IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 28, 2018, accepted November 22, 2018, date of publication November 30, 2018,
date of current version December 27, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2883588*

# Image Manipulation Detection and Localization Based on the Dual-Domain Convolutional Neural Networks

**ZENAN SHI[1,2], XUANJING SHEN[1,2], HUI KANG[1,2], AND YINGDA LV[3]**
[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[3]Center for Computer Fundamental Education, Jilin University, Changchun 130012, China

Corresponding author: Hui Kang (kanghui@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672259, Grant 61876070, and Grant 61602203, and in part by the Natural Science Foundation of Jilin Province under Grant 20170520064JH.

**ABSTRACT** In multimedia forensics, many efforts have been made to detect whether an image is pristine or manipulated with high enough accuracies based on specially designed features and classifiers in the past decade. However, the important task for localizing the tampering regions in a fake image still faces more challenges compared with the manipulation detection and relatively a few algorithms attempt to tackle it. With this in mind, a technique that utilizes the dual-domain-based convolutional neural networks (D-CNNs) taking different kinds of input into consideration is proposed in this paper. In the proposed framework, two sub-networks, named the spatial-domain CNN model (Sub-SCNN) and the frequency-domain-based CNN model (Sub-FCNN), are designed and trained, respectively. With the well-trained parameters, a transfer policy is applied to the training process of the D-CNN. While CNNs are capable of learning classification features directly from data, in their standard form they tend to learn features related to the image's content. To overcome this issue in image forensics tasks, a new image pre-processing layer is proposed to jointly suppress image's content and adaptively learn manipulation detection and localization features. After investigating the properties of datasets, two post-processing operations are finally proposed and compared to obtain the final results of the pixel-wise manipulation region localization. The D-CNNs is trained and validated using 75 percent of images in the CASIA v2.0 and tested using the remaining images in the CASIA v2.0, all images in Columbia Uncompressed and Carvalho datasets. The extensive experiments show that the proposed post-processing operations optimize the final tamper probability map, and our framework with the combination of Sub-SCNN and Sub-FCNN significantly outperforms the state-of-art techniques with the best F1 scores on the datasets.

**INDEX TERMS** Convolutional neural networks, multimedia forensics, post-processing operation, transfer policy.

## I. INTRODUCTION

Multimedia information, such as digital images, is frequently used in the numerous important settings, such as evidence in criminal investigations and military scenarios [1]. However, with the availability and pervasiveness of digital image editing tools, this information can easily be altered or tampered leaving no visual traces of any modification [2]. As a result, image manipulation detection has become a very important and challenging task due to the strong resemblance of a forged image to its original one in multimedia forensics. To determine the authenticity of digital images, most of the state-of-the-art image manipulation detection methods exploit the three principal classes of detectors which are based on the features descriptors or the frequency domain characteristics [3], [4], based on inconsistent shadows or noise levels [5]–[7] and finally based on double JPEG compression [8], [9] according to the individuation of image forgeries. Although research in image forensics has dramatically advanced, these approaches still suffer from important drawbacks that not all images have this individuation.

Recently, deep learning, such as convolutional neural networks, has shown its promising performance in a wide variety of computer vision tasks, including object detection [10], semantic segmentation [11] and so on. And there have a

few recent works which exploit the Stacked Autoencoder model (SAE) [12] and CNN [13], [14] to detect tampered images. In particular, Zhang *et al*. [12] used image statistics as input to SAE to obtain the tampered image block features and localize the tampered regions by a multilayer perceptron. The extensive use of deep learning in many areas has motivated and led the multimedia forensics community to analyze whether it is possible to force a CNN to learn manipulation detection features and capture pixel value dependencies induced by image tampering operation. In standard form, the CNNs provide a promising performance of automatically leaning features related to the image's content, whereas, the image's content should be suppressed in image forensics tasks. If the CNNs are used in their existing form, this will lead to a classifier that identifies the objects and scenes as opposed to learn image forensics classification features. To tackle this problem, a common method has emerged namely pre-processing layer based deep learning approach used in steganography. And the main idea behind [13], [14] is to develop a sort of pre-processing model, designed to suppress image content before training the CNN model. Inspired by the powerful steganalytic features called the spatial-domain rich model (SRM) [15], many methods have been proposed and applied successfully in the field of image tamper detection [16]. By simply merging the detection results of a statistical feature-based detector and a copy-move detector, the good performance is achieved for forgery detection [16]. In order to identify the authenticity and processing history of an image, a new forensic approach using CNN and new CNN layer, called a constrained convolutional layer, is proposed in [17]. And the experimental results demonstrate that this model can detect multiple different editing operations with up to 99.97%. Although the interest of neural network in image forensics domain is growing, a real comprehension of what is possible to accomplish with it is still in an early stage.

Since the use of deep learning approaches for multimedia security applications is still in its infancy, many of current detection algorithms only deduce that whether a given image has been manipulated and do not attempt to localize the tampered regions which require to determine which pixels in an image have been manipulated. However, in order to formulate the framework, the convolutional layers along with long-short term memory (LSTM) cells are exploited to capture discriminative features in [18]. And the CNN-LSTM model shows promising results in localizing manipulated regions. An effective solution to the splicing localization problem based on fully convolution network (FCN) is presented in [19]. The proposed model is based on the FCN VGG-16 network architecture incorporating several modifications, which outperforms some existing splicing localization algorithms.

Since in practical forensics applications, figuring out the tampered regions compared to forgery detections is more important and necessary. In this paper, our objective is to train a network model that, given a to-be-tested image, is able to reliably localize the possible tampering regions.

**Algorithm 1** Training the Proposed *D-CNNs* Model

**Input**: the training set of pristine patches $\boldsymbol{Au\_set}$, the training set of forged patches $\boldsymbol{For\_set}$, the 14 high-pass filters $\boldsymbol{F}$, the untrained *D-CNNs* $\boldsymbol{D}$, the paremeters of pre-trained *Sub-SCNN* model $\boldsymbol{P_S}$, the paremeters of pre-trained $Sub-FCNN$ model $\boldsymbol{P_F}$.

```
Preprocess Au_set and For_set with the 18 high-pass filters F
Calculate the statistic features of Au_patch and For_patch
Transfer the parameters and use them directly in D
i = 1
While i ≤ max_iter do
    Do feedforward pass
    Upadete the filter weights of D through Adam optimizer with Eq.(6) and backpropagate errors
    i = i + 1
    If training accuracy converges then
    Exist
end
```

**Output**: a well trained *D-CNNs* $\boldsymbol{D}$

To improve the performance of detecting and localizing manipulated image regions, different kinds of CNNs-based approaches have been presented to classify the image patch and pixel-wise segmentation, and different inputs to the network are taking into consideration. We perform end-to-end training to learn the discriminative features between manipulated and non-manipulated regions through back-propagation using the ground truth labels and image mask information. The proposed model shows promising results in patch classification on validating datasets, as well as in localizing manipulated regions at pixel level on testing datasets. Our main contributions can be summarized as:

- A dual domain-based CNN architecture is proposed that takes different kinds of input to the CNN into consideration. And in the proposed *D-CNNs*, the two sub-networks of the *Sub-SCNN* which is exploited perform image manipulation detection and localization starting from the RGB color images and the *Sub-FCNN* which is introduced taking statistical features based on the 3 level Daubechies-based Discrete Wavelet Transformation (DWT) as input to the net are designed in the different domain;
- In *Sub-FCNN* framework, four additional statistical features are introduced for each different DWT sub-band over all three color channels to both further improve the accuracy of patch classification in the *Sub-FCNN* and the *D-CNNs* models;
- In the proposed framework, two sub-networks of *Sub-SCNN* and *Sub-FCNN* are designed and trained respectively. With the parameters of pre-trained *Sub-SCNN* and *Sub-FCNN* networks, a transfer policy is applied to the

training process of *D-CNNs* to achieve higher accuracy and avoid a long time training process under the normal learning policy;
- According to the properties of image datasets, two different post-processing operations are applied on different datasets to finalize the pixel-wise manipulation region localization, which can reduce the false detection rate and locate the edge contour of tampered region more precisely.

The remainder of this paper is organized as follows. An overview about CNNs in literature is presented in Section 2. In Section 3, the algorithm framework is described and its details are introduced briefly. Section 4 discusses our main contributions and assesses our proposed *D-CNNs* architecture through a set of experiments. Finally, the conclusion is provided in Section 5.

## II. CONVOLUTIONAL NEURAL NETWORKS

Deep learning approaches, such as convolutional neural networks, can provide the end-to-end analysis. In general, their architecture, which is the set of learnable parameters (e.g. weights and biases) and components that we need to design in a network, is a cascade of alternating four basic layers: convolutional layers, activation layers, pooling layers and regulation layers (e.g. BN: batch normalization layers).

— *Convolutional layer*. In the CNN architecture, the convolutional layer is to use $k$ filters to convolve the input images, generating $k$ new feature maps for subsequent processing. Let us denote the $j^{th}$ output feature map in $n^{th}$ layer by $F_j^{(n)}$, we have:

$$F_j^{(n)} = \sum_i w_{ij}^{(n)} * F_i^{(n-1)} + b_j^{(n)} \tag{1}$$

where $*$ denotes a 2d convolution, $w_{ij}^{(n)}$ and $b_j^{(n)}$ represent the convolutional filter and bias respectively.

— *Activation layer*. The convolutional layers are also followed by an activation layer to transform the input feature map through nonlinear mapping.

$$F_j^{(n+1)} = f(F_j^n) \tag{2}$$

where $f$ () is a point-wise activation function, such as Tanh, ReLU [20], ELU [21], and leaky ReLU (LReLU) [22], etc. The activation layer also called the nonlinear mapping layer. As the name implies, the activation function is introduced to increase the expressive power of the entire network, that is, non-linearity. Otherwise, the stack of several linear operation layers can only play the role of linear mapping and cannot form complex functions.

— *Pooling layer*. The set of convolutional layers yields a large volume of feature maps. To reduce the dimensionality of input features and make the extracted features compact, the pooling layer is also added after the convolutional layers and is defined as:

$$F_j^{(n+1)} = pool(F_j^n) \tag{3}$$

where *pool* () represents the pooling function. Generally, there are three kinds of pooling function in existing CNN models: max, average and stochastic pooling. In particular, the max-pooling layer works as a sliding window with a stride distance which selects the maximum value within the sliding window.

— *Regulation layer*. In addition, to enforce the data far away from saturation regions, the regulation layers such as dropout layers and Batch Normalization (BN) are preferred to use in the state-of-the-art CNN models [23]. In particular, in the BN layer, the two learnable parameters $\{\gamma, \beta\}$ are introduced to ensure that the feature distribution is not corrupted. Through them, each data item $x_i$ in a mini-batch $B = \{x_1, x_2, \ldots, x_m\}$ of size $m$ is transformed into $y_i$:

$$y_i = \gamma \hat{x}_i + \beta \tag{4}$$

$$\hat{x}_i = \frac{x_i - E_M(x_i)}{\sqrt{Var_M(x_i) + \varepsilon}} \tag{5}$$

In (5), $E_M(x_i)$ and $Var_M(x_i)$ represent the mean and the variance of $x_i$ in the batch $B$. The main function of the batch normalization makes a network converge in a faster speed than a network without batch normalization.

The non-linear activation, pooling, and regulation operation are optional in a specific layer. For a classification problem, on the top of the several cascaded convolutional layers, there are one or more fully-connected layers followed by a soft-max classifier that performs classification.

The training of a CNN model is an iterative process that is completed by alternating between feed-forward and back-propagation operations. Like many other machine learning models (support vector machines, etc.), the ultimate aim of convolutional neural networks is relying on minimizing loss functions to learn model parameters. In this paper, cross entropy loss function is employed to calculate the error between the true class labels and the network outputs. There are a variety of solvers to minimize average loss. The stochastic gradient descent (SGD) optimizer is considered to train our networks. The adaptive updating process of weight $w_{ij}^{(n)}$ at each iteration of the back-propagation algorithm is shown as follows:

$$\nabla w_{ij}^{(n+1)} = \lambda \cdot \frac{\partial L}{\partial w_{ij}^{(n)}} - m \cdot \nabla w_{ij}^{(n)} + d \cdot \lambda \cdot w_{ij}^{(n)} \tag{6}$$

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} - \nabla w_{ij}^{(n+1)} \tag{7}$$

where $\nabla w_{ij}^{(n)}$ denotes the gradient of $w_{ij}^{(n)}$ and $\lambda$ is the learning rate. $L$ is the loss function. The bias $b_j^{(n)}$ in (1) also undergo the above iterative update process like weights $w_{ij}^{(n)}$. For fast convergence as explained by LeCun *et al*. [24], the weight decay and momentum are used in this paper and they are denoted by $d$ and $m$ respectively in (6).

The structural advantages of CNNs make them suitable for capturing the statistical properties of the lower and deeper layers of the image. Therefore, the CNN models are widely used in image-related tasks and achieve excellent performance.

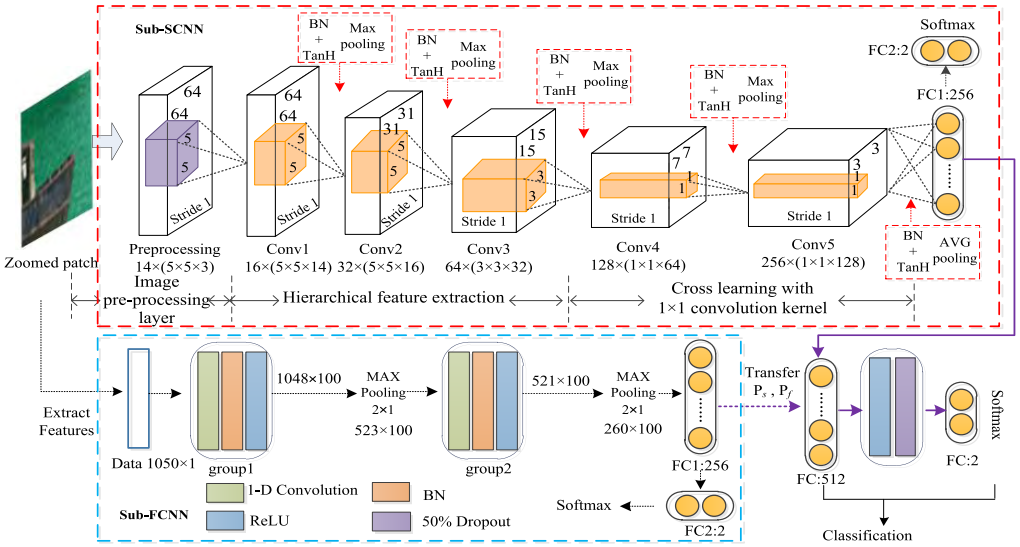



**FIGURE 1.** Architecture of the D-CNNs; The Sub-SCNN marked with red dotted line (in the top row); The Sub-FCNN marked with blue dotted line (in the bottom row).

## III. PROPOSED METHOD

### A. NETWORK ARCHITECTURE

Many existing state-of-the-art approaches perform image manipulation detection proceed by extracting discriminant features directly from data. However, in order to accurately localize tampered regions, it is also essential to learn the inherent characteristics of the tampered regions. Therefore, the *D-CNNs* architecture is designed that takes different kinds of input to the CNN into consideration in this paper. The overall framework of the proposed D-CNNs method with details about the size of each layer is shown in Fig 1.

The architecture of *D-CNNs* consists of two different sub-networks: *Sub-SCNN* and *Sub-FCNN* up to their respective first fully connected layer and has the ability to learn the inter-model relations between features extracting from the R, G, B three channels in the spatial domain and from the three-level Daubechies-based DWT in the frequency domain. Then the output of the fully connected layers of the two nets (256 dimensions each) are joined together is fed to the classification block which consists of two fully connected layers, a ReLU layer, and a dropout layer. More specially, the first fully connected layer with 512 output neurons followed by a ReLU layer and a dropout layer generate a vector of 512 elements. At last, the output of the final fully connected layer with two neurons is sent to a two-way softmax connection, which can produce the probability that the input patch is classified into tampered class and authentic class.

As can be seen from Fig 1, the input of our proposed *D-CNNs* is a three-channel color image patch sized $64 \times 64$ pixels, and primary part of the proposed architecture is two sub-networks. In what follows, the overview of each presented sub-network as well as the different layers are introduced in detail.

#### 1) SPATIAL DOMAIN-BASED CNN

A CNN-based approach, named spatial-domain CNN (*Sub-SCNN*), is proposed in this section, and its architecture is shown in Fig 1 with the red dotted line. One can notice from spatial network that the main body of the network can be divided into three parts: image pre-processing layer, hierarchical feature extraction and cross learning with $1 \times 1$ convolution kernel.

- *Image pre-processing layer*: Different from the research of semantic segmentation, the image tampering localization approach proposed in this paper needs to locate the tampering areas instead of every object in the image. By contrast, our approach would like to suppress the image's content and adaptively learn the traces left by image manipulation operations. To accomplish this, the first convolutional layer of CNN model in [15] is served as the pre-processing to capture the changes of local pixel relationships. Up to now, it is observed that this strategy is adopted in many existing forensics and steganography algorithms which use all SRM kernels proposed in [25] to initialize the filters in the first layer. Based on our extensive experiments, we find that the detection performance would not increase significantly with increasing the number of the employed filters. Considering the trade-off between the model computational complexity and detection performance, 14 filters which are formed by five basic filters (in Fig 2) used in the calculation of the residual maps in SRM and their rotations filters, are selected in our image pre-processing layer. To capture the differences of adjacent pixels in the different direction, as for the filter (a), it takes the center as the origin and is clockwise rotated by 90°, 180°, and 270° to form three new high pass filters with the first order. And as for the filter (b), three new high pass filters

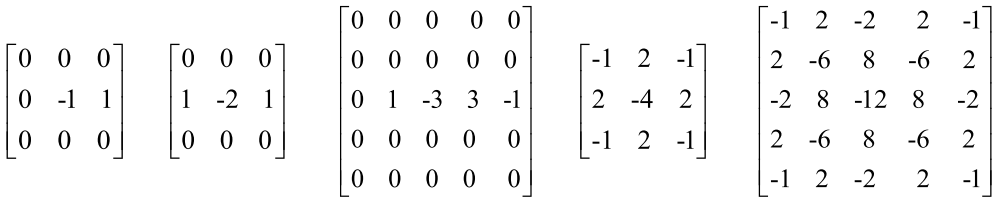



**FIGURE 2.** The basic filters used in channel expansion layer.

with the second order can be obtained by rotating filter (b) along the vertical, diagonal, anti-diagonal directions. The filter (c) also undergoes the above process like filter (b). As described in SRM, the 14 basic filters can be divided into 5 residual classes, which include 4 filters in class "1st," 4 in the class "2nd," 4 in the class "3rd," 1 in class "SQUARE3 × 3" and 1 in class "SQUARE5 × 5." And the 14 filters are formulated as $5 \times 5$ weight matrices $\{W_1, W_2, \ldots, W_{14}\}$ filled with unused elements with zero.

In our CNN architecture, the three-channel color patches are the input of the image pre-processing layer. Therefore, each of the 14 output feature maps corresponds to 3 weight matrices of size $5 \times 5$, and 42 weight matrices in total are needed to be initialized with above-mentioned 14 filters. To make the tampering traces exposed to a larger extent by pre-processing layer, the 3 basic filters used for each of output feature maps should be similar but not identical.

For the $j_{th}$ output feature map, the corresponding weight kernel $\mathbf{W}^j = \begin{bmatrix} W_1^j & W_2^j & W_3^j \end{bmatrix}$ is initialized as:

$$\mathbf{W}^j = \left[ W_{((3*j-3) \bmod 14)+1} W_{((3*j-2) \bmod 14)+1} W_{((3*j-1) \bmod 14)+1} \right] \tag{8}$$

where $j$ equals from 1 to 14. Expect for the image pre-processing layer, the Gaussian distribution with 0 mean and standard deviation 0.01 is adopted for the initialization of weights in other layers.

- *Hierarchical feature extraction*: To learn higher-level prediction error features, three regular convolutional layers, namely "Conv1" with 16 filters of size $5 \times 5 \times 14$ and a stride equal to 1, "Conv2" with 32 filters of size $5 \times 5 \times 16$ and stride of 1 and "Conv3" with 64 filters of size $3 \times 3 \times 32$ and stride of 1 respectively, are used and each of them followed by a batch normalization layer, an activation function layer and a pooling layer. In the previous section, an "image pre-processing layer" yields the resulting residuals and increases the channel number of feature maps of the prediction residual from 3 to 14 in this layer. The 14 residual feature maps are fed as input to the model part of hierarchical feature extraction. Then the output dimensions of these three convolutional layers are $64 \times 64 \times 16$, $31 \times 31 \times 32$ and $15 \times 15 \times 64$. Different sizes of convolution kernels are used to learn the new representation of different feature maps. In this part, the max pooling layer with an overlapping kernel of size $3 \times 3$ and stride of 2 is used after each regular convolutional layer. The three max-pooling layers reduced the dimensions of feature maps from $64 \times 64 \times 16$ to $31 \times 31 \times 16$, from $31 \times 31 \times 32$ to $15 \times 15 \times 32$ and $15 \times 15 \times 64$ to $7 \times 7 \times 64$. Therefore, we can see that the pooling operation can reduce the feature maps dimensions and computational cost of training process. More specifically, it can retain the most representative features by calculating the max value within a local $3 \times 3$ sliding window to improve the accuracy.
- *Cross learning with* $1 \times 1$ *convolution kernel*: The hierarchical features are learned by learning local spatial association within the different local receptive field in the same feature maps. Next, the new association is learned by cross learning with $1 \times 1$ convolution kernel between these feature maps. For a pixel, a $1 \times 1$ convolution is equivalent to performing a fully connected calculation on all features in the feature maps, simulating more nonlinear features. This has been demonstrated to improve the learning ability of CNN in image steganalysis [26].

This part of cross learning consists of two convolutional layers, namely "Conv4" and "Conv5" respectively, and each of them also followed by a batch normalization layer, an activation function layer and a pooling layer. The "Conv4" layer with 128 filters of size $1 \times 1 \times 64$ applied with stride 1 generates $7 \times 7 \times 128$ feature maps. After a batch normalization layer and a ReLU layer, a max-pooling with the kernel size of $3 \times 3$ applied with stride 2 produces $3 \times 3 \times 128$ feature maps. Similarly, the "Conv5" layer with 256 filters of size $1 \times 1 \times 128$ applied with stride 1 generates $3 \times 3 \times 256$ feature maps. After a batch normalization layer and a ReLU layer, a max-pooling with kernel size of $3 \times 3$ applied with stride 2 produces a vector of 256 elements.

After the cross learning, the deepest convolutional features learned by the previous layers as input are classified primarily passed to two fully connected layers. The first fully connected layer with 256 output neurons produces the 256 element feature vector. And the last fully connected layer with two output neurons followed by a soft-max layer acts as logistic regression classifier during the *Sub-SCNN* training stage. The classification layer output two classes, a class of tampering

image and another class that corresponds to the unaltered image class.

### 2) FREQUENCY DOMAIN-BASED CNN

In the secondly proposed sub-network, *Sub-FCNN*, a statistical feature extraction is performed on a given input patch calculating from the DWT following the idea in [12] as the input of the sub-network. And the tampering traces and certain correlation in natural images are captured by DWT, which transforms an image from spatial to the frequency space and generates different scale sub-bands [27].

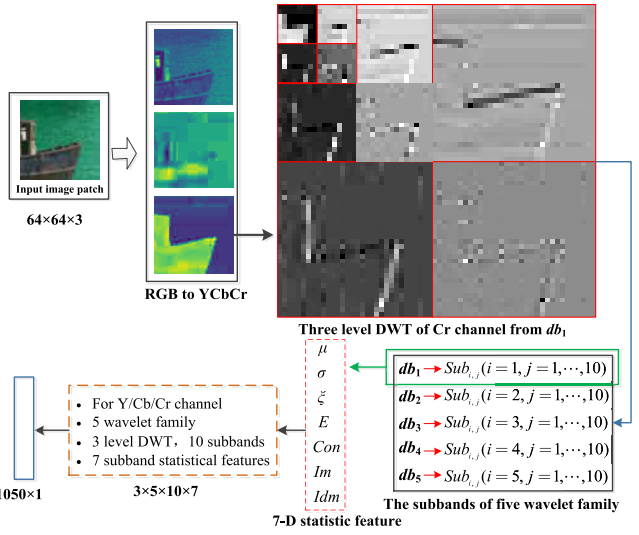



**FIGURE 3.** The workflow of image patch statistical feature extraction.

As is shown in Fig 3, in detail, a 64 × 64 × 3 raw image patch is firstly converted into YCbCr color space, and for each channel, a three-level Daubechies-based DWT is applied. At each scale, three different sub-bands are generated including the horizontal, vertical and diagonal orientations. Then ten sub-band coefficient matrices can be obtained by the multi-level wavelet decomposition. And a family of daubechies wavelets is adopted to preserve sufficient correlation of coefficients among multiple vanishing moments. Therefore, for each channel, fifty sub-bands can be generated by applying the 3 level Daubechies-based DWT from $db_1$ to $db_5$ ($db_i$ represents Daubechies wavelet with m vanishing moments) and the $j^{th}$ sub-band of $db_i$ wavelet is denoted by $Sub_{i,j}(i \in [1,5], j \in [1,10])$. Instead of directly using all these wavelet sub-band coefficients as the raw input, three statistics which represent the relationships among the basic statistic among different sub-bands are selected in [28]. Inspired by the above ideas, four new features are introduced to form seven features in our paper. As different features can capture different information regarding the structural arrangement of surfaces or changes in intensity or color brightness [29], they together can better describe the inherent characteristics of image patches. Specifically, for each DWT sub-band $Sub_{ij}$, seven statistics are extracted and given by

$$f = <\mu, \sigma, \xi, E, Con, Im, Idm> \tag{9}$$

$$\mu = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i,j) \tag{10}$$

$$\sigma = \sqrt{\frac{1}{MN-1}\sum_{i=1}^{M}\sum_{j=1}^{N} |p(i,j)-\mu|} \tag{11}$$

$$\xi = \sum_{i=1}^{M}\sum_{j=1}^{N} p^2(i,j) \tag{12}$$

$$E = -\sum_{i=1}^{M}\sum_{j=1}^{N} P(i,j)\log(p(i,j)) \tag{13}$$

$$Con = \sum_{i=1}^{M}\sum_{j=1}^{N} (i-j)^2 p(i,j) \tag{14}$$

$$Im = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{1}{|i-j|} p(i,j)(i \neq j) \tag{15}$$

$$Idm = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{1}{1+(i-j)^2} p(i,j) \tag{16}$$

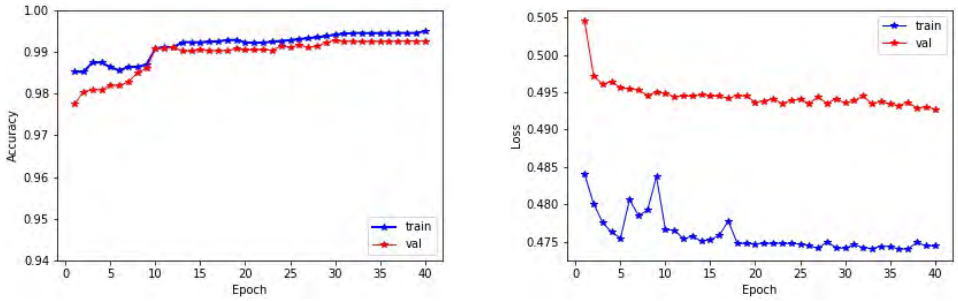
where $\mu$ is the average of elements in the DWT sub-band coefficient arrays. $\sigma$ quantifies the amount of variation of the DWT coefficients, $\xi$ is the coefficient energy which can measure the amount of information retained in the considered sub-bands, $E$ measures the discrepancy of elements in the DWT sub-band coefficient arrays, $Con$ reflects the sharpness of the DWT sub-bands, $Im$ measures the degree of changes in the local coefficient arrays and $Idm$ measures the homogeneity whose larger value stands for small differences of elements in the DWT sub-bands. The formulations to extract these statistics can be defined as follows where $p(i,j)$ denotes a DWT sub-band coefficient array to be analysed with the size of $M \times N$.

In the process of statistical feature extraction, for an input image patch, the feature vectors extracted in different channels can be concatenated together into $p$, which will finally be expressed as the following vector:

$$p = <f_Y(Sub_{i,j}), f_{Cb}(Sub_{i,j}), f_{Cr}(Sub_{i,j}) | i \in [1,5], j \in [1,10]> \tag{17}$$

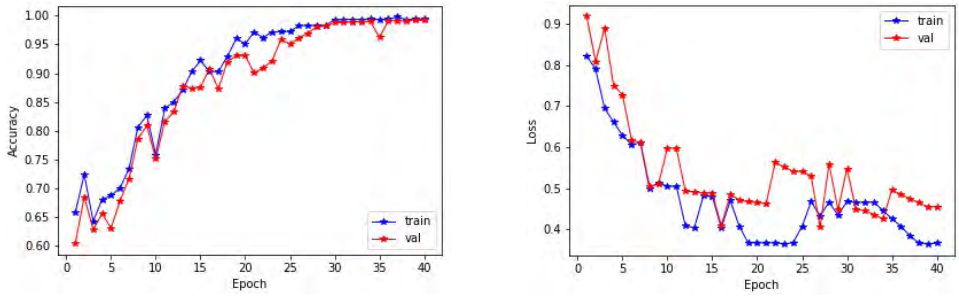
where $f_Y(Sub_{ij})$ denotes the 7-D statistical feature extracted in the $j^{th}$ sub-band of $db_i$ wavelet over the Y channel. By applying a three-level DWT from $m$ daubechies wavelets to an image block on a channel, the features that can be generated are a total of 350 dimensions. Therefore, the 1050-dimension vector which contains 450-dimension used in [28] is formed over all three color channels and the overall processing workflow is shown in Fig 2.

The 1050-dimension feature vector is then used as input to train the proposed *Sub-FCNN* model, in order to distinguish




(a) (b)

**FIGURE 4.** Transfer learning policy: accuracy (a) and loss (b) in each epoch in the training process of the D-CNNs.




(a) (b)

**FIGURE 5.** Normal learning policy: accuracy (a) and loss (b) in each epoch in the training process of the D-CNNs.

whether the input image patch is authentic or tampered. The architecture of *Sub-FCNN* is illustrated in Fig 1 with blue dotted line.

One can notice from *Sub-FCNN* network that the extracted features are firstly fed to the group1 block which has a 1-D convolutional layer (Conv), a batch normalization layer (BN) and a ReLU activation layer following by a max pooling layer. Similarly, the output of the first pooling layer will be sent to the group2 block with "Conv+BN+ReLU" followed by the second pooling layer. The first fully connected layer contains 256 neurons and the number of neurons in the output layer is equal to the number of classes in the considered forensics task. The output layer is followed by a soft-max activation function and the class of the input image patch corresponds to the higher activated neuron in the output layer.

### B. NETWORK TRAINING

A transfer learning policy is used during the training process of the *D-CNNs* network. Different from the traditional parameter initialization, the parameters of pre-trained *Sub-SCNN* network and *Sub-FCNN* network are used directly in the transfer training policy network. And as described in Section 3, the parameters of the image pre-processing layer are fixed and initialized with selected 14 filters. While in normal training processing, the parameters of *Sub-SCNN* and *Sub-FCNN* networks are initialized with random values which belong to the normal distribution. Before using the transfer learning policy, we need to respectively train the *Sub-SCNN* and *Sub-FCNN* networks on the same image dataset. Until each of sub-network converges, the parameters of layers that are trained in the different domain networks, such as the convolutional layer, BN layer, and fully connected layer, are saved. We separately denote the parameters of *Sub-SCNN* and *Sub-FCNN* model as $\boldsymbol{P_S}$ and $\boldsymbol{P_F}$. On the same image dataset, the whole process to train the proposed *D-CNNs* model is illustrated as shown in Algorithm 1.

Two experiments are performed to verify the validity of the transfer training policy. Under the transfer training policy, the accuracy and loss in each epoch of the proposed *D-CNNs* which is trained on CASIA2 dataset is illustrated in Fig 4. The accuracy in the training process is defined as the proportion which is calculated by the number of tampered or authentic patches being correctly classified divided by the number of training patches. Since we use the parameters from pre-trained two sub-networks, the validation accuracy is high enough at the very beginning of the training process, and it remains a stable trend after epoch 10. Another experiment is performed that parameters of convolutional layers, BN layers and fully-connected layers in the *D-CNNs* network are initialized to random values which belong to normal distribution. And the accuracy and soft-max loss of the proposed *D-CNNs* with normal training policy are shown in Fig 5. Compared to the former case, it takes much longer training time to reach a stable trend around epoch 35. The accuracy of

transfer learning networks is slightly higher than that of normal learning networks. Therefore, we use the trained networks under transfer learning policy for the proposed *D-CNNs* training.

### C. MANIPULATION REGION LOCALIZATION

Through the description in the previous section, a well-trained *D-CNNs* model can be obtained, which can predict the class of the input image patch. For each image to be detected, the proposed framework of manipulation region localization consists of the following two steps.

- *Producing the initial tampering possibility map:* For each image to be tested, it is analysed by the sliding window of the scale as $s \times s$ with a stride of $st$ ($s = 64$ and $st = 8$ in our paper). The image patch size is chosen empirically as too small size cannot capture sufficient information while too large size cannot localize the tampered regions precisely. Then, a source image to be analysed (denoted by $S$) with the size of $h \times w$ is divided into $h^{fea} \times w^{fea}$ image patches by the sliding window manner, where $h^p = floor\ ((h - s)/st)+1$ and $w^p = floor\ ((w - s)/st)+1$. Therefore, a tampering possibility map $M^p$ of size $h^p \times w^p$ where each element indicates the probability that the corresponding patch is tampered or authentic. At the same time, a matrix $N$ of the same size as $S$ is also generated, and each pixel value in the matrix records the number of patches containing pixel at this position in the $S$ by conducting in a sliding window manner. Inevitably, for some pixels in $S$, the corresponding value in $N$ is equal to zero, which means that the number of blocks containing them is zero and these pixels usually appear around the edges of the image. In order to get the tampering possibility map $M$ with the same size of the input, the element $m_{i,j}$ in $M$ is computed as

$$m_{i,j} = \frac{m^p}{n_{i,j}}(n_{i,j} \neq 0) \tag{18}$$

where $m^p$ is the tampering probability value of the patch containing pixel $S_{i,j}$ in $S$, and $n_{i,j}$ denotes the corresponding value in $N$. When $n_{i,j} = 0$, we simply set $M_{i,j}$ the same probabilities as the nearest pixel whose $n_{i,j}$ is not equal to 0. To smooth the mosaic artifacts in the possibility map $M$, mean filtering is applied on the $M$ by (19) So that the smoothed possibility map $\overline{M}$ can be obtained. Thereafter, given an image, we output an initial possibility map $\overline{M}$ with the same image size of the input by conducting a block-wise prediction in a sliding window manner by the well-trained *D-CNNs* model.

$$\overline{m}_{i,j} = \frac{1}{s \times s} \sum_{i'=-\frac{s}{2}}^{\frac{s}{2}-1} \sum_{j'=-\frac{s}{2}}^{\frac{s}{2}} m_{i+i',j+j'} \tag{19}$$

- *Optimize the tampering possibility map to finalize the pixel-wise manipulation region localization:* In order to reduce the false detection rate and locate the edge contour of the tampered region more precisely, two different post-processing operations are applied on different datasets according to the properties of image datasets in this paper.

For the CASIA v2.0 dataset, the tampered region has certain semantics such as an animal, a car and such contextual information can be referred from the consistency between the nearby pixels in the spatial domain. And the consistency and correlation among nearby pixels are used to divide the image region in some clustering algorithms. Thus, three experiments based on different classical image segmentation algorithms are performed to divide the image into different segments. For each segment $s$, the final prediction label $L$ is determined by comparing the average tampering probability value of pixels $(i, j) \in s$ with a pre-defined threshold as

$$L(s) = \begin{cases} 1, & \frac{1}{num_s} \sum_{(i,j)\in s} \overline{M}(i,j) \geq \beta \\ 0, & otherwise \end{cases} \tag{20}$$

where $num_s$ calculates the total number of pixels in the given segment $s$ and $\overline{M}(i, j)$ represents the tampering probability value of the pixel $(i, j)$. In this paper, $\beta$ is set 0.5, which is a midpoint of the maximal probability. $L(s)$ labeled as 1 represents tampered segment, vice verse.

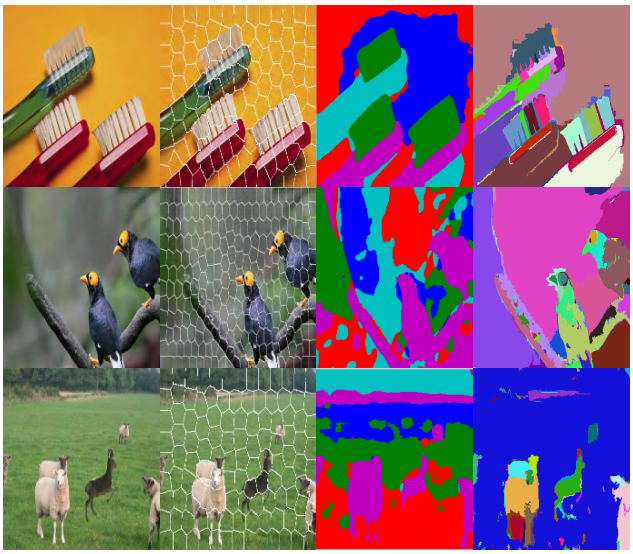


**FIGURE 6.** Segmentation results of three images selected from CASIA2. Starting from the first column: 1: input image, 2: results of SLIC [30], 3: results of image segmentation algorithm [31], 4: results of graph-based image segmentation algorithm [32].

Fig. 6 shows the segmentation results of three examples. Instead of manually setting the number of clusters in the traditional SLIC (Simple Linear Iterative Clustering) segmentation algorithm, the number of clusters of the image is calculated according to the complexity of the image in [30]. However, as shown in Fig 1, we can see from the segmentation results in the second column that the edges of some objects in the image are not very accurate, such as the bristles

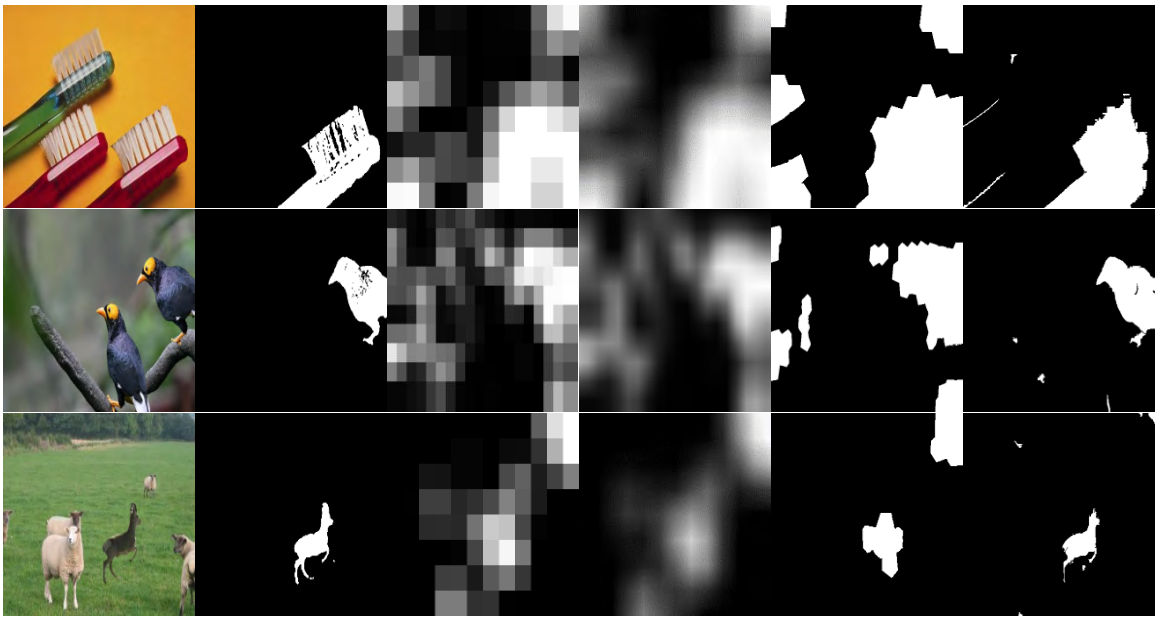


**FIGURE 7.** Three examples to show how the proposed model localizes the tampered region in pixel-level: Starting from the first column: 1: forged images used in experiments, 2: ground truth, 3: initial tampering possibility map, 4: smoothed tampering possibility map, 5: binary tampering possibility map, 6: the final integration result (the white regions indicate the area that was predicted as tampered using the proposed model.).

of the green toothbrush, the mouth of birds, and the head of goats. Moreover, when the image texture is rich, the super pixel block is too small to effectively remove the misclassified areas. When the image contains more smooth regions, the super pixel block is too large to locate the small tamper regions precisely.

The segmentation result of the third column shows that the similar texture regions are all divided into the same portion, which will cause the regions having similar textures with the tamper regions to be marked as tamper regions, thereby reducing the detection rate. As shown in the segmentation results in the third column, the graph-based image segmentation algorithm [31] not only divides the different image regions precisely and meticulously, but also similar texture regions are divided into different segments. Thus, to finalize the pixel-wise forged region localization, we leverage a segmentation technique [32] is leveraged to perform such image region division.

Based on the above, three prediction examples are shown in Fig 7. In order to compare with post-processing operation based on segmentation, the directly binarized tampering possibility map marks the pixel in the tampering possibility map as 1 when its value is greater than or equal to 0.5 and marks the pixel in the tampering possibility map as 0 when its value is less than 0.5. The qualitative differences between binarized post-processing and the segmentation-based post-processing operation are visualized by our experimental results. By integrating the initial probability map and the selected segmentation algorithm, it can be seen that the final fusion result can significantly optimize the tamper probability map and reduce the regions that are mistakenly classified.

However, considering that the characteristics of the tampering regions in Columbia and Carvalho datasets are different from characteristics of tampering regions in CASIA v2.0, a simple post-processing with the binary operation is applied on tampering probability map in Columbia and Carvalho. The specific theoretical analysis and the comparison of experimental results are introduced in Section 4.7.

Finally, our proposed image manipulation localization process can be summarized as the following steps:

Step 1: Train the *Sub-SCNN* and *Sub-FCNN* models separately to learn the optimal features for tampered regions.

Step 2: Determine the probability of each image block being tampered with using *D-CNNs* by transferring pre-trained parameters of spatial and frequency domains in a slide window manner and normalize the accumulated values to compose an initial tampering probability map with the same size as the input image.

Step 3: For the test image, obtain its segmentation results which are a set of segments. For each segment, determine whether it is tampered using (20).

Step 4: Post-processing operation based on segmentation is used to refine the initial tampering possibility map and the final result of image manipulation localization is obtained.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

To assess the performance of our proposed *D-CNNs* for performing image manipulation detection and localization, a set of experiments and analysis are conducted. In these experiments, the contributions of additionally introduced four statistical features in *Sub-FCNN* are first evaluated. Next, we compare the performance of our proposed architecture

with different structural design choices, e.g., the choice of pooling and activation functions. Finally, some experimental results on different datasets are extrapolated and presented to provide a qualitative view of the achieved performance mainly in terms of manipulation localization.

### A. DATA COLLECTION

#### 1) IMAGE DATASET

The experiments are conducted on the tampered images of three public domain benchmark databases: CASIA Tampered Image Detection Evaluation Database Version 2.0 (CASIA v2.0) [33], Columbia Uncompressed [34] and Carvalho [35], which are open for downloading and widely used in image manipulation localization. Ground truth masks are provided for the Columbia Uncompressed and Carvalho. For a tampered image in the CASIA v2.0 dataset, it involves human efforts to manually label the forged region using the provided reference information about its donor and host images. The number of tampered images in Columbia and Carvalho are 180 and 100 respectively, and some sample tampered images on CASIA v2.0 are shown in Fig 1.

#### 2) PATCH SAMPLING

As introduced in subsection 4.2, the input of the training process of the *D-CNNs*, the *Sub-SCNN* and the *Sub-FCNN* is labeled patches sample from the training images. And, the sliding window manner is adopted to get the tampering possibility map of the investigated image, thus we should sample patches on the images for training and validation. Then we extract patches from training images which are considered as training set. Similarly, the validation set is also obtained in this process. During the patches generation, the sliding window with size of $64 \times 64$ and the stride $st$ is set as 2. In order to augment the manipulated patches and reduce the amount calculation, we first obtain the bounding box that includes the whole manipulated region and some non-manipulated region on the tampered images by combing the corresponding ground truth mask. The sliding window with a fixed scale slides across the tampered image region in the box, and the patches tampered with 30% to 90% are labeled as manipulated. Some sampled manipulated patches contain some part of the non-manipulated region, which are useful for the proposed model to learn the transformation from the manipulated region to non-manipulated region. However, the proportions of the tampering area to the entire image differ greatly. In some images, more than ten thousand patches can be generated, while in some images, several or dozens of patches can be generated with stride of $st = 2$. To prevent overfitting caused by the imbalance of patches distribution in the training procedures, we set upper threshold $T$ for patch sampling. While more than $T$ patches are sampled, we randomly select $T$ patches, and $T$ is set 500 in our paper. After the manipulated patches are generated, the pristine patches without any tampered pixels are sampled in the same tampered images with a stride of $st = 10$, and its number is not higher the $T = 500$. And the numbers of training and testing images are shown in Table 1.

**TABLE 1. The numbers of sampled patches on three datasets.**

| Dataset | Type | No. of Tampered Images | |
|---|---|---|---|
| CASIA v2.0 | Training | 3330 | 5123 |
| | Validating | 512 | |
| | Testing | 1281 | |
| Columbia | Testing | 180 | |
| Carvalho | Testing | 100 | |

Before training on the CASIA v2.0 dataset, we first split the whole tampered images in every dataset into three subsets – training (65%), validation (10%) and testing (25%), and these subsets are randomly selected. Through the method of patch sampling mentioned above, the pristine and manipulated patches of the training subset and validation subset are respectively collected, then the training set of pristine patches ***Au_set*** and the training set of forged patches ***For_set*** are formed. With these newly generated patches, the whole network is trained end-to-end. Specifically, in our experiments on the CASIA v2.0 dataset, the training, validation and test data is randomly divided three times and the average results are reported in the following section.

### B. IMPLEMENTATION DETAILS

To train the proposed model, TensorFlow [36] is used to define different layers of the network. All the experiments are conducted on a machine with Intel (R) Core (TM) i7-7800X CPU @ 4GHz, 64GB RAM and two NVIDIA GeForce GTX 1080 Ti GPUs. In our experiments, the momentum $m$ is fixed to 0.9 and $L2$ regularization is used. The corresponding weight decay $d$ is 0.0005. All weights are initialized by random numbers generated from the Gaussian distribution with 0 mean and standard deviation of 0.01, and all the biases are initialized with 0. We shuffled the training set between epochs.

Besides, the step decay of learning rate is used in our experiments. The learning rate is initialized 0.001 and scheduled to decrease 10% when the validation accuracy stopped improving, and we stop the training after reducing the learning rate three times. In training process of proposed *D-CNNs*, it is decreased for every 10000 iterations. And with the well-trained *D-CNNs*, the probability of pristine or manipulated is obtained for each input patch.

### C. EVALUATION METRICS

With the well-trained *D-CNNs*, for each test image, a binary mask indicating the tampered and pristine regions can be generated. In this paper, by regarding the spliced pixels as positive samples and the authentic pixels as negative ones, the precision, recall, fallout, accuracy, *F1*-Measure, *MCC* (Mattews Correlation Coefficient) which are per-pixel localization metrics are utilized to evaluate the performance of the

proposed methods, and they are defined as follows:

$$Precison = \frac{TP}{(TP + FP)} \tag{21}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{22}$$

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{23}$$

$$F1 = \frac{2TP}{(2TP + FP + FN)} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{24}$$

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{25}$$

where *TP* (True Positive) is the number of tampered pixels classified as tampered; *FP* (False Positive) is the number of authentic pixels classified as tampered; *TN* (True Negative) is the number of authentic pixels classified as authentic; and *FN* (False Negative) is the number of tampered pixels classified as authentic.

### D. CONTRIBUTIONS OF ADDITIONALLY INTRODUCED FOUR STATISTICAL FEATURES IN SUB-FCNN

As introduced in subsection 4.1.2, for each different DWT sub-band, four additional statistical features are introduced based on three features. Therefore, the 1050-dimension vector which contains the basic 450-dimension used in [28] is formed over all three color channels. In our first experiment, the effect of four additional features which are extracted from different sub-bands over three channels to form 600-dimension vector is examined when performing patch classification for the validate set from CASIA 2.0. To conduct the experiment, 65% of tampered images are selected randomly, and their corresponding sampled forged and pristine patches are used as input for training the proposed three networks, and the 10% of tampered images are selected and their corresponding sampled forged and pristine patches as the validate set to evaluate the contributions. And results are shown in Table 2 and Table 3. Note that the accuracy metric reported in Table 2 and Table 3 is the proportion of all patches in the validate set that are correctly classified. In other words, it is per-patch classification evaluation indicator.

**TABLE 2. Results of patch classification for CASIA 2.0 in validate set.**

| The Proposed the Network Model | Accuracy |
|---|---|
| Sub-FCNN CNN (450-D) | 94.70% |
| Sub-SCNN | 98.54% |
| D-CNNs | **98.89%** |

Table 2 and Table 3 show the accuracy of *Sub-SCNN* is 98.54%. We compare the results of the *Sub-FCNN* with 450-D feature vector as the input to our proposed the *Sub-FCNN* trained on 1050-D vector in the two Tables. It is obvious that the accuracy of patch classification achieved by the 1050-D features as the input in the *Sub-FCNN* model is 1.57% higher than one achieved by 450-D features. The results also show that additional introduced features in *Sub-FCNN* can make the accuracy of *D-CNNs* with well-trained parameters increase from 98.89% to 99.25%. These results demonstrate that demonstrated that the additionally introduced statistical features are able to both improve the accuracy of patch classification in the *Sub-FCNN* and the *D-CNNs* model.

**TABLE 3. Results of patch classification for CASIA 2.0 in validate set.**

| The Proposed the Network Model | Accuracy |
|---|---|
| Sub-FCNN (1050-D) | 96.27% |
| Sub-SCNN | 98.54% |
| D-CNNs | **99.25%** |

### E. DESIGN CHOICES OF IMAGE PRE-PROCESSING LAYER

An important one of several design choices is the design of the image pre-processing layer, which can suppress the image's content and extract prediction error features in our network. In order to evaluate the advantage of using image pre-processing, the proposed *Sub-SCNN* model is trained and evaluated using three different choices for the pre-processing layer: (1) using the proposed image pre-processing layer, (2) without using the image pre-processing layer and using a standard convolutional layer as the beginning of the *Sub-SCNN*, (3) replacing 14 filters in the image preprocessing layer with a fixed high-pass filter which is commonly employed in the forensics and steganalysis.

To assess the performance gains achieved by the image pre-processing layer, both the training and validate datasets described in subsection 4.4 are also used in this subsection. And we report the patch classification accuracy that *Sub-SCNN* achieves using each choice of the filter as its beginning in Table 4. Additionally, the RER (Relative Error Reduction) is also reported achieved by the image pre-processing layer instead of each alternative and its formula is shown as follows:

$$RER = (E_1 - E_2)/E_1 \tag{26}$$

**TABLE 4. Results of the Sub-SCNN with different setting in the image pre-processing layer.**

| Pre-processing operation | Accuracy | RER (w.r.t. ours) |
|---|---|---|
| Image pre-processing layer | 98.54% | — |
| W/out image pre-processing layer | 95.74% | 65.72% |
| High-pass filter | 97.13% | 49.12% |

where $E_1$ is the error achieved by the lower performing method and $E_2$ corresponds to the error achieved by the

higher performing method. From the Table 4, one can notice that when *Sub-SCNN* network is trained without the image pre-processing layer, its performance decreased by 2.8%. This corresponds to the RER of 65.72% achieved by using the image pre-processing layer. Additionally, using the image pre-processing layer achieves the accuracy of 1.41% higher than when a fixed high-pass filter is used. This corresponds to RER of 49.12% over a fixed high-pass filter. These results demonstrate the advantage of using the image pre-processing layer.

### F. ARCHITECTURE DESIGN CHOICE OF SUB-SCNN

There is no systematic way of determining the necessary depth in a CNN architecture and one has to empirically choose the appropriate number of convolutional layers, specific pooling functions and activation functions in the network. The overall performance of our proposed *D-CNNs* mainly depends on the design of the two different sub-networks: *Sub-SCNN* and *Sub-FCNN*.

Therefore, to assess the performance of our proposed *D-CNNs* for performing image manipulation detection and localization, a set of experiments and analysis are conducted. The phenomenon that for the *Sub-FCNN* with 1050-D feature vector as input, the changes in the number of 1-D convolutional layer, pooling function, and activation function will not have a great impact on the accuracy of the validate set. And as shown by the patch classification accuracy of two sub-networks on the validating set in Table 3, it can be found that the *Sub-SCNN* plays a more important role in the results of the proposed *D-CNNs* model.

In order to determine the optimal architecture of the *Sub-SCNN* model, three sets of experiments, namely (1) the choice of different number of convolutional layer, (2) the choice of the pooling layer, and (3) the choice of the activation function. To accomplish this, the fixed architecture of the *Sub-SCNN* is defined in Fig 1 with the red dotted line. Then the one architectural design choice is changed in each experiment such as the choice of the different number of the convolutional layer or pooling layer. And the same training and validating datasets that described in previous subsection are used to examine the influence of several different design choices and their results are shown respectively in the following four subsections.

#### 1) THE NUMBER OF CONVOLUTIONAL LAYER

Three experiments are conducted to investigate the impact of the number of convolutional layers on the performance of the *Sub-SCNN*. In the experiments, the optimal number of convolutional layers used in the *Sub-SCNN* is identified by letting the number of layers vary from 1 to 3 and the patch classification accuracy in the validate set achieved by our proposed *Sub-SCNN* under each scenario can be evaluated. The results of our experiments show that our choice of "Conv_3" which represents that the number of convolutional layers is 3 with 98.54% accuracy rate outperforms the other choices of layer numbers. More specifically, with "Conv_1"

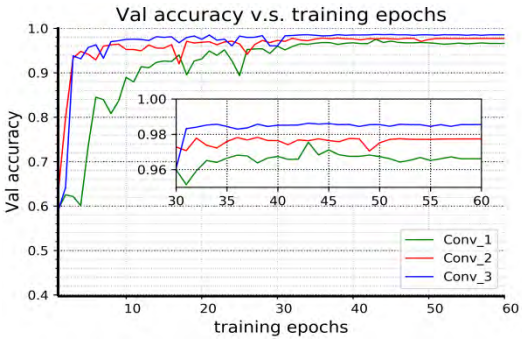



**FIGURE 8.** Sub-SCNN validating accuracy v.s. training epochs, blue: Conv_3 (the number of convolutional layers is equal to 3), red: Conv_2, green: Conv_1.

**TABLE 5.** Results of Sub-SCNN with different number of convolutional layers.

| #. Convolutional layer | Accuracy |
|---|---|
| Conv_1 | 96.61% |
| Conv_2 | 97.70% |
| Conv_3 | **98.54%** |

*Sub-SCNN* can achieve 96.61% accuracy and 97.70% accuracy when applying two convolutional layers. Fig 8 depicts the validating accuracy versus the training epochs curves for three choices about the number of selected convolutional layers. One can observe from this that "Conv_3" network converges slightly quicker to a higher accuracy.

#### 2) POOLING OPERATIONS

To evaluate the impact on our *Sub-SCNN*'s performance using different types of pooling layers, i.e., average-pooling, max-pooling and max-pooling with average pooling after the "Conv5" layer, three CNN models are trained using the architecture described in Fig 1 with the red dotted line. The $1 \times 1$ convolutional filters in the "Conv5" are applied to learn the association between feature maps. Because of this, the choice of pooling before the fully-connected layers that perform classification is very important. Table 6 summarizes best patch classification accuracy achieved by different choices of pooling function. And the choice of max-pooling with avg-pooling after "Conv5" layer maximizes the performance of *Sub-SCNN* model with an accuracy of 98.54% and outperforms the choice of average-pooling and max-pooling

**TABLE 6.** Results of Sub-SCNN with different pooling operations.

| Pooling operations | Accuracy |
|---|---|
| Avg-pooling | 95.22% |
| Max-pooling | 96.66% |
| Max-pooling w/ avg-pooling after Conv5 | **98.54%** |

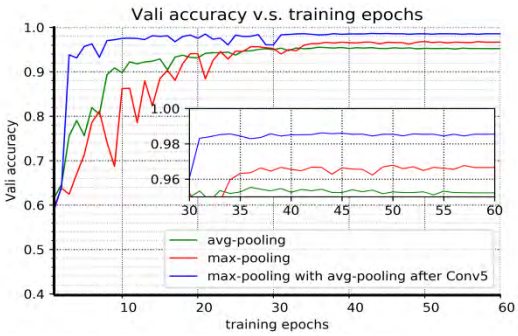



**FIGURE 9.** Sub-SCNN validating accuracy v.s. training epochs, blue: max-pooling with avg-pooling after the "Conv5" layer, red: max-pooling, green: avg-pooling.

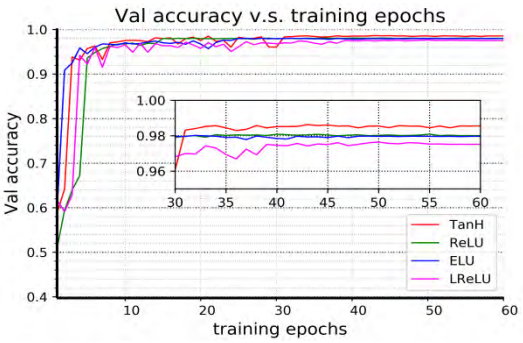



**FIGURE 10.** Sub-SCNN validating accuracy v.s. training epochs, red: TanH, green: ReLU, blue: ELU, magenta: LReLU.

by 3.32%, 1.88% respectively. From Fig.9, we can observe that proposed *Sub-SCNN* with the choice of max-pooling layers converges slower to its overall accuracy than the other two alternatives.

#### 3) ACTIVATION FUCTION

Finally, four kinds of activation functions, i.e., TanH, ReLU, ELU, and LReLU, are selected to evaluate the impact on our Sub-SCNN's performance. As shown in Table 7, the performance of Sub-SCNN model with TanH activation function is 0.54% better than the model with ReLU, 0.58% better than the model with ELU and 1.04% better than the model with LReLU. Fig.10 describes the validating accuracy versus the training accuracy curves for the four choices of the activation function. And we find that the TanH and ReLU networks converge slightly quicker to a higher accuracy and the TanH network achieves the best accuracy of 98.54%.

**TABLE 7.** Results of Sub-SCNN with different activation functions.

| Activation functions | Accuracy |
|---|---|
| TanH | **98.54%** |
| ReLU | 98.00% |
| ELU | 97.96% |
| LReLU | 97.50% |

### G. EXPERIMENTAL RESULTS AND COMPARISON

To comprehensively evaluate the localization performance of our trained *D-CNNs* model, a set of comparison experiments using the same testing images in CASIA v2.0, Columbia and Carvalho describe in Table 1. We first compare the proposed model with two state-of-the-art methods, the first of which using the CFA features to detect the forged region (denoted by the features they extracted as CFA) [37] and the second propose a new camera-based technique for tampering localization (denoted by the technique they utilized as CBT). And the implementation of the first algorithm is provided in an available Matlab toolbox by the author. The second method is re-implemented by strictly following their instructions in this paper [38]. The model denoted as D-CNNs-seg represents that the post-processing operation of graph-based image segmentation is selected on the smoothed tampering possibility map. However, the model denoted as D-CNNs-bi utilizes binary post-processing operation on the smoothed tampering possibility map and its output probability map is a binary image where the tampered regions (marked as 1) are defined as areas with values greater than 0.5 and pristine region (marked as 0) are defined as areas with values less than 0.5. Note that the probability map generated by the two comparison methods does not use post-processing operations but only performs simple binarization.

Table 8 shows a comparison between the proposed model and other methods testing images from the CASIA v2.0. From the results, we can find that the D-CNNs-seg robustly performs the best over all criteria. In the CASIA v2.0, it can be seen that the utilizing of post-processing increases the four evaluation indicators by 0.1, 0.11, 0.06 and 0.13. On average, our *F1* of the model D-CNNs-seg is 0.59, and 3.93 times, 3.28 times the baselines in CAF and CBT respectively. These results show the proposed model works well for image manipulation localization.

Then a set of experiments are conducted on Columbia and Carvalho to evaluate our trained *D-CNNs* model which is trained using training datasets from CASIA v2.0. In the experiments, the post-processing of graph-Based image segmentation and simple binarization are also used separately on the smoothed tampering possibility map generated by our *D-CNNs*. The average performance of our model with different post-processing operations on Columbia and Carvalho are shown in Table 9. One can notice that all the metrics of Proposed-bi model is better than the metrics of D-CNNs-seg. It is evident that our *D-CNNs* model with post-processing based on segment algorithm doesn't give satisfactory accuracy as well as with binary post-processing. However, these results also demonstrate that our proposed model is able to better locate the tampering regions, and achieve the *F1* value of 0.69 *F1* on Columbia and 0.58 *F1* on Carvalho.

To analyze the phenomenon caused by different post-processing operations, we select the two images on each of

**TABLE 8.** Experimental results of comparison on CASIA v2.0.

| Dataset | Models | Recall | Precision | Accuracy | F1 |
|---|---|---|---|---|---|
| CASIA V2.0 | D-CNNs-seg | 0.77 | 0.51 | 0.85 | 0.62 |
| | D-CNNs-bi | 0.67 | 0.40 | 0.79 | 0.49 |
| | CFA [37] | 0.12 | 0.15 | 0.70 | 0.15 |
| | CBT [38] | 0.28 | 0.17 | 0.61 | 0.18 |

**TABLE 9.** Experiment results of different post-processing operations on columbia and carvalho.

| Dataset | Models | Recall | Precision | Accuracy | F1 |
|---|---|---|---|---|---|
| Columbia | D-CNNs-bi | 0.81 | 0.61 | 0.86 | 0.69 |
| | D-CNNs-seg | 0.71 | 0.51 | 0.77 | 0.59 |
| Carvalho | D-CNNs-bi | 0.75 | 0.48 | 0.83 | 0.58 |
| | D-CNNs-seg | 0.57 | 0.30 | 0.65 | 0.39 |

**TABLE 10.** Experimental results of different post-processing operations on columbia and carvalho.

| | Image result | *Recall* | *Precision* | *Accuracy* | *F1* |
|---|---|---|---|---|---|
| Fig 11 (a1) | D-CNNs-seg (f1) | 0.70 | 0.88 | 0.96 | 0.78 |
| | D-CNNs-bi (g1) | 0.96 | 0.85 | 0.98 | 0.90 |
| Fig 11 (a2) | D-CNNs-seg (f2) | 0.59 | 0.88 | 0.94 | 0.71 |
| | D-CNNs-bi (g2) | 0.89 | 0.88 | 0.97 | 0.88 |
| Fig 11 (a3) | D-CNNs-seg (f3) | 0.63 | 0.50 | 0.75 | 0.56 |
| | D-CNNs-bi (g3) | 0.99 | 0.75 | 0.92 | 0.86 |
| Fig 11 (a4) | D-CNNs-seg (f4) | 0.15 | .0.42 | 0.75 | 0.22 |
| | D-CNNs-bi (g4) | 0.99 | 0.62 | 0.85 | 0.77 |

**TABLE 11.** Average $F1$ and $MCC$ scores of the comparison between the proposed and other algorithm on the columbia and carvalho datasets.

| | Columbia | | Carvalho | |
|---|---|---|---|---|
| Model | *F1* | *MCC* | *F1* | *MCC* |
| NOI3 | 0.45 | 0.21 | 0.27 | 0.11 |
| CFA[37] | 0.47 | 0.23 | 0.29 | 0.16 |
| ADQ1[9] | 0.50 | 0.27 | 0.29 | 0.15 |
| BLK[39] | 0.52 | 0.33 | 0.31 | 0.18 |
| NOI2[40] | 0.53 | 0.35 | 0.32 | 0.19 |
| E-MFCN[19] | 0.61 | 0.48 | 0.48 | 0.41 |
| D-CNNs | **0.69** | **0.49** | **0.58** | **0.43** |

datasets of Columbia and Carvalho are selected respectively, and their results are displayed in Table 10. In order to have an intuitive comparison, four examples are illustrated in Fig 11. An obvious advantage when using no post-processing based on image segmentation can be observed on Table 10. Compared to the post-processing operation, only binarizing the image will increase the accuracy of the model tampering region, and this phenomenon is more obvious on the Carvalho dataset.

The regions in initial tampering possibility map and smoothed tampering possibility map which are more likely to be tampered with are in white color. As is shown in Fig 11, from each result of two images in the Columbia dataset, we observe that: 1) our results with binary operation match

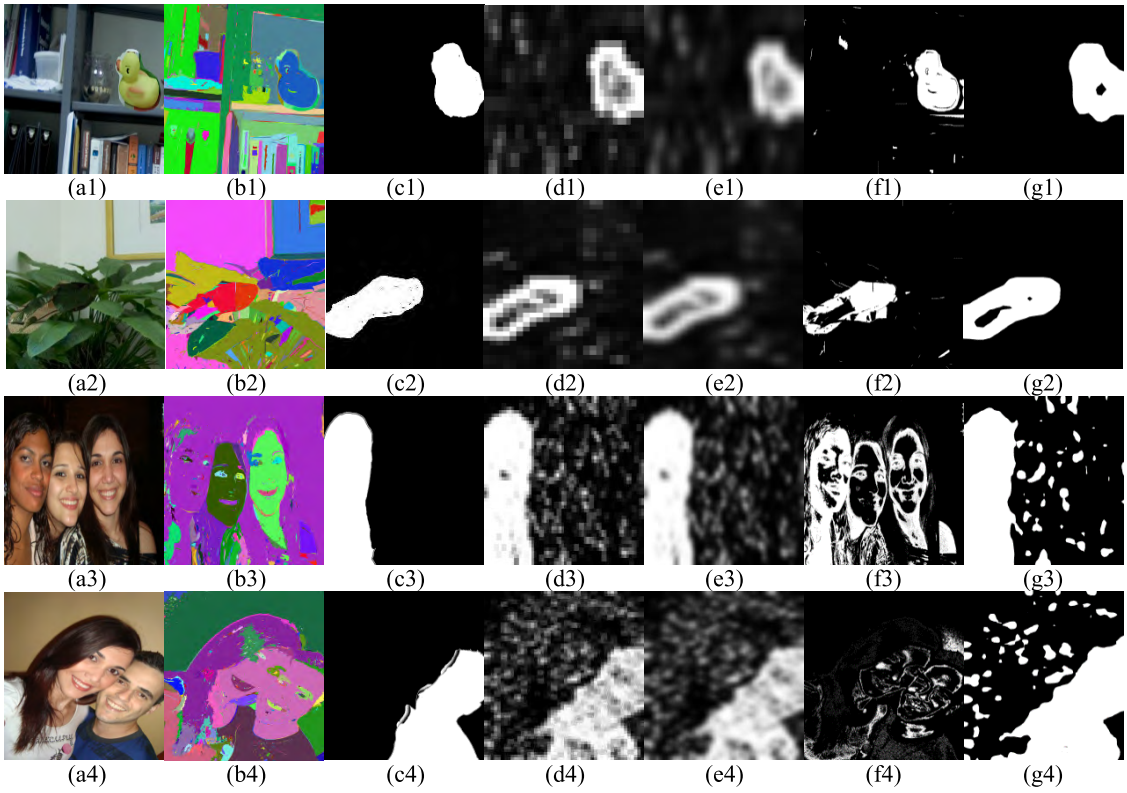



**FIGURE 11.** Output result of proposed model with two choice of post-processing on the Columbia and Carvalho datasets. Starting from the first column: 1: forged images used in experiments ((a1) and (a2) are selected from the Columbia. (a3) and (a4) are selected from the Carvalho.), 2: results of graph-Based image segmentation algorithm [32], 3: ground truth, 4: initial tampering possibility map, 5: smoothed tampering possibility map, 6: tampering possibility map with post-processing [32], 7: the final binary tampering possibility map.

the ground truth well. Using the post-processing operation based on tampering possibility map can actually match well with some objects in the input images. e.g., in example a1, the duck region is detected in image localization result f1; in example a2, the contour of tampering region is detected in result f2. 2) However, the empty areas that are not detected can't be well filled in result f2, and it is prone to include more false positives as we can see many other small white regions in f1 and f2. 3) Compared with the tampering regions in CASIA v2.0, the each of tampering region in Columbia is not a specific semantic object and is an area randomly divided from a donor image. And the segmentation algorithm divided the image tampering region into many small regions. Therefore, the regions which are not detected on the initial tampering probability map can't be filled by the post-segment processing based on segment, and the true positive is reduced.

It is known that the images in the Carvalho dataset are all images of people and each image has at least two peoples' heads. For each result of two images in the Carvalho dataset, we observe that: 1) the segmentation algorithm used in the post-processing operation clusters the black-skinned woman's cheek and image backgrounds in b3 to fail to separately identify them, and it is prone to include more false positives as we can see the contours of other pristine areas containing other persons in result f3 and f4. 2) as shown in b4, each person's cheek is not accurately segmented and the facial texture features of different people are clustered into a same segment by the segmentation algorithm. 3) our method with the binary operation is able to locate the tampering areas, although the results of g3 and g4 exist the phenomenon of misclassification.

Therefore, both theoretical analysis and experimental results show that binary post-processing is more suitable for the Columbia and Carvalho datasets.

Finally, the comparative experiments between the proposed *D-CNNs* model and some recent state-of-the-art image manipulation localization algorithms are also conducted on the Columbia and Carvalho datasets by training on CASIA v2.0. The methods proposed in [37, 9, 39, 40, 19] are denoted by the features they extracted as CFA, ADQ1, BLK, NOI2, and E-MFCN respectively. As noted in [19], the implementation of these existing algorithms is provided in a publicly available Matlab toolbox in [39]. And the listed comparison results are just as reported in [19]. For each method, the average *F1* and *MCC* scores across the dataset are computed, and the results are shown in Table 11. We can see that the *D-CNNs* model outperforms the benchmarking algorithms on all two datasets, in terms of both *F1* and *MCC* score.

## V. CONCLUSIONS AND FUTURE WORK

In this paper, we present a unified framework based on the convolutional neural network to locate the doctored regions at the pixel level. And in the proposed method, two sub-networks, referred to as *Sub-SCNN* and *Sub-FCNN*, are jointed to localize manipulated regions where a transfer policy is applied. We attempt to conduct a study on two post-processing operations which are selected to utilize on initial probability map output by the proposed *D-CNNs* model to finalize the pixel-wise manipulation region localization in the different image databases. The design principle of our *D-CNNs* model is elaborated and its rationality is systematically validated by running a number of experiments. The proposed method is evaluated on manipulated images from the CASIA v2.0, Columbia and Carvalho datasets. Experimental results suggest that the *Sub-SCNN* model can be used directly and, when combined with *Sub-FCNN* model, can lead to superior performance. The detail results show that our approach can efficiently locate the tampering regions and outperform existing image manipulation localization methods on these datasets, with the *D-CNNs* performing the best.

Our future work will focus on designing a more effective network architecture which is robustness against typical post-processing operations and searching for some high-level cues for better detecting image manipulation and locating the forged regions.

## REFERENCES

[1] B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," in *Proc. Int. Symp. Electron. Imag.*, 2017, pp. 77–86.
[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] W. Wang, J. Dong, and T. Tan, "Exploring DCT coefficient quantization effects for local tampering detection," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 10, pp. 1653–1666, Oct. 2014.
[4] D. Cozzolino, G. Poggi, and L. Verdoliva, "Splicebuster: A new blind image splicing detector," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Jan. 2016, pp. 1–6.
[5] E. Kee, J. F. O'Brien, and H. Farid, "Exposing photo manipulation with inconsistent shadows," *ACM Trans. Graph.*, vol. 32, no. 3, Jun. 2013, Art. no. 28.
[6] B. Mahdian and S. Saic, "Using noise inconsistencies for blind image forensics," *Image Vis. Comput.*, vol. 27, no. 10, pp. 1497–1503, Sep. 2009.
[7] S. Lyu, X. Pan, and X. Zhang, "Exposing region splicing forgeries with blind local noise estimation," *Int. J. Comput. Vis.*, vol. 110, no. 2, pp. 202–221, Nov. 2014.
[8] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of JPEG artifacts," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1003–1017, Jun. 2012.
[9] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis," *Pattern Recognit.*, vol. 42, no. 11, pp. 2492–2501, Nov. 2009.
[10] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: A survey," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 84–100, Jan. 2018.
[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.
[12] Y. Zhang, L. L. Win, J. Goh, and V. L. Thing, "Image region forgery detection: A deep learning approach," in *Proc. Singapore Cyber-Secur. Conf. (SG-CRC)*, 2016, pp. 1–11.
[13] Y. Rao and J. Ni, "A deep learning approach to detection of splicing and copy-move forgeries in images," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2017, pp. 1–6.
[14] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2016, pp. 5–10.
[15] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, Jun. 2012.
[16] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection through residual-based local descriptors and block-matching," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2015, pp. 5297–5301.
[17] B. Bayar and M. C. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2691–2706, Nov. 2018.
[18] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. S. Manjunath, "Exploiting spatial structure for localizing manipulated image regions," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 4970–4979.
[19] R. Salloum, Y. Ren, and C.-C. J. Kuo, "Image splicing localization using a multi-task fully convolutional network (MFCN)," *J. Vis. Commun. Image Represent.*, vol. 51, pp. 201–209, Feb. 2018.
[20] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[21] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. (Nov. 2015). "Fast and accurate deep network learning by exponential linear units (ELUs)." [Online]. Available: https://arxiv.org/abs/1511.07289
[22] B. Xu, N. Wang, T. Chen, and M. Li. (Nov. 2015). "Empirical evaluation of rectified activations in convolutional network." [Online]. Available: https://arxiv.org/abs/1505.00853
[23] S. Ioffe and C. Szegedy. (Mar. 2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167
[24] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. New York, NY, USA: Springer, 2012, pp. 9–48.
[25] Y. Liu, Q. Guan, X. Zhao, and Y. Cao, "Image forgery localization based on multi-scale convolutional neural networks," in *Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur.*, 2018, pp. 85–90.
[26] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, May 2016.
[27] C. Tu and T. D. Tran, "Context-based entropy coding of block transform coefficients for image compression," *IEEE Trans. Image Process.*, vol. 11, no. 11, pp. 1271–1283, Nov. 2002.
[28] Y. Zhang and V. L. L. Thing, "A semi-feature learning approach for tampered region localization across multi-format images," *Multimed. Tools Appl.*, vol. 77, pp. 25027–25052, Oct. 2018.
[29] H. Farid and S. Lyu, "Higher-order wavelet statistics and their application to digital forensics," in *Proc. Comput. Vis. Pattern Recognit. Workshop*, 2003, p. 94.
[30] C. Pun, X. Yuan, and X. Bi, "Image forgery detection using adaptive oversegmentation and feature point matching," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1705–1716, Aug. 2015.
[31] M. T. McCann, D. G. Mixon, M. C. Fickus, C. A. Castro, J. A. Ozolek, and J. Kovačević, "Images as occlusions of textures: A framework for segmentation," *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 2033–2046, May 2014.
[32] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004.
[33] J. Dong, W. Wang, and T. Tan, "CASIA image tampering detection evaluation database," in *Proc. IEEE China Summit Int. Conf. Signal Inf. Process.*, Jul. 2013, pp. 422–426.

**ZENAN SHI** was born in 1993. She received the M.E. degree in computing science from Jilin University, China, in 2017, where she is currently pursuing the Ph.D. degree with the College of Computer Science and Technology. Her research interests include multimedia forensics and pattern recognition, especially on image splicing detection and localization.

**XUANJING SHEN** received the B.E., M.E., and Ph.D. degree from the Harbin Institute of Technology, China. He became a Lecturer with the Harbin Institute of Technology and then joined the Jilin University of Technology (now merged in Jilin University) in 1992. He is currently a Professor and a Ph.D. student advisor with the College of Computer Science and Technology, Jilin University. He has wide research interests including medical image segmentation, multimedia forensics, optical-electronic hybrid system, intelligent measurement system, and video understand technology.

**HUI KANG** received the M.E. and Ph.D. degrees from Jilin University in 1996 and 2007, respectively. She is currently an Associate Professor with the College of Computer Science and Technology, Jilin University. Her research interests include image processing, information integration, and distributed computing.

**YINGDA LV** was born in 1983. She received the Ph.D. degree in computer science and technology from Jilin University in 2015. She is currently a Lecturer with the Center for Computer Fundamental Education, Jilin University. Her research interests include image processing, identification for image authenticity, and pattern recognition.