the considered platforms use the same threshold value and it is subject to changes over time (for example, during our experiments, the threshold for *Twitter* changed from 1024 to 2048).

C. Meta-data Alterations

The best evidence to obtain information, for investigation purposes, are meta-data embedded in JPEG files. These meta-data are technically known as EXIF and can store information like the device that acquired an image, the date and time of acquisition and also the GPS coordinates. For our purposes, EXIF data were divided into two categories: "camera data" which co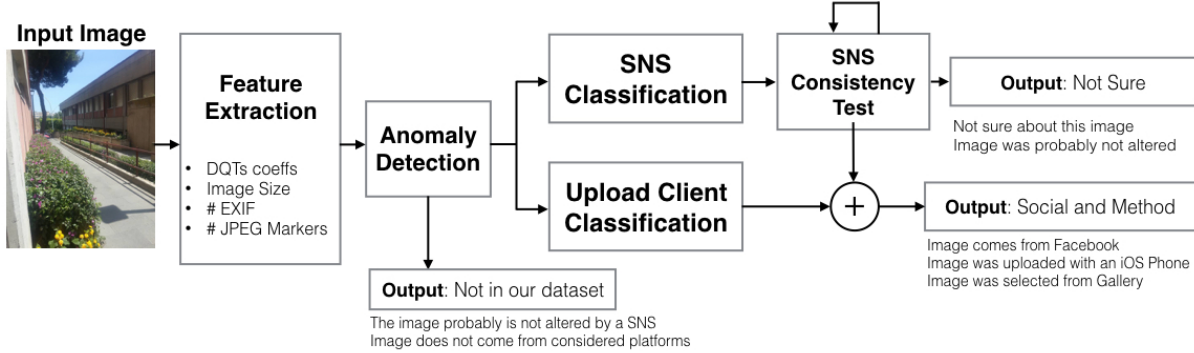ntains all those key-valued that allow to identifying the device that acquired the image and "other data" for every other information.

In Table III, the results of the analysis on EXIF data are resumed for each platform. In particular, it is reported if "camera data" and "other data" are deleted, maintained or just edited throughout the processing. Unfortunately, most of the SNSs delete all meta-data, specifically those related to camera data.

D. Image JPEG Compression Alterations

The images considered in our dataset are all encoded in JPEG format, both the original versions and the downloaded ones. Thus, an analysis on how the SNS processing affects the JPEG compression has been carried out. We focused on the Discrete Quantization Tables (DQTs) used for JPEG compression (extracted by DJPEG: an open source tool part of libjpeg project [18]).

Fig. 1. Classification scheme for Image Ballistics in the era of Social Network Services. The proposed approach encodes JPEG information from an input image into a feature vector. The obtained feature vector is evaluated through an Anomaly Detector that filters out images not processed by a SNS. If the input image is not an anomaly, the feature vector goes through other two classifiers: a SNS Classifier and an Upload Client Classifier. The output of the SNS Classifier is further processed through a SNS Consistency Test that checks if the features of the input image and the predicted SNS are consistent to re-compression and resizing conditions. The final output depends on this last stage: if all features are compatible with the classified SNS then the obtained prediction, joined with upload client prediction, is outputted. Otherwise the consistency test is repeated, for the next most probable predicted SNS, until it is satisfied or it stalls on the same predicted platform. In this case the overall output will be "Not Sure".

Considering how platforms affect DQTs, it is possible to divide them into two categories:

- Platforms that always re-compress images (*Facebook*, *Twitter*, *Telegram*, *WhatsApp*, *Instagram*);
- Platforms that re-compress images at a given condition (*Google+*, *Tumblr*, *Tinypic*, *Imgur*).

The compression follows the same rules we described for resizing. In fact, a threshold-based evaluation is performed on the longest image side and, if it is bigger than the threshold, the image is compressed using a DQT that will be different from the original one. This is not true for all the considered platforms; *Flickr* allows the user to choose the threshold (if any), while on *Imgur* the threshold is fixed in terms of size in MegaBytes; specifically, if the input image size is greater than 5.45MB, than the re-compression is performed, otherwise nothing happens (see also Table III).

IV. IMAGE BALLISTICS OF SOCIAL DATA

Starting from the results of the analysis reported in previous Sections, regarding the alterations on JPEG elements of processed images, it is possible to assess that such alterations bring pieces of information about the history of the image but they could be insufficient, if considered alone, for investigation purposes. Hence, we represent all the observed alterations into a set of features to be used as input for an automatic classifier. The following elements are then embedded into proper numerical features:

- The DQTs coefficients divided in 64 coefficients for the Chrominance table and 64 for the Luminance one, which represent the JPEG compression alterations;
- Image size (width and height in pixels), which brings information about size alterations;
- Number and typology of EXIF data (key-value couples), which describes meta-data alterations;
- Number of markers in JPEG files as defined in [19].

The listed features were chosen in order to represent each kind of alteration described in previous Sections.

The Quality Factor (QF) was not considered among them, as it was done in [16], for being dependent on DQTs coefficients and thus not bringing any new useful information. In particular, QF does not have a unique method to be computed and this can be be a source of error for classification purposes.

PRNU was also not taken into consideration among our features, because, as already mentioned, the heavy processing done on images by SNSs destroys/modifies any information coming from the sensor which acquired the image.

Given the features listed before and the image dataset described so far, a correspondence between features and the SNS has been established. This is particularly true for platforms that always operate a re-compression and heavily alter images. Starting from this correspondence, an automatic classification approach for image ballistics was built. Given an input image, the proposed method allows knowing not only from which platform it comes from, but also which client application was used to upload the image (browser web application, iOS native app or Android native app). Moreover, for images uploaded from iOS and Android native apps, the proposed approach is able to differentiate between images taken from the camera application embedded into the native apps or images chosen through gallery selection. This demonstrates that fingerprints observable on images are left both by SNSs and the client applications carrying out the upload.

A. Implementing image ballistics: a classification engine

Given a JPEG image I, our objectives are to define:

1) if there is a compatibility between the non-related JPEG elements of I (i.e. filename, EXIF data) and the processing pipeline of SNSs;
2) if there is a compatibility between the JPEG elements of I and the processing pipeline of SNSs;
3) which SNS is compatible with the JPEG elements of the image, with a certain degree of confidence, and what is the uploading source in terms of operating system (OS) and application.

We represent each image I as a 44-dimensional vector

$$\boldsymbol{v} = \{w, h, |E|, m, l_j, c_k\}, \qquad (1)$$

where

- $w \times h$ is the size in pixels of I;
- $E = \{key, value\}$ is an associative array containing the EXIF metadata, thus $\|E\|$ is the number of metadata found in the structure of I;
- m is the number of JPEG markers in I;
- $l_j, j = 0, \ldots, 31$ are the first 32 coefficients of the luminance quantization table;
- $c_k, k = 0, \ldots, 7$ are the first 8 coefficients of the chrominance quantization table.

Moreover, we define $fn(I)$ as the filename of the image I.

At the first stage, we consider $fn(I)$ and E. If there is a matching between $fn(I)$ and the renaming patterns observed in Section III-A, our approach confirms the compatibility between I and the SNS with the matched pattern. Also, E is taken into account, looking for the "Exif.Image.UniqueCameraModel" key. If it is set, then our system returns that value.

Thus, the whole dataset representation is

$$\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$$

where N is the total number of images. In order to train the SNS and Upload Scenario classifiers, we augment this representation with the corresponding labels. Thus, the final representation for a generic image I_i is

$$\boldsymbol{l}_i = \{\boldsymbol{v}_i, sns_i, uc_i, sm_i\}$$

where sns_i is the SNS, uc_i is the client application and sm_i is the image selection method.

Our classifier performs a two-steps analysis. First, we implement an Anomaly Detector to exclude the images that have not been processed by SNSs, then we run in parallel a K-NN Classifier and a Decision Tree [20] to asses respectively the SNS of origin and the uploading scenario (OS + application).

Given the representations $\boldsymbol{v}_{I_1}$ of an image I_1 and $\boldsymbol{v}_{I_2}$ of an image I_2, we define the cosine distance between $\boldsymbol{v}_{I_1}$ and $\boldsymbol{v}_{I_2}$

$$d(\boldsymbol{v_1}, \boldsymbol{v_2}) = \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{|\boldsymbol{v_1}||\boldsymbol{v_2}|} \qquad (2)$$

as a measure of similarity between I_1 and I_2. Therefore, it is possible to build a distance matrix $\boldsymbol{D}$ of size $N \times N$ where the element d_{ij} is equal to the distance between the images I_i and I_j. We will refer to the $r-$th row of this matrix as $\boldsymbol{D}_r$ and to the $c-$th column as $\boldsymbol{D}^c$. It is important to note that $\forall\ I_i, I_j,\ 0 \leq d(\boldsymbol{v_i}, \boldsymbol{v_j}) \leq 1$, and specifically, the more is the similarity, the more the distance will be closer to 1. Exploiting this property, we define the Anomaly Detector as

$$a(\boldsymbol{v}_i, \boldsymbol{D}) = \begin{cases} (\boldsymbol{v}_i, i) & if\ \sum_{j=1}^{K} d_{ij} < T \\ \text{not processed} & otherwise \end{cases} \qquad (3)$$

where $T \in [0, K]$ is defined as the Anomaly Threshold. In other words, since the more two images are similar, the more their distance will be closer to 1, we make sure that at least $\lfloor K \rfloor$ samples in our dataset are similar to the query image representation. Then, when $a(\boldsymbol{v}_i, \boldsymbol{D}) = 0$, the representation is far apart the samples, and we can state that probably the image has not been processed.

The output of a is then used as input by 1-NN (4) and Decision Tree algorithms [20].

$$knn(\boldsymbol{v}_i, i) = sns_j \mid d_{ij} = \min \boldsymbol{D}^i \qquad (4)$$

$$dt(\boldsymbol{v}_i, i) = (uc_j, sm_j) \qquad (5)$$

where uc_j and sm_j are the leaves obtained following the path with $\boldsymbol{v}_i$ as input. Hence, the classification scheme, shown in Figure 1, can be formalized as follows

$$C(\boldsymbol{v}_i, \boldsymbol{D}) = knn(a(\boldsymbol{v}_i, \boldsymbol{D})) \oplus dt(a(\boldsymbol{v}_i, \boldsymbol{D})) \qquad (6)$$

1-NN algorithm looks for the closest sample in the dataset, and assigns the same SNS to the query image. A Decision tree (Eq. 5) builds classification in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes. The algorithm used for building the decision tree is the ID3 [20] which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy to construct a decision tree by evaluating $\boldsymbol{v} \in \boldsymbol{V}$.

Finally, the output of the 1-NN Classifier sns_j is processed through a SNS Consistency Test. Let be $S = \{sns_1, \ldots, sns_n\}$ the set of SNSs that operates a recompression at the condition $max(w, h) > C_{sns_i}$ where C_{sns_i} is the conditional threshold for the $i-$th SNS and w and h as listed in Table III.

Given that $sns_j \in S$, if $max(w, h) < C_{sns_j}$ it is an anomaly. The test is then repeated for the next most probable prediction from the SNS Classifier until the corresponding condition is satisfied or the loop stalls on the same SNS prediction. In this last case, the result of the classification is "not sure"; otherwise, a SNS prediction is reached and outputted (sns_j) with the predicted upload client application (uc_j) and image selection method (sm_o).
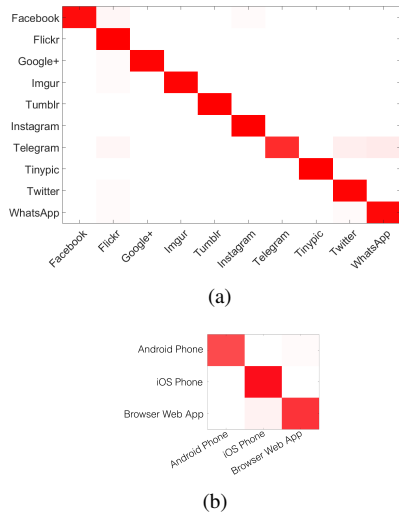
Figure 1 shows a schematic representation of the proposed classification engine.

B. Classification Results

In this Section, validation results for the proposed approach are reported to demonstrate its goodness. The anomaly detector was validated by taking from our dataset 240 random images that suffered alterations, and 240 images that did not pass through any alteration. The anomaly detector achieved the best error rate, equal to 3.37%, with K = 3 and T = 2.90.

The entire approach for image ballistics described in the previous Section was then tested through a 5-fold cross

Fig. 2. Heatmaps for Confusion Matrices obtained from 5-cross validation on our dataset. The reported values, coded in heatmap colors, are the average value between the 5 runs of cross validation. (a) Confusion Matrix for Social platform Classification, (b) Confusion Matrix for upload method classification.

(a)

(b)

validation test. In Figure 2, confusion matrices reporting the average value through the 5 runs are shown.

The accuracy obtained for the SNS classification task was 96% with best K equal to 3 while the accuracy value for the upload client classification task was 97.69% with an accuracy of 91% for the prediction of image selection method, given iOS or Android native app as prior.

A different approach, with a cascade of classifiers, was also tested with each classifier being alternately the predictor for the other one, but the overall results were slightly worse. The classification scheme reported in Figure 1 was the best approach we obtained throughout our tests.

C. Discussion

In our experiments, we observed that, as happens for different camera devices of the same model [15], different images, from the same platform, have slightly differences in DQT coefficients. Hence, the most discriminative features were chosen among those listed in Section IV. At conference time more details about this aspect will be presented.

Another consideration is needed about the SNSs fingerprints described in this work and regarding the fact that all the alterations observed can change according to software development and releases. For these reasons, in order to solve the task of predicting the SNS that processed an image and the client carrying out the upload, the proposed automatic and probabilistic approach is justified for being able to scale and readapt through time, just by updating the reference dataset.

V. Conclusions and future works

In this work, we presented a dataset for image ballistic and proposed a classification engine to discover if an image has been processed by a Social Network Service and, if the answer is positive, by which SNS among the 10 considered platforms.

The proposed approach performed the task of Image Ballistics with good accuracy by predicting the SNS that process an image and the corresponding upload method, with an accuracy respectively of 96% and 97.69%.

We think that this work can open new perspectives on the field of Image Forensics: the approach can be upgraded by considering other formats (e.g., PNG) and new features related to image contents.

References

[1] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, p. 22, 2013.

[2] M. C. Stamm, M. Wu, and K. J. R. Liu, "Information forensics: An overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[3] A. R. Bruna, G. Messina, and S. Battiato, *Crop Detection through Blocking Artefacts Analysis*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, vol. 6978, pp. 650–659.

[4] W. Luo, Z. Qu, J. Huang, and G. Qiu, "A novel method for detecting cropped and recompressed image block," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, April 2007, pp. II.217–220.

[5] S. Battiato, G. M. Farinella, E. Messina, and G. Puglisi, "Robust image alignment for tampering detection," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1105–1117, Aug 2012.

[6] E. Kee, M. K. Johnson, and H. Farid, "Digital image authentication from JPEG headers," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1066–1075, 2011.

[7] E. Kee and H. Farid, "Digital image authentication from thumbnails," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2010, pp. 75 410E–75 410E.

[8] T. Gloe, "Forensic analysis of ordered data structures on the example of JPEG files," in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2012, pp. 139–144.

[9] Y. Chen and V. L. Thing, "A study on the photo response non-uniformity noise pattern based image forensics in real-world applications," in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2012, p. 1.

[10] A. E. Dirik and A. Karaküçük, "Forensic use of photo response non-uniformity of imaging sensors and a counter method," *Optics express*, vol. 22, no. 1, pp. 470–482, 2014.

[11] 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.

[12] F. Galvan, G. Puglisi, A. R. Bruna, and S. Battiato, "First quantization matrix estimation from double compressed JPEG images," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 8, pp. 1299–1310, 2014.

[13] S. Battiato and G. Messina, "Digital forgery estimation into DCT domain: A critical analysis," in *Proceedings of the First ACM Workshop on Multimedia in Forensics*, ser. MiFor '09. New York, NY, USA: ACM, 2009, pp. 37–42.

[14] H. Farid, "Digital image ballistics from JPEG quantization: A followup study," *Department of Computer Science, Dartmouth College, Tech. Rep. TR2008-638*, 2008.

[15] J. D. Kornblum, "Using JPEG quantization tables to identify imagery processed by software," *Digital Investigation*, vol. 5, Supplement, pp. S21 – S25, 2008, the Proceedings of the Eighth Annual {DFRWS} Conference.

[16] M. Moltisanti, A. Paratore, S. Battiato, and L. Saravo, *Image Manipulation on Facebook for Forensics Evidence*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9280, pp. 506–517.

[17] A. Castiglione, G. Cattaneo, and A. D. Santis, "A forensic analysis of images on online social networks," *Intelligent Networking and Collaborative Systems, International Conference on*, vol. 0, pp. 679–684, 2011.

[18] "DJPEG – LibJPEG open-source project on GITHUB," https://github.com/LuaDist/libjpeg.

[19] J. Miano, *Compressed image file formats: jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[20] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.