Two-stream Convolutional Neural Network for Image Source Social Network Identification

***Abstract*—The identification of the source social network from an image is a relatively new research area in the image forensic domain. The classification of the source social network can be a crucial element for the growing number of cases of social-media related crimes, such as cyberbullying. This paper takes into consideration the state-of-the-art approaches addressing this problem and proposes a new methodology to improve the results obtained to date. Our identification technique is based on the idea that social networks perform some processing on the uploaded images, such as resizing or re-compression, and leave some artifacts on them. We propose to use DCT features and image noise residual analysis to detect such artifacts. A two-stream convolutional neural network, which combines the inputs from these two artifact domains, is trained to classify the source social network of images coming from three different datasets. This paper explores the two domains, proposes strategies for managing unbalanced datasets, provides details about the proposed two-stream CNN, and presents the results achieved by our method compared with the current state-of-the-art approaches.**

***Keywords*-Digital image forensics; Social network identification; Convolutional neural networks; Two-stream neural networks; Noiseprint.**

I. Introduction

We live in a society where social networks (SNs) have become an integral part of our daily life. Billions of images are exchanged every day on SNs for a variety of purposes, which sometimes include malicious activities. Cyberbullying, violence instigation, and psychological harassment are sometimes linked to media files exchanged via SNs like, for example, WhatsApp, Facebook, or Instagram. When a device is confiscated from a suspect, an image can become a criminal evidence and therefore detecting the origin of that image can be really useful to help with the investigation.

The goal of this research is to propose and develop a methodology that enables the identification of the source SN from an image. Each SN possesses its own fingerprint that depends on its processing algorithm. Therefore, the main idea is to analyze the artifacts that are left on an image to identify the source SN. The identification is performed blindly, using only the data that is extracted from the image, without trusting the image header data, which could be easily removed or edited without modifying the image content. The presented methodology uses two types of feature domains, where artifacts are more easily detectable: the discrete cosine transform (DCT) domain and the photo response non-uniformity (PRNU) domain. The identification is performed by a convolution neural network (CNN), trained for this task, in two steps: first, the artifacts are extracted and then used for the classification. The method is tested on three publicly available datasets. The proposed CNN model is called *two-stream* network, because it takes as input two sets of features (DCT and PRNU). The two sets of features are extracted independently of each other and then concatenated and used for the classification of the source SN.

The contributions of the paper are the following:

- We propose a two-stream CNN model that achieves 98% correct classification of three source SNs (Facebook, Flickr and Twitter) in average over three test datasets;
- We propose a new method to encode the DCT coefficients in a compact feature vector, which provides better performance compared to the state of the art;
- We propose the use of *Noiseprint*, a CNN based camera model feature extractor, to analyze the SN noise pattern, which also provides better performance compared to the state of the art;
- We propose solutions for issues related to unbalanced datasets, from assuring balanced patch-level CNN training, to unbiased image-level classification.

II. Related work

Many image forensic tasks have achieved excellent performance thanks to the use of machine or deep learning [13], [1], [12]. In particular, CNNs have shown a great potential and are becoming the standard for solving image-related problems [14]. When considering the identification of the origin from an image, the literature usually refers to *source digital camera identification*, rather than source SN identification. The source camera identification problem has been widely addressed by the image-forensic researchers [11], [4], [9]. Machine learning and deep learning have been adopted by the majority of the best-performing state-of-the-art methods [13], [8].

The identification of the source SN from an image is a relatively new and unexplored research field. Some previous works approach the problem by considering different features and are based on several assumptions. For example, in [10] the use of the filename and metadata, as well as the resizing and re-compression factors, is proposed to identify the source SN using a k-Nearest Neighbors (k-NN) classifier. In this case, the authors assume that metadata and filenames

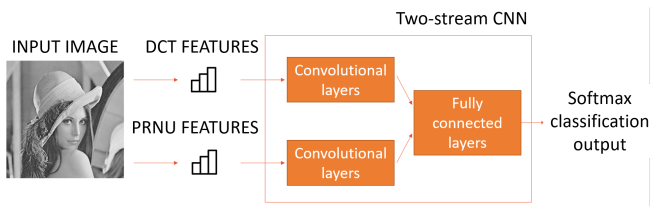


Figure 1: Scheme of the proposed *two-streams CNN*.

can be trusted, proposing a so-called *non-blind approach*. Instead, in this article, we are interested in providing a *blind* approach, which trusts only the image content. The number of state-of-the-art blind approaches is very limited: we identified two works based on the analysis of DCT-block features. The first one [7] proposes the use of DCT-block features and a bagged decision tree classifier. The second one [2] uses also DCT-block features but with a CNN classifier. Another blind approach is suggested by the paper [6]: it uses the residual noise of the image and a CNN classifier to identify the source SN. These methods are based on the assumption that once an image is uploaded to a SN, it goes through some kind of processing that leaves some artifacts on it. These artifacts create a distinctive pattern usually called *fingerprint*. For what concerns the DCT-block domain, JPEG quantization causes a perturbation in the distribution of the coefficients, while for the residual noise domain, the photo response non-uniformity (PRNU) pattern of the image is modified by the processing proper to a SN.

For the PRNU extraction, a recent work provides a pre-trained CNN for image forensic task, named *Noiseprint* [8]. Noiseprint generates a camera-model fingerprint pattern that the authors of [8] proved to be very useful for image forgery detection. In this paper, we investigate the applicability of Noiseprint for source SN identification.

Regarding two-stream neural networks, the literature works dealing with source SN identification have not exploited this architecture yet. However, in other areas of digital image forensics, the two-stream neural network architecture has proven to be successful. For example, a method for the detection of double JPEG compression is presented in [3], which is based on the analysis of two different features coming from the spatial domain and from the frequency domain, including DCT-block features. Another two-stream neural network method is proposed in [16] for the detection of image manipulations through the analysis of the original image and its local noise features.

III. Proposed method

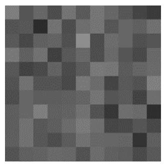
The proposed method is based on the assumption that the images uploaded to a SN go through some processing that leaves traces **with a specific pattern** on them. The most commonly used compression methods by SNs are JPEG compression and resizing. JPEG compression is a lossy compression algorithm that compress the image by quantizing the coefficients of the DCT block before storing them using the Huffman-coding. JPEG quantization disrupts the distribution of the DCT-block coefficients, creating artifacts that can be used to determine the source SN of the image. Resizing is achieved by reducing image size into a smaller image with the interpolation of the pixels. Even if more subtle, interpolation can create artifacts too. Sometimes, additional manipulations can be applied by the processing algorithm of the SN, notably to compensate the other transformations that may cause blurring. All these types of processing generate artifacts in the resulting image, which can be more or less detectable depending on the domain that we are analyzing. Such artifacts are usually extracted by using preprocessing modules before being fed to a neural network. A review of existing techniques for image-forensic analysis based on deep learning can be found here [5].

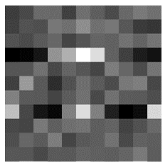
The method that we propose here combines two domains where artifacts coming from the SN image processing can be detected: DCT-block and PRNU analysis. Our assumption is that the combination of the two domains may enhance the artifacts diversity and, therefore, the possibility to correctly identify the source SN. A CNN architecture is used to perform feature extraction for both domains. We call this approach *two-stream network* because starting from the same image, two domains are separately analyzed and fed into the CNN via two inputs. Both inputs go through a separate and different series of convolutional layers before being concatenated and classified (see Figure 1). More details on the network architecture are given in section III-C.

A. DCT block domain

The DCT blocks can be generated by taking the Y channel of the $YCbCr$ color space of the input image and splitting it into 8×8 blocks of pixels and then computing the discrete cosine transform for each block. The original image can be reconstructed by applying the inverse process, using the inverse DCT (IDCT). JPEG compression uses this technique to reduce the image size on disk by storing only the DCT-block coefficients instead of the plain RGB data. Before storing the DCT coefficients, JPEG compression applies quantization by dividing each coefficient by a predetermined value contained in the quantization table. The results of the division are then rounded to the nearest integer and compressed with Huffman-coding.

In our case, the crucial aspect of JPEG compression is the analysis of the distribution of the DCT-block coefficients that permits to detect the quantization artifacts. Considering that each SN applies its own series of transformations that changes the distribution of the DCT coefficients (see Fig. 2) we can train our CNN on this type of features to identify the source SN. However, feeding the network directly with the plain DCT coefficients is not ideal considering that


(a) Uncompressed


(b) Facebook

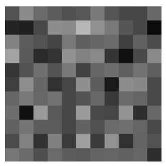

(c) Flickr

Figure 2: Comparison of the DCT-block features: (a) original; (b) after Facebook processing; and (c) after Flickr processing.

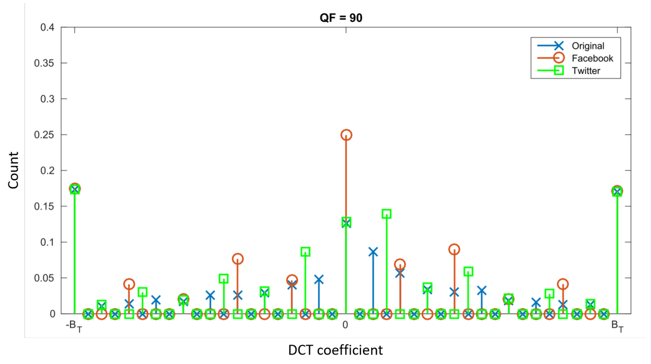


Figure 3: Example of DCT-block coefficient distribution histogram for the same image processed by different social networks. Image taken from [7].

quantization leaves traces that are difficult to detect by a CNN. Therefore, it is recommended to encode the DCT-block coefficients in a way that makes the quantization artifacts more detectable for the CNN.

The work presented in [2] proposes the use of a histogram-based approach that encodes the DCT coefficients by counting the occurrences of a given value in the DCT-blocks (see Figure 3). The image is firstly cropped in non-overlapping patches of size $N \times N$ to avoid repercussions in DCT, which is affected by the content and size of the considered image [15]. The DCT is then computed for each patch and, for each 8×8 DCT block, the first 9 spatial frequencies in zig-zag scan order are selected to compute the histogram. For each spatial frequency (i, j), the histogram $h(i, j)$ representing the occurrences of a value from the quantized DCT coefficients is built. The histogram ranges from -50 to $+50$ (101 bins), as most of the coefficients fall within these values, and the output of the encoding is therefore $101 \times 9 = 909$ values. Coefficients above 50 or below -50 are counted in the last and first bin, respectively [7]. Thanks to this encoding, the pattern due to the quantization is extracted and made detectable for a CNN.

Even if the encoding proposed in the previous work [2] achieves good results, we investigated other encoding methods with the aim of further improving performances. One limitation of the previously proposed encoding is the loss of information concerning coefficients above $+50$ or below -50. An easy solution would be to increase the range of the histogram, but this would not be ideal with a distribution of small number of values for numerous bins.

We therefore developed a different encoding scheme that makes quantization artifacts even more detectable and uses all the values of the DCT block. This encoding is based on the normalization of the DCT coefficients in a limited range of values prior to the computation of the histograms. Given a coefficient value x and a quantization factor q, the normalization is defined as follows:

$$x_n = \frac{x}{q} - round(\frac{x}{q}) \tag{1}$$

The normalized values go from 0, which indicates that it is likely that the value x has been quantized with the factor q, to ± 0.5, which indicates that it is unlikely instead.

We thus define the DCT-block features by computing the histograms of the DCT-block coefficients, for each image patch p and spatial frequency f, as follows:

$$hist_{p,f}(\frac{x_{p,f}}{q} - round(\frac{x_{p,f}}{q})) \tag{2}$$

The set of histograms extracted from an image patch describes the quantization artifacts and can be used by the CNN to classify the source SN. The downside of this encoding procedure comes from the quantization factor, which is not known beforehand. Therefore, we have to try all the possible quantization factor values q up to 20 to detect the artifacts. In fact, as we consider only the coefficients in the first 9 spatial frequencies of the DCT block, the values of q are small and, therefore, we can set the maximum value at 20. For each value of q, where $q = 1, 2, 3, ..., 20$, we compute the histogram of DCT-block coefficients ranging from -0.5 to 0.5 (see Eq. 2), with the number of bins equal to 11 (one bin for each decimal interval from $[-0.5$ to $0.5]$).

For each image patch, the computed histograms are concatenated to form the feature vector. The final length of this encoding vector is 20 possible quantization factors $\times 11$ bins $\times 9$ spatial frequencies $= 1980$ values. As demonstrated by the experimental evaluation reported below, the proposed encoding procedure achieves very good performances.

B. PRNU domain

The Photo Response Non Uniformity (PRNU) is a distinctive pattern due to imperfections in the silicon wafer during the sensor manufacturing, different even among cameras of the same model. These imperfections imply that the pixels have different sensitivities to light. PRNU is usually used for source digital camera identification [11], [4], but a previous work has shown that it can be used for an accurate classification of the source SN of an image as well [6]. In fact, the processing algorithm of a SN affects and slightly modifies the PRNU pattern (see Figure 4), but not enough

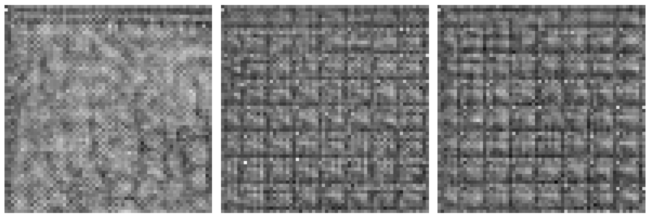

(a) Uncompressed (b) Facebook (c) Flickr

Figure 4: Comparison of a 64×64 pixels patch of Noiseprint, extracted from the same image, (a) original, (b) after the upload to Facebook, (c) and after the upload to Flickr.

to completely suppress the PRNU. Therefore, by analyzing the PRNU after the image upload to a SN, it is possible to detect artifacts that can help in identifying the source SN. Although the analysis of PRNU is also based on the study of artifacts due to (but not limited to) re-compression as for the analysis of DCT blocks, the features extracted in this domain are different and independent of those coming from the DCT blocks and, therefore, the combination of the two could enhance the classification of the source SN.

The PRNU is constant in images taken with the same camera sensor. Hence, the standard approach of PRNU extraction is to take multiple images from the same camera and extract the noise using a denoising filter. By calculating the average of the extracted noise patterns, Gaussian distributed noise with a zero mean tends to disappear, while only the sensor pattern noise remains. This is the same approach used by the work presented in [6], which performs image source SN identification based on residual noise extraction. There is a clear limitation to this method, which is the requirement of having numerous images from the same sensor in order to get an accurate extraction of the PRNU. In order to address this issue, the work [6] uses the residual noise from a single image instead of the camera PRNU computed over several images. The problem in using the residual noise is that it is strongly affected by the image content and contains a considerable amount of noise that does not come from the SN uploading process, making the identification of the source SN for an image more difficult.

We address this problem differently by using a recently developed PNRU extractor, namely *Noiseprint* [8]. Noiseprint is a pre-trained CNN that, given an image, generates a camera model fingerprint, called *noiseprint*, where the scene content is largely suppressed and model-related artifacts are enhanced. Even if the Noiseprint CNN does not extract the PRNU directly, the generated pattern can be considered as PRNU-related, as it correlates with the camera sensor artifacts present in the image. Noiseprint has shown a great potential in image forgery detection (another task that can be solved by the PRNU analysis), but the authors suggest that the Noiseprint pattern can be useful for other image forensic tasks as well. We therefore decided to test

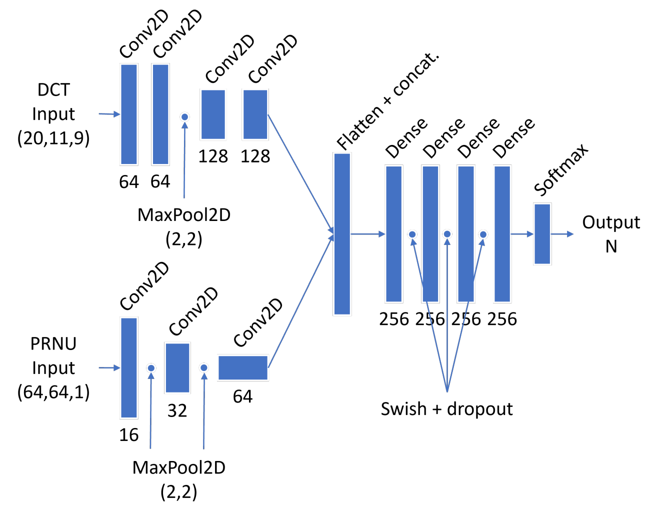


Figure 5: Proposed two-stream CNN architecture.

if the Noiseprint can be used to determine the source SN of an image. We present the results of this evaluation along with the comparison with the previous noise-residual-based work [6] in section IV-C.

C. CNN Design

The structure of the *two-stream* CNN was developed by firstly designing and evaluating each stream separately. The proposed architecture for the two-stream CNN is illustrated in Figure 5. The design of each stream is different because the input data have different nature as well as different size. Therefore, the size and number of layers are not the same. The datasets were split into training (80%), validation (10%), and test (10%) sets, randomly, with a fixed seed so that the splitting could be reproducible for different training and test sessions. For the DCT stream, both DCT-block encoding methods (the one from the state of the art and ours) were tested using two different network structures, due to the different input shape. With this setup, we conducted the hyper-parameters tuning phase by repeating the same training session, with the same fixed seed, and by introducing both intuitive and random changes to the parameters in order to find the best model structure. At each change, the performance of the model is evaluated using the validation set and the model with the highest performance is chosen.

The CNN has two inputs: encoded DCT block and image patch of size 64×64 (for Noiseprint). Each input goes through a series of convolutional layers with 3×3 kernels and *relu* activation, and max pooling layers. The number of filters in the convolutional layer is increased at each max pooling layer. Batch normalization is applied before each convolutional layer, except for the first one. The two streams are then flattened and concatenated before the classification layers. The classification layers are fully connected layers (called *Dense* in the Keras implementation) and use the *swish* activation function: $swish(x) := x \times sigmoid(\beta x) = \frac{x}{1+e^{-\beta x}}$, where β is a constant.

Dropout is applied before each Dense layer except for the first and the last ones. The output of the network is a Dense layer with *soft max* activation and size N given by the number of classes (3 in our experiments). The loss function is *categorical cross-entropy* and the optimizer is Nesterov-accelerated Adaptive Moment Estimation (Nadam).

IV. Experimental results

A. Database

The proposed method adopts a supervised learning technique, therefore a labelled dataset is required to train the two-stream CNN model. Being the number of works addressing the problem of source SN identification limited, the number of publicly available labeled datasets is limited too. For the sake of comparison with the state of the art, we decided to use the same three datasets as in previous works:

UCID Social: this dataset is generated by taking the 1338 images of the *UCID* dataset, compressing each of them with 10 different JPEG quality levels (from 50 to 95 with steps of 5), then uploading each image to three SNs (Facebook, Flickr, and Twitter). The images are then downloaded and stored in different labelled folders. The number of images in the three classes are therefore $1338 \times 10 \times 3 = 40140$. The dataset also contains multi-class images: images that have been first uploaded to a SN (e.g. Facebook), then downloaded and re-uploaded to another SN (e.g. Twitter) and finally stored in a multi-class labeled folder. These images can be used for multi-class origin identification, which is out of the scope of this research.

IPLab: this dataset is generated by taking 240 images and by uploading them to 8 image sharing services including SNs and messaging services. The images are then downloaded and stored in different labelled folders. In this paper we use only three SNs for our research, namely Facebook, Flickr, and Twitter.

Social Public: this dataset is generated by taking three target SNs (Facebook, Flickr, Twitter) and by downloading 1000 images from each of them. The images are stored in labelled folders to create the dataset.

UCID Social and *IPLab* are *controlled environment datasets*, meaning that the data was generated by taking a set of images and uploading them to each SN, then downloading them to create a dataset where each class contains the same original images that have undergone different processing.

Social Public is an *uncontrolled environment dataset*, meaning that the data was generated by downloading a set of random images from the SNs. Thus, the set of images for each class (i.e. SN) is different.

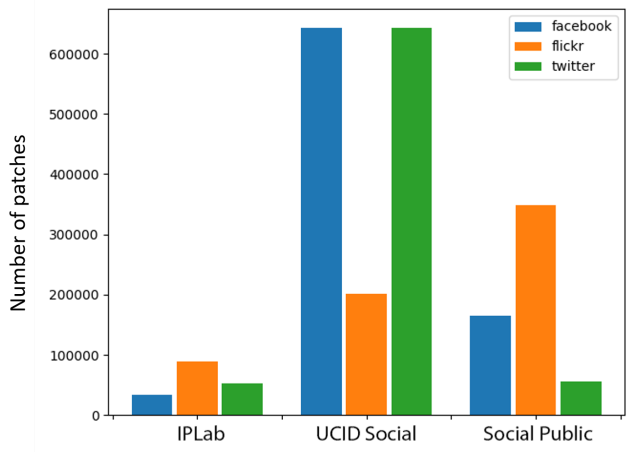


Figure 6: Number of patches per class, for each of the three datasets.

B. Dataset balance

In order to train a neural network (NN) it is advisable to have a balanced training set, where the number of samples per class is equal for each class, otherwise the model could converge by privileging the most represented class. In each of the three selected datasets the data is perfectly balanced at the image level: 13380 images per class in the *UCID Social* dataset, 1000 images per class in the *Social public* dataset, and 240 per class in the *IPLab* one. However, because our training samples are 64×64 pixels patches, we have to make sure that the datasets are balanced at the patch-level too. Since the images are processed by different SNs, and the processing includes resizing in some cases, the final number of extracted patches is going to be different for each class (see Figure 6), which is not ideal for training our two-stream CNN.

In order to solve this issue, for a given training dataset we consider each class of the dataset (Facebook, Flickr, and Twitter) as a standalone data generator and, thanks to the *Tensorflow* Dataset APIs, we create an infinite-loop generator for each class, with reshuffling. We then interleave the three classes generators, by taking a sample from each class in turn, to create an infinite Dataset generator of perfectly balanced batches of training data. The same class-balance issue can be a problem also for the validation and test set, as the metrics used (e.g. recall and accuracy) can output misleading results if the validation classes are unbalanced. We therefore made validation and test results balanced by defining class weights.

A class weight is a multiplicative factor applied to the calculation of a metric or loss function associated with each sample of a given class. For the validation and test sets, the total weight of each class is first calculated by summing up the sample weights. If the total class weights are different, a correcting class weight factor f_k is multiplied to each sample weight for each class. More details about class weights will be given in section IV-D2.

C. Single-stream CNN evaluation

1) PRNU-based method: Each dataset (*UCID Social*, *Social Public*, and *IPLab*) is split at the image level (training 80%, validation 10%, test 10%) and evaluated separately.

Before evaluating the *two-stream* CNN, the single streams are evaluated separately to validate the developed methods. First, we verify our assumption about the use of Noiseprint: we want to make sure that the Noiseprint-based stream is appropriate for SN identification and that the performances are at least as good as for the method proposed by the previous work based on residual noise [6]. We therefore evaluated the *single-stream* Noiseprint-based CNN alone. In order to provide a direct comparison with the results obtained in [6], we reproduced the same experiment on the UCID Social dataset on the same three SNs (Facebook, Flickr, and Twitter) and computed the same evaluation metric, that is classification *precision*:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

where TP = True Positives and FP = False Positives.

The evaluation is performed on the test set of the *UCID Social* dataset at patch-level, by considering the precision metric on the classified patches. The stratified repeated random sub-sampling validation (SRRSV) is applied with 5 iterations to validate the performance results. Our Noiseprint-based solution achieves an average precision of 90%, while the previous PRNU-based work [6] achieves only about 80% (see Table I) with the same protocol, suggesting that Noiseprint works very well as feature extractor for source SN identification.

UCID Social	*Facebook*	*Flickr*	*Twitter*
Residual noise [6]	72.80%	93.15%	72.49%
Single-stream Noiseprint (Our)	87.6%	95.8%	91.5%
Two-stream (Our)	**99.32%**	**99.67%**	**97.83%**

Table I: Comparison of **patch-level** classification *precision* for each class of the *UCID Social* dataset. The proposed Noiseprint-based single-stream CNN and the proposed two-stream CNN are compared with the method based on residual noise presented in [6].

2) DCT-based method: Then, we evaluate the DCT-block-based *single-stream* CNN alone. For this evaluation, the proposed method is compared with the previous DCT-based work [2]. Both CNN are very similar with a difference only in the processing of the DCT-based input. Therefore, we decided to determine which of the two DCT-block encoding techniques achieves the best performances, and to define the CNN architecture to be used in the final *two-stream* CNN accordingly. By comparing the performance of the two different encoding methods on three test sets, separately, we acknowledge only a slight performance improvement (recall difference $< 0.5\%$, where $recall = \frac{TP}{TP+FN}$) for our proposed encoding compared to the one proposed by [2]. Moreover, the model convergence speed is faster when using our encoding, with about 20% fewer epochs required in the training phase. Due to these two advantages we adopt our encoding for the *two-stream* CNN.

D. Two-stream CNN evaluation

For the evaluation of the *two-stream* network, we follow a similar approach to single-stream evaluation, by evaluating the performances on the three datasets separately and reporting the classification performances for the three classes: Facebook, Flickr, and Twitter. Also, for sake of comparison with the state of the art, we adopt the same protocol as in the article [6], and perform the evaluation in two steps: first, we evaluate classification performances at patch level, then at image level.

1) Patch-level evaluation: We firstly present the results for the patch-level evaluation. For a comparison with the state-of-the-art method based on residual noise [6], we report in Table I the classification precision. The results show that the two-stream CNN achieves better performance values than [6] and has further improved the classification precision compared to the single-stream Noiseprint method.

To have a direct comparison with the results of the state-of-the-art DCT-based method [2], we provide the confusion matrices for the three datasets as the authors of [2] did in their paper. The results are validated with a 5 repetition SRRSV and indicate that the patch-level performance of the proposed method is higher compared to the one in the DCT-based paper [2], see Table II.

UCID Social	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**97.60%**	96.15%	0.20%	0.19%	2.20%	3.66%
Flickr	0.00%	0.03%	**100%**	99.79%	0.00%	0.18%
Twitter	0.67%	0.59%	0.13%	0.11%	**99.30%**	**99.30%**
Social Public	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**97.30%**	94.00%	2.52%	6.00%	0.18%	0.00%
Flickr	7.11%	1.76%	89.61%	**92.13%**	3.28%	6.11%
Twitter	0.10%	0.00%	3.49%	13.47%	**96.41%**	86.53%
IPLab	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**97.30%**	94.00%	2.52%	6.00%	0.18%	0.00%
Flickr	7.11%	1.76%	89.61%	**92.13%**	3.28%	6.11%
Twitter	0.10%	0.00%	3.49%	13.47%	**96.41%**	86.53%

Table II: Two-stream CNN evaluation: Confusion matrix for the **patch-level** classification of the *UCID Social*, *Social Public* and *IPLab* datasets, among Facebook, Flickr, and Twitter and comparison with the SOTA DCT-based work [2].

2) Image-level evaluation: If we consider the original issue of finding the source SN of a given image, we need to perform the evaluation at the *image level*. In order to obtain

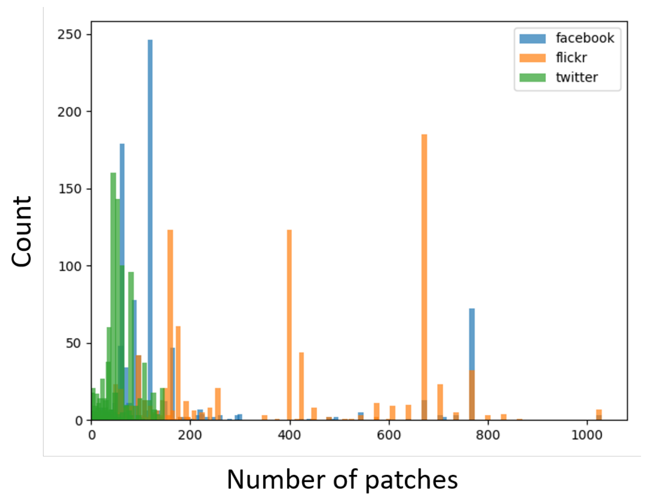


Figure 7: Comparison between the distribution of the number of patches per image in the Facebook, Flickr, and Twitter classes of the *Social Public* dataset. Flickr has many large (> 600 patches) images, while Twitter has only very small images.

the classification for the entire image, we simply extract all the 64×64 pixels non-overlapping patches from the image, and we classify each of them using the trained CNN. We then consider the majority-voted class as the predicted class for the entire image. However, the distribution of the number of patches per image (i.e. image size) is different for each class (Facebook, Flickr, Twitter) in the three datasets (see Figure 6). The model could then be biased by this unbalanced partition, as some classes may be underrepresented compared to the others, for example Twitter in the Social Public dataset has an overall number of patches much lower than the other SNs, as illustrated in Figure 7. In order to fix this issue, we modify the weight of each training sample to be inversely proportional to the number of patches from an image: the weight of a sample coming from an image with N patches is therefore $\frac{1}{N}$. However, because we want to keep the overall weight balanced for each class, we also multiply the weight by the average number of patches per image. The final weight for a sample w_{Nk}, from an image with N patches of class k, is therefore defined by the number of patches $|P_k|$ and total number of images I_k of class k (see Eq. 4).

$$w_{Nk} = \frac{|P_k|}{|I_k|} \cdot \frac{1}{N} \tag{4}$$

In order to obtain an equal number of samples for each class in the dataset with the help of an infinite data generator (protocol introduced in section IV-B), we need to modify the weight definition given above and consider the *frequency* of appearance of the generated patches. For example, the frequency F_k of appearance of a patch from a given image in the training set is defined by its number N of patches or the associated weight $|P_k|$ (see Figure 5). In order to obtain a new class weight factor f_k (also mentioned in section IV-B) that is balanced according to the number of patches per image and per class, we combine this frequency F_k with the proposed weight w_{Nk} (see Eq. 6).

$$F_k = N \cdot \frac{1}{|P_k|} = \frac{N}{|P_k|} \tag{5}$$

$$f_k = w_{Nk} * F_k = \frac{|P_k|}{|I_k|} \cdot \frac{1}{N} \cdot \frac{N}{|P_k|} = \frac{1}{|I_k|} \tag{6}$$

Therefore, the overall weight for all the patches from a given image during the training phase depends only on the number of images per class $|I_k|$, which is perfectly balanced in our datasets. This will provide the same exact weight for each image in the training phase, regardless of its class and number of patches.

We tested the proposed weighting technique by training the CNN *without* and *with* the sample weights. The image-level performance increased from an average 89% to 95% recall when tested on the *Social Public* dataset and from 95% to 98% recall on the *IPLab*.

The image-level comparison between the performance obtained by the proposed two-stream CNN, trained with the sample weight, and the state-of-the-art methods [6] and [2], is shown in table III and IV, respectively. The proposed method outperforms the state of the art for all tests except for the classification of Twitter images on Social Public, where the DCT-based method [2] achieves 100% correct classification while our achieves 98.67% (see Table IV).

UCID Social	**Facebook**	**Flickr**	**Twitter**
Two-stream (Our)	**100%**	**99.80%**	**98.62%**
Residual noise [6]	87.35%	97.42%	87.73%

Table III: Two-stream evaluation: Comparison of **image-level** classification *precision* for each class of the *UCID Social* dataset. The proposed two-stream CNN is compared with the method based on residual noise in [6].

V. Conclusion

This paper proposes an improvement to the state-of-the-art approaches that address the problem of image source social network (SN) identification with the use of a two-stream convolutional neural network (CNN). The key elements of the proposed methodology are: the development of a two-stream CNN that takes as input both DCT-block and PRNU features for the classification of the images; the use of the Noiseprint generated pattern for the PRNU-feature extraction; a new DCT-block encoding method to make quantization artifacts more detectable; and the analysis and proposal of solutions for issues related to unbalanced datasets, from assuring balanced patch-level CNN training, to unbiased

UCID Social	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**98.20%**	97.37%	0.20%	0.00%	1.40%	2.63%
Flickr	0.00%	0.00%	**100%**	**100%**	0.00%	0.00%
Twitter	0.00%	0.00%	0.00%	0.00%	**100%**	**100%**

Social Public	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**91.21%**	88.24%	0.00%	0.00%	8.79%	11.76%
Flickr	0.00%	0.99%	**98.15%**	97.03%	1.85%	1.98%
Twitter	0.10%	0.00%	1.23%	0.00%	98.67%	**100%**

IPLab	Our	[2]	Our	[2]	Our	[2]
	Facebook		*Flickr*		*Twitter*	
Facebook	**97.86%**	96.01%	2.14%	3.99%	0.00%	0.00%
Flickr	2.45%	1.68%	**97.55%**	97.06%	3.28%	1.26%
Twitter	0.00%	0.00%	0.00%	1.26%	**100%**	98.74%

Table IV: Two-stream evaluation: Confusion matrix for the **image-level** classification of the *UCID Social*,*Social Public* and *IPLab* datasets among Facebook, Flickr, and Twitter and comparison with the SOTA DCT-based work [2].

image-level classification. We showed that Noiseprint can be used for SN identification and produces higher results compared to previous works based on residual noise. We took care of using balanced datasets and to maximize the image-level performances by using a patch-level majority-vote based approach. We evaluated our method using three different SN datasets and three SNs: Facebook, Flickr, and Twitter. The results produced by our two-stream CNN outperform those of the state of the art, both at patch- and image-level.

References

[1] R. Agarwal, D. Khudaniya, A. Gupta, and K. Grover. Image forgery detection and deep learning techniques: A review. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1096–1100, 2020.

[2] I. Amerini, T. Uricchio, and R. Caldelli. Tracing images back to their social network of origin: A cnn-based approach. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2017.

[3] Irene Amerini, Tiberio Uricchio, Lamberto Ballan, and Roberto Caldelli. Localization of jpeg double compression through multi-domain convolutional neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1865–1871, 2017.

[4] B. Balamurugan, S. Maghilnan, and M. R. Kumar. Source camera identification using spn with prnu estimation and enhancement. In *2017 International Conference on Intelligent Computing and Control (I2C2)*, pages 1–6, 2017.

[5] Alexandre Berthet and Jean-Luc Dugelay. A review of data preprocessing modules in digital image forensics methods using deep learning. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 281–284, 2020.

[6] R. Caldelli, I. Amerini, and C. T. Li. Prnu-based image classification of origin social network with cnn. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1357–1361, 2018.

[7] R. Caldelli, R. Becarelli, and I. Amerini. Image origin classification based on social network provenance. *IEEE Transactions on Information Forensics and Security*, 12(6):1299–1308, 2017.

[8] Davide Cozzolino and Luisa Verdoliva. Noiseprint: A cnn-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 05 2019.

[9] Chiara Galdi, Frank Hartung, and Jean-Luc Dugelay. Socrates: A database of realistic data for source camera recognition on smartphones. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*, pages 648–655. INSTICC, SciTePress, 2019.

[10] Oliver Giudice, Antonino Paratore, Marco Moltisanti, and Sebastiano Battiato. A classification engine for image ballistics of social data. In Sebastiano Battiato, Giovanni Gallo, Raimondo Schettini, and Filippo Stanco, editors, *Image Analysis and Processing - ICIAP 2017*, pages 625–636, Cham, 2017. Springer International Publishing.

[11] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006.

[12] D. Pan, L. Sun, R. Wang, X. Zhang, and R. O. Sinnott. Deepfake detection through deep learning. In *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pages 134–143, 2020.

[13] A. Roy, R. S. Chakraborty, U. Sameer, and R. Naskar. Camera source identification using discrete cosine transform residue features and ensemble classifier. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1848–1854, 2017.

[14] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[15] Qing Wang and Rong Zhang. Double jpeg compression forensics based on a convolutional neural network. *EURASIP Journal on Information Security*, 2016:1–12, 2016.

[16] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Learning rich features for image manipulation detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1053–1061, 2018.