

Two-Stage Copy-Move Forgery Detection with Self Deep Matching and Proposal SuperGlue

Yaqi Liu, Chao Xia, Xiaobin Zhu, and Shengwei Xu

Abstract—Copy-move forgery detection identifies a tampered image by detecting pasted and source regions in the same image. In this paper, we propose a novel two-stage framework specially for copy-move forgery detection. The first stage is a backbone self deep matching network, and the second stage is named as Proposal SuperGlue. In the first stage, atrous convolution and skip matching are incorporated to enrich spatial information and leverage hierarchical features. Spatial attention is built on self-correlation to reinforce the ability to find appearance similar regions. In the second stage, Proposal SuperGlue is proposed to remove false-alarmed regions and remedy incomplete regions. Specifically, a proposal selection strategy is designed to enclose highly suspected regions based on proposal generation and backbone score maps. Then, pairwise matching is conducted among candidate proposals by deep learning based keypoint extraction and matching, i.e., SuperPoint and SuperGlue. Integrated score map generation and refinement methods are designed to integrate results of both stages and obtain optimized results. Our two-stage framework unifies end-to-end deep matching and keypoint matching by obtaining highly suspected proposals, and opens a new gate for deep learning research in copy-move forgery detection. Experiments on publicly available datasets demonstrate the effectiveness of our two-stage framework.

Index Terms—Image forensics, two-stage copy-move forgery detection, self deep matching, Proposal SuperGlue.

1 INTRODUCTION

INCREASING availability and sophistication of digital image editing tools cause a major problem of that we even can not believe what we see [1]. Image forgery is becoming a global epidemic which deeply affects our daily life for that some forgers use elaborately forged images to spread fake news or do other unscrupulous businesses [2]. Copy-move forgery is a kind of image forgery in which one or several regions are pasted elsewhere in the same image in order to hide or duplicate objects of interest. Copy-move forgery detection techniques have always been a hot topic in image forensics [3], [4], [5], [6], and play important roles in cybersecurity and multimedia security [7], [8].

Conventional copy-move forgery detection methods adopt handcrafted features, and can be broadly divided into two categories, i.e., block-based approaches [6], [9], [10], [11], [12], [13], [14], and keypoint-based approaches [5], [15], [16], [17], [18], [19]. Their major difference is that block-based methods aim at exploring local features from abundant overlapping patches, while keypoint-based methods concentrate on patches of keypoints [20]. Nowadays, deep learning techniques have dominated various image processing tasks including image forensics [1], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. Deep learning based copy-move forgery detection has also been investigated in [2], [31]. In [31], Wu et al. proposed an end-to-end deep neural network for predicting copy-move forgery masks. They construct a convolutional neural network for feature extraction, then compute self-correlation maps of convolutional

features, and finally reconstruct forgery masks through a deconvolutional network. In [2], Wu et al. extended their network to a two-branch architecture: one branch localizes potential manipulation regions via visual inconsistencies; the other branch detects copy-move regions via visual similarities. According to the observations in [1], [26], [32], it is a very challenging task to localize forged regions in realistic forged images which barely have visual inconsistencies. As for the branch for detecting visual similarities, it tries to explore high-level low-resolution convolutional features [33], limiting the ability to detect accurate boundaries and small forged regions. Hence, we focus on digging deeper into visual similarity clues in our work.

In this paper, we propose a novel two-stage copy-move forgery detection framework which integrates end-to-end deep matching with proposal based keypoint matching. The pipeline of this two-stage framework is shown in Fig. 1.

In the first stage, a backbone self deep matching network is constructed to generate backbone score maps which indicate suspicious probabilities of pixels. Our backbone network integrates atrous convolution, skip matching, and spatial attention. Atrous convolution can increase the resolution of feature maps, and skip matching can investigate hierarchical information. Particularly, we discover the inherent connections between spatial attention and self-correlation, and propose a self-correlation module with spatial attention. Previous copy-move forgery detection methods [2], [31] only adopt VGG16 [33], we further study the feasibility of constructing deep matching based on deeper networks (ResNet50, ResNet101 [34]) and light-weight networks (MobileNet [35], [36], [37], ShuffleNet [38], [39]). The backbone network is regarded as a filter to efficiently detect suspected forged regions, while the results may inevitably contain false-alarmed regions or incomplete regions. Thus, we propose the second stage to remove false-alarmed regions and

- Y. Liu, C. Xia and S. Xu are with Beijing Electronic Science and Technology Institute, Beijing 100070, China.
E-mail: liuyaqi@besti.edu.cn, xiachao@besti.edu.cn
- X. Zhu is with the Department of Computer Science and Technology, School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.

(Corresponding author: Chao Xia.)

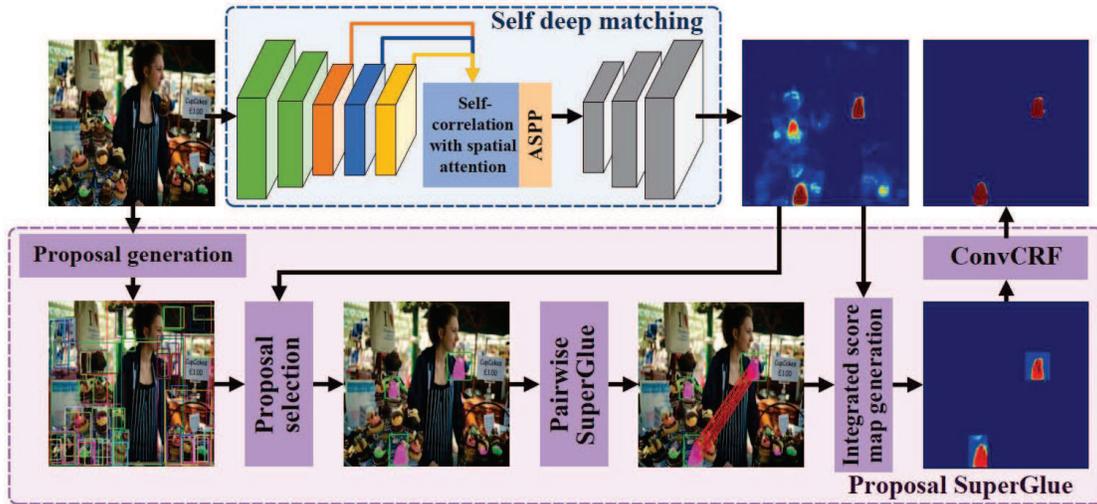


Fig. 1. Overview of our two-stage copy-move forgery detection with self deep matching and Proposal SuperGlue. The first stage is a backbone self deep matching network based on atrous convolution, skip matching, and self-correlation with spatial attention. Red, blue, yellow blocks denote three convolutional blocks with the same scale, which are re-constructed by atrous convolution and skip matching. The second stage is named as Proposal SuperGlue. Proposal selection is conducted based on generated proposals and the backbone score map. Pairwise SuperGlue is conducted among candidate highly suspected proposals. Then, integrated score maps are generated by integrating matched keypoint scores and backbone scores. ConvCRF is constructed to refine integrated score maps.

remedy incomplete regions.

In the second stage, a proposal based keypoint matching method is proposed and named as Proposal SuperGlue. Proposal SuperGlue mainly consists of two components: (1) A proposal selection module obtains several highly suspected proposals from a large number of bounding boxes provided by a proposal generation method [40]. Proposal selection takes advantage of both backbone score maps and appearance clues, bridging the gap between deep matching and keypoint matching. (2) Proposal matching and label generation are devised to remove false alarms, remedy incomplete regions, and generate pixel labels from score maps. Deep-learning keypoint extraction (SuperPoint [41]) and matching (SuperGlue [42]) are conducted among candidate proposals. An integrated score map generation method is designed to integrate keypoint matching results and backbone score maps. And an integrated score map refinement method is presented based on an improved fully connected CRF, i.e., ConvCRF (Convolutional Conditional Random Field) [43].

Specifically, our main contributions of this paper can be summarized as follows:

- An innovative two-stage copy-move forgery detection framework is proposed based on self deep matching and Proposal SuperGlue. We imaginatively integrate end-to-end deep matching with keypoint matching through highly suspected proposals.
- A backbone self deep matching network is constructed based on atrous convolution, skip matching and spatial attention. Inherent connections between self-correlation and spatial attention are elaborately investigated.
- Proposal SuperGlue, which incorporates proposal generation and deep-learning keypoint matching with a series of postprocessing procedures, is proposed to effectively remove false-alarmed regions

and remedy incomplete regions.

The structure of this paper is as follows: In Section 2, we discuss related work. In Section 3, we elaborate the proposed framework. In Section 4, experiments are conducted. In Section 5, we draw conclusions.

2 RELATED WORK

In this section, we briefly review the state-of-the-art copy-move forgery detection methods, attention mechanism, proposal generation and local feature matching which are the key techniques researched in our work.

Copy-move forgery detection. Conventional copy-move forgery detection methods mainly consist of three components [6]: (1) feature extraction: extracting suitable features from pixels of interest; (2) matching: computing their best matching based on their associated features; (3) post-processing: processing and filtering vague detections to reduce false alarms. According to the formulations of feature extraction and subsequent matching schemes, these methods can be classified into two categories, i.e., block-based and keypoint-based methods. In block-based methods, a variety of features have been investigated for describing overlapping blocks and dense matching, e.g., DCT (Discrete Cosine Transform) [13], DWT (Discrete Wavelet Transform) and KPCA (Kernel Principal Component Analysis) [9], Zernike moments [10], PCT (Polar Cosine Transform) [11], [44], PCET (Polar Complex Exponential Transform) [14], LBP (Local Binary Patterns) [12], Circular Harmonic Transforms (CHT) [6]. In keypoint-based methods, the commonly used features are SIFT (Scale Invariant Feature Transform) [5], [15], [16], [18] and SURF (Speeded-Up Robust Features) [19], [45]. Although great progress has been made in the study of copy-move forgery detection, it is still an unresolved challenging task for that duplicate regions

may be small or smooth, and have gone through complicated rotation, resizing, compression and noise addition [14]. Besides, all the above conventional copy-move forgery detection methods rely on hand-crafted features and each module is optimized independently [2]. Consequently, two kinds of end-to-end deep learning based copy-move forgery detection methods were proposed by Wu et al. in [2], [31].

Attention mechanism. In [46], Sutskever et al. constructed a multi-layer long short term memory (LSTM) to map the input sequence to a fixed-length vector, and another deep LSTM to decode the target sequence from the vector. In [47], Bahdanau et al. adopted the attention mechanism to dynamically generate the vectors. Since then, the attention mechanism has been widely applied to solve sequential decision tasks [48], and numerous attention-based models have been proposed [49], [50], [51]. The attention mechanism can bias the allocation of available processing resources to the most informative components of input signals [52], and has also been applied to solve multimedia problems, e.g., image classification [52], [53], object detection [54], image super-resolution [55], video classification [56]. In these tasks, consistent improvements have been gained by adopting attention mechanisms to recalibrate informative convolutional features.

Proposal generation. In our work, we try to generate bounding boxes enclosing suspected regions. Proposal generation is a kind of technique that has been widely researched before the arrival of end-to-end object detection [57]. It aims to find out a set of (ranging from hundreds to thousands per image) proposal regions or bounding boxes which may contain objects [58]. Since we can get hundreds of proposals which are near to contours in images, they cover suspected regions with high probability. Proposal generation approaches can be divided into two categories: conventional methods and deep-learning methods. Conventional methods leverage low-level grouping and saliency clues, e.g., objectness scoring [59], [60], seed segmentation [61], [62], [63], superpixel merging [64], [65]. Deep-learning approaches construct deep-network architectures to obtain proposals. For example, Deepbox [66] learns a convolutional network to rerank proposals generated by EdgeBox [60]. Multibox [67] constructs a deep network to generate bounding box proposals. DeepMask [40] and SharpMask [68] can generate and refine segmentation proposals with high efficiency. These deep-learning approaches give us an opportunity to efficiently generate suspected boxes enclosing forged regions from a small set of candidate proposals.

Local feature matching. It mainly consists of five steps, (1) detecting interest points, (2) computing visual descriptors, (3) matching visual descriptors with a nearest neighbor (NN) search, (4) filtering incorrect matches, (5) estimating a geometric transformation [42]. In recent years, researchers have been trying to learn better sparse detectors and local descriptors [41], [69], [70], [71], [72] from data using Convolutional Neural Networks (CNNs), and attempting to improve their discriminative ability by using various strategies, e.g., a wider context using regional features, log-polar patches, unsupervised learning. Although tremendous progress has been made in this field, these sets of matches are still estimated by NN search. In [42], a novel approach based on graph neural networks, i.e., SuperGlue,

is proposed to establish pointwise correspondences from off-the-shelf local features: it acts as a middle-end between hand-crafted or learned front-end and back-end. SuperGlue outperforms other learned approaches and achieves state-of-the-art results on pose estimation. In keypoint-based copy-move forgery detection approaches [5], [15], [16], [17], [18], [19], hand-crafted local features and NN search have been widely researched. In our work, we try to integrate learning based detector, descriptor and matching into a unified copy-move forgery detection framework.

3 METHODOLOGY

Our two-stage copy-move forgery detection framework consists of self deep matching and Proposal SuperGlue, as shown in Fig. 1. The first stage is a backbone self deep matching network, which generates score maps in an end-to-end manner. Firstly, we introduce the main architecture of our backbone network, including several alternative formulations in section 3.1.1. Then, we introduce self-correlation with spatial attention in section 3.1.2. The second stage is called Proposal SuperGlue, which is proposed to remove false-alarmed regions and remedy incomplete regions. In section 3.2.1, we introduce our proposal selection strategy based on deep-learning proposal generation. In section 3.2.2, we introduce proposal-based point matching, integrated score map generation and refinement.

3.1 Self Deep Matching

3.1.1 Backbone Network Architecture

In our work, we adopt VGG16 as our basic feature extractor, remove pooling operators in the fourth and fifth convolutional blocks [33], and adjust the fifth block by adopting atrous convolution to keep their original field-of-views. Atrous convolution can generalize standard convolution, adjust filter’s field-of-view and control the resolution of convolutional features [73], [74], [75]. Let $\mathbf{y}(i_c, j_c)$ denote the output of the atrous convolution of a 2-D input signal $\mathbf{x}(i_c, j_c)$, and the atrous convolution can be computed as:

$$\mathbf{y}(i_c, j_c) = \sum_{k_1, k_2} \mathbf{w}(k_1, k_2) \times \mathbf{x}(i_c + r_{ac}k_1, j_c + r_{ac}k_2) \quad (1)$$

where $k_1, k_2 \in [-fl(\frac{K}{2}), fl(\frac{K}{2})]$ ($fl(\cdot)$ is a floor function), $\mathbf{w}(k_1, k_2)$ denotes a $K \times K$ filter, atrous rate r_{ac} determines the stride with which we sample the input signal. In the fifth block of our basic architecture, atrous rate r_{ac} is set to 2. Consequently, we generate three groups of larger feature maps with the same size, i.e., \mathbf{F}_3 , \mathbf{F}_4 and \mathbf{F}_5 in Fig. 2. These hierarchical feature maps are all fed into our self-correlation module with spatial attention to compute the correlation maps. This kind of skip connections between multi-level feature maps and correlation computation is named as skip matching, and can effectively leverage rich hierarchical information provided by the feature extractor. And in the next section, we will introduce our self-correlation with spatial attention in detail.

Based on the computed correlation maps, we construct an Atrous Spatial Pyramid Pooling (ASPP) module to capture their multiscale information. As shown in Fig. 2, we construct 3 parallel atrous convolutional layers with 3×3

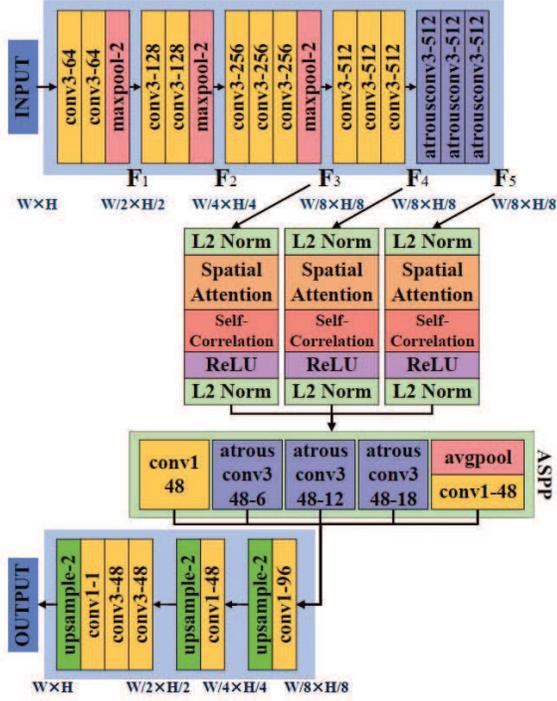


Fig. 2. The architecture of backbone network with VGG16.

filters and atrous rates of $\{6, 12, 18\}$. Besides, we construct a convolutional layer with 1×1 filters, and a global average pooling layer followed by a convolutional layer with 1×1 filters to capture local features and image-level features respectively. All these convolutional layers output 48-channel feature maps, and the five groups of feature maps are concatenated and fed into subsequent layers which are constituted of convolutional and upsampling layers.

Alternative formulations. Previous end-to-end copy-move forgery detection approaches [2], [31] adopt VGG16 for feature extraction, and we do not know the performance of deeper or light-weight networks. Thus, we formulate the popular ResNet50 and ResNet101 [34] as deeper feature extractor, decrease the strides of fourth and fifth convolutional blocks, and set the atrous rates as 2 and 4 respectively. Light-weight networks are specifically tailored for mobile and resource constrained environments [36]. We reformulate three popular and competitive light-weight networks, i.e., MobileNetV2 [36], MobileNetV3 [37] and ShuffleNetV2 [39]. We enlarge feature maps of the last two convolutional blocks by decreasing strides and adopting atrous convolution. In these formulations, we still can get 3 sets of feature maps with the same size. And these features are fed into correlation layers and subsequent score map generation layers.

3.1.2 Self-Correlation with Spatial Attention

In this section, we detailedly introduce the proposed self-correlation with spatial attention, and discuss the connections between self-correlation and spatial attention. Let \mathbf{F}_l denote the l -th block feature maps, and $\mathbf{F}_l(i, j)$ denotes a c -dimensional descriptor at (i, j) . Note that $\mathbf{F}_l \in \mathbb{R}^{h \times w \times c}$, $i \in [1, h]$, $j \in [1, w]$, h and w indicate the height and width of the feature map, and $h = w$ in our work. Before the attention and correlation computation, L2-normalization is con-

ducted, $\bar{\mathbf{F}}_l(i, j) = \text{L2_norm}(\mathbf{F}_l(i, j)) = \mathbf{F}_l(i, j) / \|\mathbf{F}_l(i, j)\|_2$. By adopting L2-normalization, we can restrict the value ranges of descriptors, and obtain normalized feature maps, i.e., $\bar{\mathbf{F}}_l$.

Spatial attention is a kind of self-attention module which calculates response at a position as a weighted sum of the features at all positions [51]. It can capture long-range dependences and allocate attention according to similarity of color and texture. In our work, spatial attention is constructed to reinforce $\bar{\mathbf{F}}_l$. $\bar{\mathbf{F}}_l$ is first transformed into two feature spaces $\mathbf{f}(\bar{\mathbf{F}}_l) = \bar{\mathbf{F}}_l \mathbf{W}_{l,f} + \mathbf{b}_{l,f}$ and $\mathbf{g}(\bar{\mathbf{F}}_l) = \bar{\mathbf{F}}_l \mathbf{W}_{l,g} + \mathbf{b}_{l,g}$. Then we compute:

$$\beta_l^{(m,n)} = \frac{\exp(s_l^{(m,n)})}{\sum_n \exp(s_l^{(m,n)})} \quad (2)$$

where

$$s_l^{(m,n)} = \mathbf{f}(\bar{\mathbf{F}}_l^{(m)})^T \mathbf{g}(\bar{\mathbf{F}}_l^{(n)}) \quad (3)$$

$\beta_l^{(m,n)}$ indicates the extent to which the model attends to the n -th location when predicting the m -th region, $m, n \in [1, h \times w]$. The output of the attention block is computed as:

$$\mathbf{o}_l^{(m)} = \sum_n \beta_{mn} \mathbf{h}(\bar{\mathbf{F}}_l^{(n)}) \quad (4)$$

where $\mathbf{h}(\bar{\mathbf{F}}_l) = \bar{\mathbf{F}}_l \mathbf{W}_{l,h} + \mathbf{b}_{l,h}$. Note that $\mathbf{W}_{l,f} \in \mathbb{R}^{c \times \frac{c}{s}}$, $\mathbf{W}_{l,g} \in \mathbb{R}^{c \times \frac{c}{s}}$, $\mathbf{W}_{l,h} \in \mathbb{R}^{c \times c}$, $\mathbf{b}_{l,f} \in \mathbb{R}^{\frac{c}{s}}$, $\mathbf{b}_{g,h} \in \mathbb{R}^{\frac{c}{s}}$, $\mathbf{b}_{l,h} \in \mathbb{R}^c$, which are implemented as 1×1 convolutional layers. $\mathbf{O}_l = \{\mathbf{o}_l^{(1)}, \mathbf{o}_l^{(2)}, \dots, \mathbf{o}_l^{(h \times w)}\}$ are the attention values for $\bar{\mathbf{F}}_l$. Consequently, the spatial attention reinforced convolutional feature maps can be computed as:

$$\tilde{\mathbf{F}}_l = \text{Atten}_l(\bar{\mathbf{F}}_l) = \lambda_l \mathbf{O}_l + \bar{\mathbf{F}}_l \quad (5)$$

where λ_l is a scale parameter which is initialized as 0 and gradually learned to assign a proper value.

Self-correlation aims to compute the similarity between every two locations in the convolutional feature maps. Scalar product is commonly used:

$$c_l^{(m,n)} = (\tilde{\mathbf{F}}_l^{(m)})^T \tilde{\mathbf{F}}_l^{(n)} \quad (6)$$

Thus, we can get a raw correlation map tensor $\mathbf{C}_l = \{c_l^{(m,n)} | m, n \in [1, h \times w]\} \in \mathbb{R}^{h \times w \times (h \times w)}$. In fact, only a small fraction of features has close relations, and the majority of features are dissimilar. This indicates that a subset of \mathbf{C}_l contains sufficient information to decide which feature is matched. Consequently, \mathbf{C}_l is sorted along the $(h \times w)$ channels, and top- T values are selected:

$$\tilde{\mathbf{C}}_l(i, j, 1:T) = \text{Top_T}(\text{Sort}(\mathbf{C}_l(i, j, :))) \quad (7)$$

A monotonic decreasing curve with an abrupt drop at some point should be observed along the T channels, as long as $\tilde{\mathbf{C}}_l(i, j)$ has matched regions. Thus, the T channels should cover the most drops, and the selection of T is discussed in experiments. In theory, our network can process arbitrary-sized images since the adoption of the top- T selection, unless $h \times w < T$. Moreover, zero-out and normalization operations are conducted on $\tilde{\mathbf{C}}_l$ to limit correlation values to certain ranges and filter redundant values:

$$\bar{\mathbf{C}}_l = \text{L2_norm}(\text{Max}(\tilde{\mathbf{C}}_l, 0)) \quad (8)$$

Since we get three groups of feature maps with the same size from feature extractor, i.e., $l \in \{3, 4, 5\}$, we can get three groups of correlation maps, i.e., $\hat{\mathbf{C}}_3, \hat{\mathbf{C}}_4$ and $\hat{\mathbf{C}}_5$. Note that the parameters of spatial attention are not shared for the computation of these three groups of correlation maps. Then, we concatenate the three groups of correlation maps, and get a correlation map tensor $\hat{\mathbf{C}} = \text{Concat}(\hat{\mathbf{C}}_3, \hat{\mathbf{C}}_4, \hat{\mathbf{C}}_5)$, where $\hat{\mathbf{C}} \in \mathbb{R}^{h \times w \times 3T}$. Since $\hat{\mathbf{C}}$ is computed from three groups of hierarchical feature maps, it contains rich correlation relations from coarse to fine.

Inherent connections between self-correlation and spatial attention. Both self-correlation and spatial attention attempt to explore correlations between every pair of features in a feature map tensor. Spatial attention conducts a scalar product in transformed spaces as Eq. (3), while self-correlation conducts a scalar product directly on feature maps as Eq. (6). Essentially, they have the same target of finding the close related regions. Spatial attention can allocate attention according to similarity of features, driving correlated regions to have closer feature distributions. Inspired by this inherent connection, we construct spatial attention before self-correlation computation. Consequently, it can reinforce the subsequent self-correlation computation.

Additionally, we have also investigated multi-head spatial attention. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions [51]. In fact, it constructs several parallel spatial attention blocks. Furthermore, we also attempt to add channel attention (SE blocks [52]) before or after spatial attention. Channel attention can highlight channel-wise informative features, and a weighted scalar product can be conducted in the correlation computation procedure. However, they can not achieve better performance with additional parameters which will be discussed in experiments.

3.2 Proposal SuperGlue

Proposal SuperGlue can be broadly divided into two steps. *In the first step*, a proposal selection method is proposed to obtain highly suspected regions from hundreds of bounding-box proposals. These bounding-box proposals are generated by a proposal generation method which exploits image appearance features to find bounding boxes near to contours. *In the second step*, we devise proposal-based keypoint matching with elaborately designed postprocessing procedures. Firstly, pairwise deep-learning keypoint matching is conducted among candidate proposals. Then, we propose an integrated score map generation method to integrate both self deep matching and keypoint matching results, so that some false-alarmed regions can be removed and incomplete regions can be complemented. Finally, in order to get good score distributions and accurate boundaries, ConvCRF is constructed to refine integrated score maps according to integrated scores and associated appearance similarity in the image. *The first step* is introduced in section 3.2.1, and *the second step* is discussed in section 3.2.2.

3.2.1 Proposal Selection

Processed by our backbone self deep matching network, we can get a score map $\mathbf{S} \in \mathbb{R}^{h_I \times w_I}$, h_I and w_I denote the

size of the score map which is the same as the size of input image \mathbf{I} . $\mathbf{S}(i_I, j_I) \in [0, 1]$, and $i_I \in [1, h_I], j_I \in [1, w_I]$. Since the feature maps of the backbone network have lower resolutions than original images, and the self-correlation is built on primitive scalar product instead of intricate similarity computing, \mathbf{S} may have false-alarmed or incomplete regions. In order to get rid of false-alarmed regions and complement incomplete regions, a proposal selection strategy is proposed to obtain highly suspected boxes for further matching.

Our motivation is that \mathbf{S} always has some isolated meaningless regions in some complicated images according to our observation. Whether we can enclose meaningful regions while ignore meaningless regions may affect the performance. However, how can we obtain several well enclosed bounding boxes based on score map \mathbf{S} and input image \mathbf{I} ? After all, there are too many bounding boxes can be generated from a single image, e.g., different scales, aspect ratios and positions. In fact, the majority of copy-move forged regions have clear contours, and might be possible to be covered by generated proposals which are relied on edges or saliency features [58]. Thus, we propose to conduct proposal generation on the input image, and select several high-quality boxes from hundreds of proposals. Our proposal selection strategy relies on score map \mathbf{S} , and consists of selecting and merging operations. We assume that there is a proposal generation function $\mathcal{P}(\cdot)$ with image \mathbf{I} as input, we can get P proposals $\mathbf{P} = \{\mathbf{p}_p | p \in [1, P]\}$. $\mathbf{p}_p = \{(x_1^p, y_1^p), (x_2^p, y_2^p)\}$ contains the coordinates of top left and bottom right corners. With score map \mathbf{S} at hand, we can get the average score in proposal \mathbf{p}_p , i.e., $s_p = f_{\text{avg}}(\mathbf{S}, \mathbf{p}_p)$. According to s_p and its relations with other proposals, we can obtain the final proposals. Our proposal selection strategy can be summarized as Algorithm 1.

In Algorithm 1, there are some basic functions: $\text{len}(\cdot)$ returns the item number of input list, $\text{IoU}(\cdot, \cdot)$ computes the Intersection over Union (IoU) [58] of two boxes, $\text{Inter}(\cdot, \cdot)$ computes their intersection, and $\text{Size}(\cdot)$ indicates the size of the input box. $\text{Merge}(\cdot, \cdot)$ is used to merge two input boxes, in other words, it generates the smallest box which can cover the two input boxes. The proposal generation function $\mathcal{P}(\cdot)$ is implemented based on DeepMask [40]. The basic idea of Algorithm 1 is that we try to reject proposals with small average scores, select proposals with higher scores from proposals which have high IoU with each other, and merge proposals or select larger boxes when they have large intersection rates. And there are some parameters need to set, proposal threshold score $s_t = 0.4$, threshold IoU $iou_t = 0.5$, threshold intersection rate $inter_t = 0.8$. Besides, all proposals or merged boxes should meet the basic requirement that they should be smaller than the half of the input image. Last but not least, iterations are conducted to avoid merged boxes with large intersection rates. By using Algorithm 1, we can get $\tilde{\mathbf{P}}$ (generally less than 10) high-quality proposals $\mathbf{P}_s = \{\mathbf{p}_{\tilde{p}} | \tilde{p} \in [1, \tilde{P}]\}$ ($\mathbf{P}_s(\tilde{p})$ indicates $\mathbf{p}_{\tilde{p}}$ in \mathbf{P}_s).

3.2.2 Keypoint Matching and Label Generation

Proposal-based keypoint matching. With high-quality proposals \mathbf{P}_s at hand, we can extract interest points from them and conduct keypoint matching. As we discussed

Algorithm 1 Proposal selection strategy.

Input: Image \mathbf{I} and score map \mathbf{S}
Output: Selected proposals \mathbf{P}_s

- 1: $\mathbf{P} = \mathcal{P}(\mathbf{I});$
- 2: $\mathbf{P}_t = \{ \};$
- 3: **for** $p = 1$ to P **do**
- 4: $s_p = f_{\text{avgs}}(\mathbf{S}, \mathbf{p}_p);$
- 5: **if** $s_p > s_t$ **then**
- 6: $flag = 1;$
- 7: **for** $i_{ps} = 1$ to $\text{len}(\mathbf{P}_t)$ **do**
- 8: $v_{iou} = \text{IoU}(\mathbf{P}_t(i_{ps}), \mathbf{p}_p);$
- 9: $v_{inter} = \text{Inter}(\mathbf{P}_t(i_{ps}), \mathbf{p}_p);$
- 10: **if** $v_{iou} > iou_t$ **then**
- 11: **if** $s_p > f_{\text{avgs}}(\mathbf{S}, \mathbf{P}_t(i_{ps}))$ **then**
- 12: $\mathbf{P}_t(i_{ps}) = \mathbf{p}_p; flag = 0; \text{Break};$
- 13: **if** $v_{inter}/\text{Size}(\mathbf{p}_p) > inter_t$ or
- 14: $v_{inter}/\text{Size}(\mathbf{P}_t(i_{ps})) > inter_t$ **then**
- 15: $\mathbf{p}_m = \text{Merge}(\mathbf{p}_p, \mathbf{P}_t(i_{ps}));$
- 16: **if** $f_{\text{avgs}}(\mathbf{S}, \mathbf{p}_m) > s_t$ **then**
- 17: $\mathbf{P}_t(i_{ps}) = \mathbf{p}_m; flag = 0; \text{Break};$
- 18: **if** $flag = 1$ **then**
- 19: $\mathbf{P}_t = \mathbf{P}_t \cup \mathbf{p}_p;$
- 20: $\mathbf{P}_s = \{ \};$
- 21: **while** $\text{len}(\mathbf{P}_s) \neq \text{len}(\mathbf{P}_t)$ **do**
- 22: **if** $\mathbf{P}_s \neq \emptyset$ **then**
- 23: $\mathbf{P}_t = \mathbf{P}_s;$
- 24: $\mathbf{P}_s = \{ \};$
- 25: **for** $i_{ps1} = 1$ to $\text{len}(\mathbf{P}_t)$ **do**
- 26: **for** $i_{ps2} = i_{ps1} + 1$ to $\text{len}(\mathbf{P}_t)$ **do**
- 27: $v_{inter} = \text{Inter}(\mathbf{P}_t(i_{ps1}), \mathbf{P}_t(i_{ps2}));$
- 28: **if** $v_{inter}/\text{Size}(\mathbf{P}_t(i_{ps1})) > inter_t$ or
- 29: $v_{inter}/\text{Size}(\mathbf{P}_t(i_{ps2})) > inter_t$ **then**
- 30: $\mathbf{p}_m = \text{Merge}(\mathbf{P}_t(i_{ps1}), \mathbf{P}_t(i_{ps2}));$
- 31: **if** $f_{\text{avgs}}(\mathbf{S}, \mathbf{p}_m) > s_t$ **then**
- 32: $\mathbf{P}_s = \mathbf{P}_s \cup \mathbf{p}_m;$
- 33: **else**
- 34: Insert $\mathbf{P}_t(i_{ps1})$ or $\mathbf{P}_t(i_{ps2})$
- 35: with higher intersection rate;

in Section 2, CNN-based interest point detection and description show a good prospect in numerous applications. Thus, we extract keypoints with corresponding descriptors from each proposal using SuperPoint [41]. It is a fully-convolutional model which operates on full-sized images and jointly computes pixel-level interest point locations and associated descriptors in one forward pass. It can be denoted as $\mathbf{K}_{\tilde{p}} = \text{SuperPoint}(\mathbf{p}_{\tilde{p}}, \mathbf{I})$, where $\mathbf{K}_{\tilde{p}}$ denotes the extracted keypoints set from proposal $\mathbf{p}_{\tilde{p}}$, and the k_p -th elements is $\mathbf{K}_{\tilde{p}}(k_p) = \{(x_{k_p}, y_{k_p}), \mathbf{d}_{k_p}\}$, \mathbf{d}_{k_p} denotes the corresponding descriptor. Then for each pair of point sets $\mathbf{K}_{\tilde{p}}(k_{p1})$ and $\mathbf{K}_{\tilde{p}}(k_{p2})$, we conduct SuperGlue [42] to get matched points and corresponding matching scores $\mathbf{M}_{\tilde{p}1}, \mathbf{M}_{\tilde{p}2} = \text{SuperGlue}(\mathbf{K}_{\tilde{p}1}, \mathbf{K}_{\tilde{p}2})$. SuperGlue uses a graph neural network and attention to solve an assignment optimization problem. Instead of learning better task-agnostic local features followed by simple matching heuristics and tricks, SuperGlue learns the matching process from pre-existing local features using a novel neural architecture for the first

time. It matches two sets of local features by jointly finding correspondences and rejecting non-matchable points. Finally, we can get M matched points and their matching scores $\mathbf{M} = \{\mathbf{M}_{\tilde{p}} | \tilde{p} \in [1, \tilde{P}]\} = \{(x_m, y_m), s_m | m \in [1, M]\}$.

Integrated score map generation. In order to map the matching scores to each pixel, we adopt a superpixel algorithm, i.e., SEEDS [76], [77]. By conducting superpixel segmentation, we get the superpixel labels $\mathbf{L}_{sp} = \text{SuperPixel}(\mathbf{I})$. Let $\mathbf{L}_{sp}(x_m, y_m)$ denote pixels whose superpixel labels are the same as (x_m, y_m) . We set scores of these pixels the same as the score of matched point (x_m, y_m) , i.e., $\mathbf{S}_{sp}(\mathbf{L}_{sp}(x_m, y_m)) = s_m$. Thus, we get our pixel-level scores \mathbf{S}_{sp} from superpixel and matched points. Besides, we also generate a pixel-level score map \mathbf{S}_p from backbone scores \mathbf{S} and candidate proposals which have matched points. Concretely, we set $\mathbf{S}_p(x, y) = \mathbf{S}(x, y)$ for (x, y) in the scope of $\mathbf{p}_{\tilde{p}}$ which contains matched points, otherwise $\mathbf{S}_p(x, y) = 0$. Thus, our integrated score map \mathbf{S}_{in} is computed based on \mathbf{S}_{sp} and \mathbf{S}_p as follows:

$$\mathbf{S}_{in} = \frac{1}{1 + \exp(-\phi(\alpha \cdot \mathbf{S}_{sp} + \beta \cdot \mathbf{S}_p + \gamma))} \quad (9)$$

where α , β and γ are three parameters to balance \mathbf{S}_{sp} and \mathbf{S}_p . We set $\alpha = \beta = 1$ to make \mathbf{S}_{sp} and \mathbf{S}_p have the same contribution. We set $\gamma = -0.5$ to make sure it has the same distribution when $\mathbf{S}_{sp}(i) = 0$. ϕ indicates the amplifying factor to control score distribution of \mathbf{S}_{in} , and is set to 4.

Integrated score map refinement for label generation. The directly computed \mathbf{S}_{in} has some small isolated regions or holes inside detected regions, because there are some false-alarm or missing-detected regions. In order to neglect regions with lower matching probability and refine contours according to image content, we formulate fully connected CRF (Conditional Random Field) [78] based on \mathbf{S}_{in} and image \mathbf{I} , to get final labels. Our problem is that we have an image \mathbf{I} which has N pixels, and we try to fulfill a segmentation task with two classes. A segmentation of \mathbf{I} is modelled as a random field $\mathbf{X} = \{X_1, \dots, X_N\}$ where each random variable X_n takes values of $\{0, 1\}$. "1" is used to label forged locations and corresponding genuine ones, while "0" is for remaining parts. A conditional random field (\mathbf{I}, \mathbf{X}) is characterized by a Gibbs distribution $P(\mathbf{X}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp(-E(\mathbf{X}|\mathbf{I}))$, where the energy function $E(\mathbf{X}|\mathbf{I})$ is given by:

$$E(\mathbf{X}|\mathbf{I}) = \sum_{i \leq N} \psi_u(X_i|\mathbf{I}) + \sum_{i \neq j \leq N} \psi_p(X_i, X_j|\mathbf{I}) \quad (10)$$

where $\psi_u(X_i|\mathbf{I})$ is called unary potential. In our work, our computed \mathbf{S}_{in} is treated as the unary potential:

$$\psi_u(X_i|\mathbf{I}) = \mathbf{S}_{in}(i) \quad (11)$$

And $\psi_p(X_i, X_j|\mathbf{I})$ is called pairwise potential. It accounts for the joint distribution of pixels i and j . It allows us to explicitly model interactions between pixels, such as pixels with similar colour are likely the same class. And ψ_p is formulated as weighted sum of Gaussian kernels:

$$\psi_p(X_i, X_j|\mathbf{I}) = \mu(X_i, X_j)k(\mathbf{f}_i, \mathbf{f}_j) \quad (12)$$

where $\mu(X_i, X_j)$ is a simple label compatibility function, which is given by the Potts model $\mu(X_i, X_j) = [X_i \neq X_j]$. It

penalizes nearby similar pixels that are assigned different labels. $k(\mathbf{f}_i, \mathbf{f}_j)$ denotes Gaussian kernels with feature vectors \mathbf{f}_i and \mathbf{f}_j in an arbitrary feature space. Specifically, contrast-sensitive two-kernel potentials are formulated in our model:

$$k(\mathbf{f}_i, \mathbf{f}_j) = \underbrace{w^{(1)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right)}_{\text{appearance kernel}} + \underbrace{w^{(2)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right)}_{\text{smoothness kernel}} \quad (13)$$

where I_i and I_j are color vectors, p_i and p_j are positions. $w^{(1)}$ and $w^{(2)}$ are linear combination weights. θ_α , θ_β and θ_γ are controlling parameters. The appearance kernel drives nearby pixels with similar color to be in the same class. The smoothness kernel removes small isolated regions. In our implementation, we adopt ConvCRF [43] for inference. ConvCRF adds the assumptions of conditional independence fully-connected CRF, and reformulates the inference in terms of convolutions which are implemented highly efficiently on GPUs.

4 EXPERIMENTAL EVALUATION

4.1 Implementation Details

Real-world copy-move forgery needs forgers to manually manipulate images and pasted regions. Therefore, the available copy-move forgery datasets are not sufficient for training an end-to-end deep matching network. Thus, we automatically generate a synthetic training set and a synthetic testing set from MS COCO 2014 training images and testing images respectively. For each image, we resize it to 512×512 , randomly select one annotated region under different transformations, and paste it to a random position of this image. All pasted regions randomly suffer four types of transformations, i.e., rotation changes in $\mathbb{U}(-60, 60)$, scale changes in $\mathbb{U}(0.5, 4)$, luminance changes in $\mathbb{U}(-32, 32)$, and deformation changes in $\mathbb{U}(0.5, 2)$ (decrease or increase the width of a tampered region). Following this strategy, we generate 120,000 training images and 1,000 testing images.

The self deep matching network is trained with a single spatial cross entropy loss, and parameters in the basic feature extraction network are initialized using VGG16 [33] which is trained for image classification. Similarly, alternative formulations are initialized using corresponding classification networks. We conduct 16-epoch training, and adopt the Adadelta optimizer [79]. The input images are randomly resized in the range of $[256 \times 256, 512 \times 512]$. Limited by our GPU memory, the batch size is set to 6 (a larger batch size may further improve our performance). As for the Proposal SuperGlue stage, no further training is needed. We directly adopt their trained DeepMask [40], SuperPoint [41], SuperGlue [42] models.

4.2 Backbone Network Ablation Study

Our backbone self deep matching network incorporates encoder-decoder architecture with atrous convolution, skip matching, feature normalization, correlation normalization and spatial attention. In TABLE 1, step-by-step analyses are

provided on the synthetic testing set. We compute the pixel-level IoU, precision, recall and F1-score for each image, and accumulate their average scores. Two protocols are adopted: ‘‘Protocol-All’’ means we compute the average scores of all evaluated images, and ‘‘Protocol-Detected’’ only computes average scores of detected images.

It shows that each component of our backbone network plays an important role in improving its localization performance. Specifically, ‘‘encoder-decoder’’ denotes a pure architecture with a feature extractor and a decoder, there is no skips, normalization and attention; In ‘‘encoder-decoder-skip’’, we add skip matching; In ‘‘encoder-decoder-skip-normfeats’’, input feature maps before correlation computation are normalized; In ‘‘SelfDM’’, correlation maps are followed by ReLU and L2-normalization. Besides, we make a discussion on the selection of T in Eq. (7). We find that it can even achieve comparable performance when $T = 16$. When we set $T = 48$, it can get higher scores. There is no further improvement with $T = 64$. More importantly, we test four types of attention-based self-correlation formulations, i.e., ‘‘SelfDM-SA’’ with spatial attention, ‘‘SelfDM-MSA’’ with multi-head spatial attention [51], ‘‘SelfDM-SACA’’ with spatial attention before channel attention and ‘‘SelfDM-CASA’’ with spatial attention after channel attention [52]. Although, ‘‘SelfDM-MSA’’, ‘‘SelfDM-SACA’’ and ‘‘SelfDM-CASA’’ have more parameters, their performance is barely satisfactory. We guess the main reasons are that: (1) single spatial attention can already reinforce correlated regions, while multiple spatial attention with redundant information may mislead our model; (2) SE blocks [52] (channel attention) improve the classification performance by densely adding them into convolutional blocks, while we only add them into self-correlation which is not sufficient enough. After comprehensive comparison, we select the version with spatial attention, i.e., ‘‘SelfDM-SA’’.

In section 3.1.1, alternative formulations are discussed. Two deeper networks (ResNet50, ResNet101) and three light-weight networks (MobileNetV2, MobileNetV3, ShuffleNetV2) are constructed. We find that ‘‘SelfDM-SA-ResNet50’’ can slightly improve the performance of our backbone network, while the performance of deeper ‘‘SelfDM-SA-ResNet101’’ is even worse. It shows that high-level features with richer semantic information are not as important as discriminative features with rich spatial information, in a deep matching task. Furthermore, light-weight networks are compared. The performance of ‘‘SelfDM-SA-MobileNetV3’’ is even better. So we finally select the default ‘‘SelfDM-SA’’ with VGG, ‘‘SelfDM-SA-ResNet50’’ and ‘‘SelfDM-SA-MobileNetV3’’ for further comparison in the next section.

4.3 Comparison with State-of-the-art Methods

We adopt four datasets for comprehensive comparisons: our synthetic testing set, CoMoFoD dataset [80], CASIA CMFD dataset [2], and MICC-F220 dataset [16].

Pasted regions in our synthetic testing set have gone through multiple changes with greater extent. Besides, there is only one pair of similar regions in each image, and there is no obvious ‘‘disturbance’’ (similar but genuine regions) for the most cases. So it can indicate the capability and

TABLE 1
Step-by-step analyses on the synthetic testing set.

Variant	Protocol-All				Protocol-Detected				T	Trainable params
	IoU	Precision	Recall	F1-score	IoU	Precision	Recall	F1-score		
encoder-decoder	0.4982	0.5930	0.7905	0.6328	0.4992	0.5942	0.7921	0.6341	48	7,772,209
encoder-decoder-skip	0.5523	0.6523	0.7927	0.6835	0.5529	0.6530	0.7935	0.6841	48	14,985,265
encoder-decoder-skip-normfeats	0.6822	0.7793	0.8332	0.7897	0.6822	0.7793	0.8332	0.7897	48	14,985,265
SelfDM	0.6959	0.7813	0.8506	0.7999	0.6959	0.7813	0.8506	0.7999	48	14,985,265
SelfDM-16	0.6818	0.7703	0.8427	0.7891	0.6832	0.7719	0.8444	0.7907	16	14,851,633
SelfDM-32	0.6881	0.7899	0.8255	0.7927	0.6881	0.7899	0.8255	0.7927	32	14,918,449
SelfDM-48	0.6959	0.7813	0.8506	0.7999	0.6959	0.7813	0.8506	0.7999	48	14,985,265
SelfDM-64	0.6942	0.8038	0.8221	0.7970	0.6949	0.8046	0.8229	0.7978	64	15,052,081
SelfDM-SA	0.7233	0.8458	0.8227	0.8216	0.7240	0.8467	0.8235	0.8225	48	15,724,148
SelfDM-MSA	0.7119	0.8466	0.8064	0.8096	0.7126	0.8475	0.8072	0.8104	48	17,940,797
SelfDM-SACA	0.7129	0.8370	0.8204	0.8136	0.7129	0.8370	0.8204	0.8136	48	15,799,236
SelfDM-CASA	0.7195	0.8472	0.8164	0.8182	0.7195	0.8472	0.8164	0.8182	48	15,799,236

TABLE 2
Feature extractor comparisons on the synthetic testing set.

Variant	Protocol-All				Protocol-Detected				Trainable params
	IoU	Precision	Recall	F1-score	IoU	Precision	Recall	F1-score	
SelfDM-SA	0.7233	0.8458	0.8227	0.8216	0.7240	0.8467	0.8235	0.8225	15,724,148
SelfDM-SA-ResNet50	0.7372	0.7809	0.9191	0.8312	0.7372	0.7809	0.9191	0.8312	30,664,372
SelfDM-SA-ResNet101	0.7176	0.8246	0.8359	0.8059	0.7183	0.8254	0.8368	0.8067	49,656,500
SelfDM-SA-MobileNetV2	0.7126	0.7957	0.8584	0.8118	0.7126	0.7957	0.8584	0.8118	4,557,012
SelfDM-SA-MobileNetV3	0.7512	0.8575	0.8467	0.8412	0.7512	0.8575	0.8467	0.8412	4,092,268
SelfDM-SA-ShuffleNetV2	0.6676	0.7692	0.8137	0.7745	0.6676	0.7692	0.8137	0.7745	2,920,602

robustness of algorithms to detect appearance similar regions under different transformations. We select a representative keypoint-based method (LiJ [5]), an advanced block-based method (Cozzolino [6]), and an end-to-end deep learning method (BusterNet [2]) for comparison. Scores in TABLE 3 are generated by their codes provided by the authors. It clearly shows that classical methods (“LiJ” and “Cozzolino”) are not robust enough against different transformations, while their detected regions mostly are accurate (comparable scores in “Protocol-Detected”). Our backbone network has strong ability to detect similar regions. Since there are few “disturbances”, the ability of Proposal SuperGlue to remove false-alarmed regions is not obvious in this dataset. However, it still can be seen that precisions are increased and overall scores are higher. Specifically, “SelfDM-SA+PS” indicates the two-stage version without ConvCRF optimization, “SelfDM-SA+PS+CRF” adds ConvCRF, and “SelfDM-SA+CRF*” directly conducts ConvCRF on the backbone network which is used to demonstrate the effectiveness of Proposal SuperGlue. Furthermore, visual comparisons are provided in Fig. 3. In rows 1 and 2, we provide images with obvious scale changes, SelfDM-SA can already achieve satisfied performance. In the column of “SelfDM-SA+PS”, proposals (light blue rectangular regions with lower scores) can enclose suspected regions, and further SuperGlue with SuperPoint can be conducted. With the help of ConvCRF, the score distribution can be optimized. BusterNet only detects their approximate locations without accurate boundaries. And it is difficult for classical

methods to detect regions under severe transformations like scale or rotation changes. In rows 3 to 6, we provide four challenging cases. Our backbone network detects some false-alarmed regions or only detects partial regions, while proposal SuperClue can enclose those regions and clearly optimize the detected regions. In the last two rows, we also provide two failure cases. SelfDM-SA can already generate an accurate result while Proposal SuperGlue causes some false-alarmed regions around boundaries in row 7. Or no meaningful proposals are obtained in row 8.

The CoMoFoD dataset consists of 200 copy-move forged images with resolution 512×512 . Besides the version with no postprocessing, these images are processed under 6 kinds of postprocessing respectively, namely brightness change (BC), contrast adjustments (CA), color reduction (CR), image blurring (IB), JPEG compression (JC), and noise adding (NA). In TABLE 4, two measure protocols are used. “Correctly Detected Average” only computes the average score of correctly detected images. An image is referred as correctly detected if its pixel-level F1-score is higher than 0.5. “Overall Average” computes their average scores of all images. We find that although our backbone network can achieve excellent performance on synthetic testing images, there is no obvious advantage on CoMoFoD. The main reason is that pasted regions in CoMoFoD have gone through only slight changes, compared methods can already achieve good performance. Besides, limited by the training set, the majority of synthetic images only have a pair of similar objects. In another word, there are few of disturbances with similar

TABLE 3
Comparisons with the state-of-the-art methods on the synthetic testing set.

Methods	Protocol-All				Protocol-Detected			
	IoU	Precision	Recall	F1-score	IoU	Precision	Recall	F1-score
Lij [5]	0.3188	0.3620	0.3723	0.3597	0.6826	0.7752	0.7972	0.7702
Cozzolino [6]	0.2377	0.3105	0.2584	0.2728	0.6911	0.9027	0.7513	0.7931
BusterNet [2]	0.3349	0.5814	0.3764	0.4213	0.4412	0.7660	0.4959	0.5550
SelfDM-SA	0.7233	0.8458	0.8227	0.8216	0.7240	0.8467	0.8235	0.8225
SelfDM-SA+PS	0.7198	0.8445	0.8232	0.8186	0.7205	0.8453	0.8239	0.8194
SelfDM-SA+PS+CRF	0.7403	0.8785	0.8176	0.8308	0.7433	0.8820	0.8209	0.8342
<i>SelfDM-SA+CRF*</i>	<i>0.7317</i>	<i>0.8921</i>	<i>0.7958</i>	<i>0.8252</i>	<i>0.7376</i>	<i>0.8993</i>	<i>0.8024</i>	<i>0.8309</i>
SelfDM-SA-ResNet50	0.7372	0.7809	0.9191	0.8312	0.7372	0.7809	0.9191	0.8312
SelfDM-SA-ResNet50+PS	0.7247	0.7742	0.9112	0.8232	0.7247	0.7742	0.9112	0.8232
SelfDM-SA-ResNet50+PS+CRF	0.7438	0.8066	0.8986	0.8358	0.7438	0.8066	0.8986	0.8358
SelfDM-SA-MobileNetV3	0.7512	0.8575	0.8467	0.8412	0.7512	0.8575	0.8467	0.8412
SelfDM-SA-MobileNetV3+PS	0.7364	0.8488	0.8391	0.8302	0.7364	0.8488	0.8391	0.8302
SelfDM-SA-MobileNetV3+PS+CRF	0.7531	0.8848	0.8281	0.8394	0.7538	0.8856	0.8289	0.8403

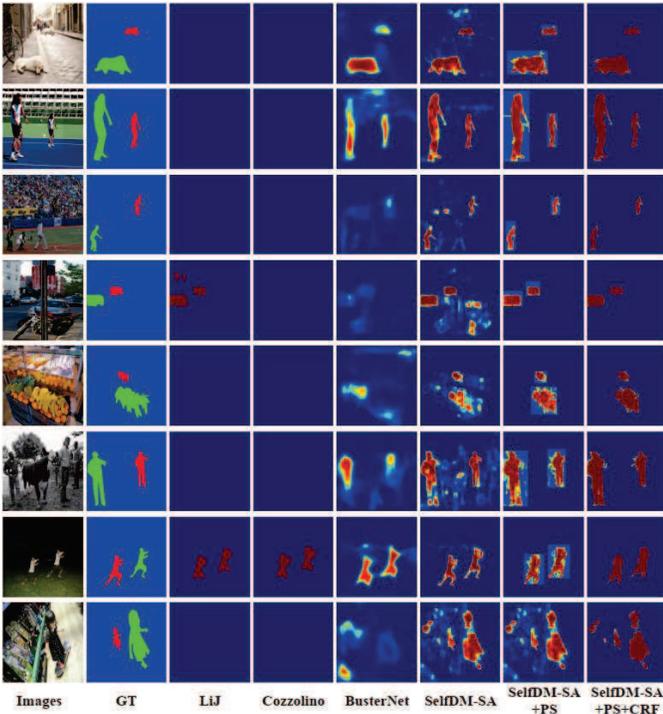


Fig. 3. Example copy-move forgery detection results on the synthetic testing set. In the “GT” column, red regions indicate forged regions and green regions are original regions.

appearance. Our backbone network has strong ability to detect similar objects under severe transformations, so that there are inevitably some false-alarmed regions (rows 1, 3, 4, 5, 6, 8 in Fig. 4), which affect the scores on CoMoFoD. With the help of Proposal SuperGlue, it can be clearly seen that we can remove some false-alarmed regions and complement miss-detected regions (Fig. 4). Thus, the performance can be obviously improved. Besides, the robustness against different postprocessing is evaluated and shown in Fig. 5. Our two-stage version, i.e., SelfDM-SA+PS+CRF, can achieve consistently higher scores under different attacks, which also demonstrate the high robustness of our method. As

for the alternative networks, i.e., SelfDM-SA-ResNet50 and SelfDM-SA-MobileNetV3, our Proposal SuperGlue can also notably improve their performance. Especially, the precision scores are improved.

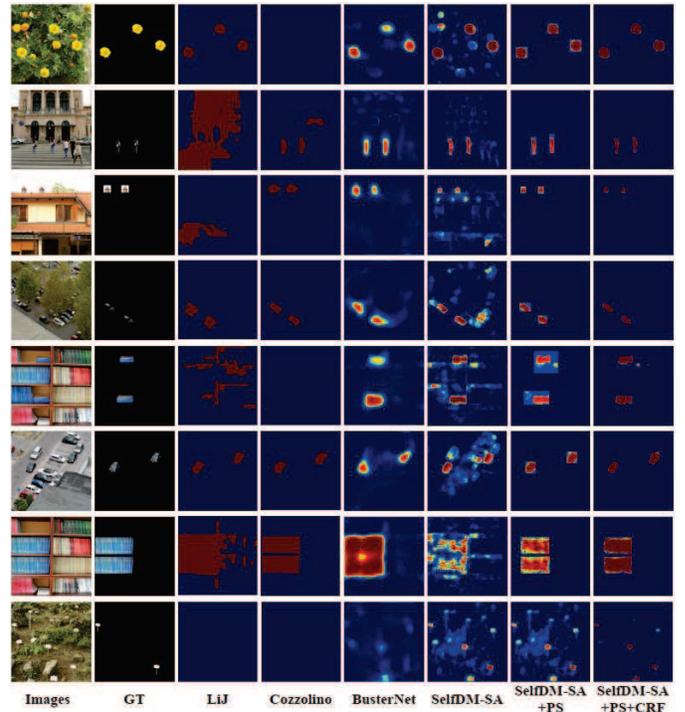


Fig. 4. Example copy-move forgery detection results on CoMoFoD.

CASIA CMFD dataset is selected from CASIA TIDEv2.0 dataset by Wu et al. [2]. There are 1313 CMFD samples and their authentic counterparts. They provide 256×256 images and masks as a HDF dataset. In our experiments, both pixel-level and image-level scores are computed. Pixel-level scores are the overall average scores of all CMFD samples (the same as “Overall Average” in TABLE 4). Our backbone network based on VGG, i.e., SelfDM-SA, can already achieve higher scores, and Proposal SuperGlue with ConvCRF can further improve its performance. Dramatically,

the MobileNetV3 version can achieve the best performance. It further demonstrates that the discriminative capability of features are more important in the deep matching task, and it is different from image understanding tasks (e.g. image classification, object detection, semantic segmentation) in which high-level features with more semantic information play a more important role.

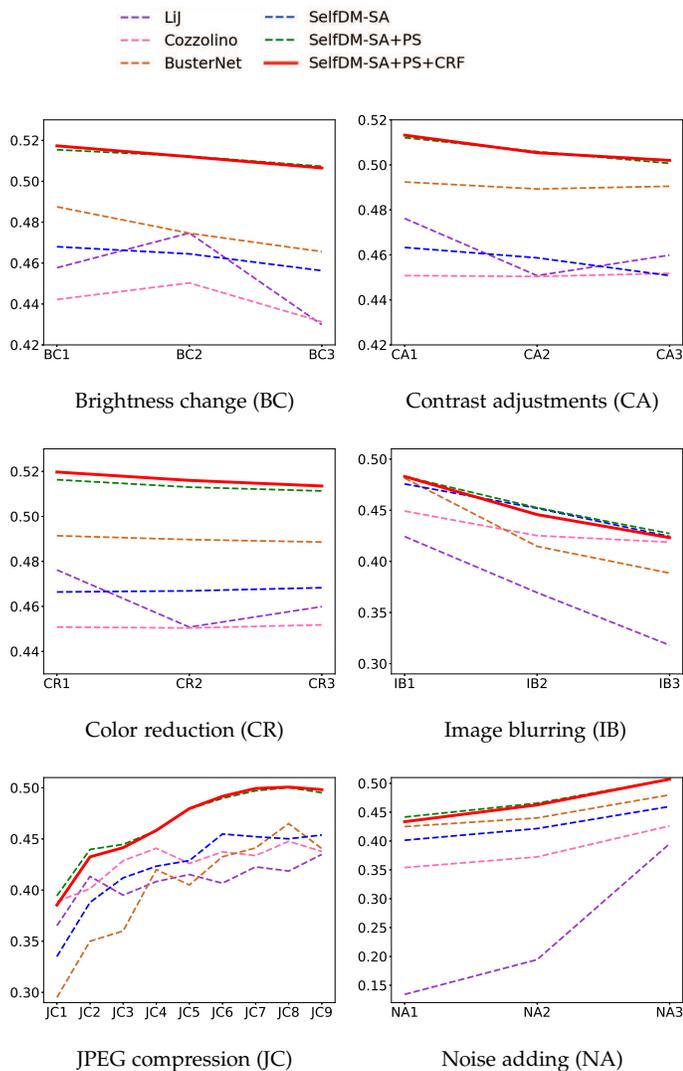


Fig. 5. Pixel-level F1 scores (y-axis) on CoMoFoD under attacks (x-axis).

MICC-F220 is composed by 220 images: 110 tampered images and 110 originals. There is no ground truth, and we evaluate the image-level performance. All the methods are evaluated by True Positive Rate (TPR), False Positive Rate (FPR) and corresponding F1-score, which are computed as: $TPR = TP / (TP + FN)$, $FPR = FP / (TN + FP)$, $F_1 = 2TP / (2TP + FP + FN)$, where TP denotes true positive, TN denotes true negative, FN denotes false negative and FP denotes false positive. SelfDM-SA and the corresponding Proposal SuperGlue version can achieve better performance than many other state-of-the-art methods. Especially, we can achieve higher TPRs. However, the alternative formulations have higher FPRs, FPRs of the ResNet50 version are even greater than 0.6. Considering all the experimental results on four datasets, we do not recommend

the use of deeper networks for feature extraction, because they have more parameters, low efficiency and high false-alarm rates. The VGG version is more stable and robust. The MobileNetV3 version has less parameters to learn and can achieve comparable performance on different datasets. Even so, we find that our two-stage framework can be applied to backbone networks with different feature extractors to achieve better performance.

5 CONCLUSION

In this paper, we propose a two-stage framework for deep learning based copy-move forgery detection. Our two-stage framework integrates self deep matching and keypoint matching by obtaining highly suspected proposals. The first stage is a backbone network which adopts atrous convolution, skip matching, and spatial attention. In the second stage, our Proposal SuperGlue is proposed to remove false alarms and complement incomplete regions. Specifically, we build a proposal selection module to enclose suspected regions, and conduct pairwise matching based on SuperPoint and SuperGlue. Integrated score map generation and refinement methods are proposed to obtain final results. Our two-stage framework can achieve consistently better performance on different public datasets. The two-stage framework relies on the performance of the backbone network. In the future, our two-stage framework can be further improved by designing a more powerful backbone network.

REFERENCES

- [1] Y. Liu, Q. Guan, X. Zhao, and Y. Cao, "Image forgery localization based on multi-scale convolutional neural networks," in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2018, pp. 85–90.
- [2] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Busternet: Detecting copy-move image forgery with source/target localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 168–184.
- [3] S.-J. Ryu, M.-J. Lee, and H.-K. Lee, "Detection of copy-rotate-move forgery using zernike moments," in *International workshop on information hiding*. Springer, 2010, pp. 51–65.
- [4] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on information forensics and security*, vol. 7, no. 6, pp. 1841–1854, 2012.
- [5] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2015.
- [6] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy-move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.
- [7] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted jpeg bitstreams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1055–1067, 2016.
- [8] T. Qiao, X. Luo, T. Wu, M. Xu, and Z. Qian, "Adaptive steganalysis based on statistical model of quantized dct coefficients for jpeg images," *IEEE Transactions on Dependable and Secure Computing*, 2019.
- [9] M. Bashar, K. Noda, N. Ohnishi, and K. Mori, "Exploring duplicated regions in natural images," *IEEE Transactions on Image Processing*, 2010.
- [10] S.-J. Ryu, M. Kirchner, M.-J. Lee, and H.-K. Lee, "Rotation invariant localization of duplicated image regions based on zernike moments," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1355–1370, 2013.
- [11] Y. Li, "Image copy-move forgery detection based on polar cosine transform and approximate nearest neighbor searching," *Forensic science international*, vol. 224, no. 1-3, pp. 59–67, 2013.

TABLE 4
Comparisons on CoMoFoD dataset with no attack.

Method	Correctly Detected Average				Overall Average		
	Detected Rate	Precision	Recall	F1-score	Precision	Recall	F1-score
Ryu2010 [3]	0.450	0.9627	0.6984	0.7993	0.4578	0.3435	0.3737
Lij [5]	0.510	0.8042	0.9586	0.8616	0.4247	0.6633	0.4644
Cozzolino [6]	0.505	0.8132	0.9384	0.8591	0.4174	0.5042	0.4440
Wu2018 [31]	0.265	0.6111	0.7148	0.6313	0.3629	0.4041	0.3113
BusterNet [2]	0.585	0.8352	0.7875	0.8009	0.5734	0.4939	0.4926
SelfDM-SA	0.475	0.7086	0.8210	0.7350	0.5722	0.5216	0.4660
SelfDM-SA+PS	0.575	0.7895	0.8087	0.7641	0.6086	0.5624	0.5151
SelfDM-SA+PS+CRF	0.545	0.8139	0.8282	0.7943	0.6375	0.5444	0.5172
SelfDM-SA-ResNet50	0.520	0.7518	0.7394	0.7208	0.5340	0.5467	0.4753
SelfDM-SA-ResNet50+PS	0.545	0.8439	0.7668	0.7786	0.5910	0.5631	0.5108
SelfDM-SA-ResNet50+PS+CRF	0.555	0.8728	0.7432	0.7773	0.6231	0.5342	0.5088
SelfDM-SA-MobileNetV3	0.465	0.6526	0.8468	0.7096	0.4833	0.5198	0.4299
SelfDM-SA-MobileNetV3+PS	0.530	0.7488	0.8137	0.7438	0.5170	0.5260	0.4645
SelfDM-SA-MobileNetV3+PS+CRF	0.505	0.7885	0.8202	0.7742	0.5600	0.5049	0.4695

TABLE 5
Performance analysis on CASIA CMFD dataset.

Method	Pixel Level			Image Level		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Ryu2010 [3]	0.2271	0.1336	0.1640	0.9701	0.2447	0.3908
Christlein [4]	0.3709	0.0014	0.0023	0.6849	0.6782	0.6815
Cozzolino [6]	0.2492	0.2681	0.2543	0.9951	0.3061	0.4682
Wu2018 [31]	0.2397	0.1379	0.1464	0.6637	0.7359	0.6980
BusterNet-simi [2]	0.4723	0.4844	0.4372	0.7153	0.8073	0.7585
BusterNet [2]	0.5571	0.4383	0.4556	0.7822	0.7389	0.7598
SelfDM-SA	0.6551	0.4353	0.4635	0.7707	0.7807	0.7757
SelfDM-SA+PS	0.6485	0.4531	0.4709	0.7860	0.7609	0.7732
SelfDM-SA+PS+CRF	0.6494	0.4520	0.4782	0.7860	0.7609	0.7732
SelfDM-SA-ResNet50	0.5358	0.4500	0.4356	0.6038	0.9238	0.7303
SelfDM-SA-ResNet50+PS	0.5359	0.4676	0.4464	0.6164	0.9018	0.7323
SelfDM-SA-ResNet50+PS+CRF	0.5729	0.4679	0.4595	0.6164	0.9018	0.7323
SelfDM-SA-MobileNetV3	0.6248	0.4778	0.4843	0.6914	0.8294	0.7542
SelfDM-SA-MobileNetV3+PS	0.6272	0.4856	0.4891	0.7040	0.8096	0.7531
SelfDM-SA-MobileNetV3+PS+CRF	0.6362	0.4752	0.4918	0.7040	0.8096	0.7531

TABLE 6
Image-level performance on MICC-F220 dataset.

Method	TPR	FPR	F1-score
Cozzolino [6]	0.8455	0.1727	0.8378
Lij [5]	0.7091	0.1727	0.7536
GoDeep [45]	0.4545	0.4182	0.4854
Zandi [20]	0.7818	0.4818	0.6908
BusterNet [2]	0.4909	0.2000	0.5806
SelfDM-SA	0.9273	0.2545	0.8500
SelfDM-SA+PS+CRF	0.9182	0.2272	0.8559
SelfDM-SA-ResNet50	0.9727	0.6909	0.7304
SelfDM-SA-ResNet50+PS+CRF	0.9545	0.6454	0.7343
SelfDM-SA-MobileNetV3	0.9636	0.4182	0.8092
SelfDM-SA-MobileNetV3+PS+CRF	0.9545	0.3909	0.8140

"An efficient scheme for detecting copy-move forged images by local binary patterns," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 1, pp. 46–56, 2013.

- [12] L. Li, S. Li, H. Zhu, S.-C. Chu, J. F. Roddick, and J.-S. Pan, "An efficient scheme for detecting copy-move forged images by local binary patterns," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 1, pp. 46–56, 2013.
- [13] T. Mahmood, T. Nawaz, A. Irtaza, R. Ashraf, M. Shah, and M. T. Mahmood, "Copy-move forgery detection technique for forensic analysis in digital images," *Mathematical Problems in Engineering*, vol. 2016, 2016.
- [14] X. Bi and C.-M. Pun, "Fast copy-move forgery detection using local bidirectional coherency error refinement," *Pattern Recognition*, vol. 81, pp. 161–175, 2018.
- [15] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 857–867, 2010.
- [16] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy-move attack detection and transformation recovery," *IEEE transactions on information forensics and security*, vol. 6, no. 3, pp. 1099–1110, 2011.
- [17] P. Kakar and N. Sudha, "Exposing postprocessed copy-paste forgeries through transform-invariant features," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1018–1028, 2012.
- [18] C.-M. Pun, X.-C. Yuan, and X.-L. Bi, "Image forgery detection using adaptive oversegmentation and feature point matching,"

- IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1705–1716, 2015.
- [19] E. Ardizzone, A. Bruno, and G. Mazzola, “Copy-move forgery detection by matching triangles of keypoints,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2084–2094, 2015.
- [20] M. Zandi, A. Mahmoudi-Aznavah, and A. Talebpour, “Iterative copy-move forgery detection based on a new interest point detector,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2499–2512, 2016.
- [21] D. Cozzolino, G. Poggi, and L. Verdoliva, “Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection,” in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2017, pp. 159–164.
- [22] Y. Wu, W. Abd-Almageed, and P. Natarajan, “Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection,” in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1480–1502.
- [23] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, “Forensicttransfer: Weakly-supervised domain adaptation for forgery detection,” *arXiv preprint arXiv:1812.02510*, 2018.
- [24] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, “Learning rich features for image manipulation detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1053–1061.
- [25] X. Cun and C.-M. Pun, “Image splicing localization via semi-global network and fully connected conditional random fields,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [26] D. Cozzolino and L. Verdoliva, “Noiseprint: a cnn-based camera model fingerprint,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 144–159, 2019.
- [27] Y. Liu, X. Zhu, X. Zhao, and Y. Cao, “Adversarial learning for constrained image splicing detection and localization based on atrous convolution,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2551–2566, 2019.
- [28] N. Yu, L. S. Davis, and M. Fritz, “Attributing fake images to gans: Learning and analyzing gan fingerprints,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7556–7566.
- [29] O. Mayer and M. C. Stamm, “Forensic similarity for digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1331–1346, 2019.
- [30] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “Faceforensics++: Learning to detect manipulated facial images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1–11.
- [31] Y. Wu, W. Abd-Almageed, and P. Natarajan, “Image copy-move forgery detection via an end-to-end deep neural network,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1907–1915.
- [32] P. Korus and J. Huang, “Multi-scale analysis strategies in prnu-based tampering localization,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 809–824, 2017.
- [33] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [37] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., “Searching for mobilenetv3,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1314–1324.
- [38] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.
- [39] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.
- [40] P. O. Pinheiro, R. Collobert, and P. Dollár, “Learning to segment object candidates,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1990–1998.
- [41] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.
- [42] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superglue: Learning feature matching with graph neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4938–4947.
- [43] M. T. Teichmann and R. Cipolla, “Convolutional crfs for semantic segmentation,” *arXiv preprint arXiv:1805.04777*, 2018.
- [44] P.-T. Yap, X. Jiang, and A. C. Kot, “Two-dimensional polar harmonic transforms for invariant image representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1259–1270, 2010.
- [45] E. Silva, T. Carvalho, A. Ferreira, and A. Rocha, “Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes,” *Journal of Visual Communication and Image Representation*, vol. 29, pp. 16–32, 2015.
- [46] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [47] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [48] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” *arXiv preprint arXiv:1703.03130*, 2017.
- [49] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International conference on machine learning*, 2015, pp. 2048–2057.
- [50] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.
- [51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [52] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
- [53] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual attention network for image classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3156–3164.
- [54] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, “Cbam: Convolutional block attention module,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.
- [55] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 286–301.
- [56] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.
- [57] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [58] Y. Liu, X. Zhang, X. Zhu, Q. Guan, and X. Zhao, “Listnet-based object proposals ranking,” *Neurocomputing*, vol. 267, pp. 182–194, 2017.
- [59] B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.
- [60] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *European conference on computer vision*. Springer, 2014, pp. 391–405.
- [61] A. Humayun, F. Li, and J. M. Rehg, “Rigor: Reusing inference in graph cuts for generating object regions,” in *Proceedings of the*

- IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 336–343.
- [62] P. Krähenbühl and V. Koltun, “Geodesic object proposals,” in *European conference on computer vision*. Springer, 2014, pp. 725–739.
- [63] P. Krahenbuhl and V. Koltun, “Learning to propose objects,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1574–1582.
- [64] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [65] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping for image segmentation and object proposal generation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 128–140, 2016.
- [66] W. Kuo, B. Hariharan, and J. Malik, “Deepbox: Learning objectness with convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2479–2487.
- [67] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable object detection using deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.
- [68] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, “Learning to refine object segments,” in *European conference on computer vision*. Springer, 2016, pp. 75–91.
- [69] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, “D2-net: A trainable cnn for joint detection and description of local features,” in *CVPR 2019*, 2019.
- [70] J. Revaud, C. R. de Souza, M. Humenberger, and P. Weinzaepfel, “R2D2: reliable and repeatable DETector and descriptor,” in *Advances in Neural Information Processing Systems*, 2019, pp. 12 405–12 415.
- [71] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “Lf-net: learning local features from images,” in *Advances in neural information processing systems*, 2018, pp. 6234–6244.
- [72] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, “Lift: Learned invariant feature transform,” in *European Conference on Computer Vision*. Springer, 2016, pp. 467–483.
- [73] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.
- [74] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.
- [75] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.
- [76] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, “Seeds: Superpixels extracted via energy-driven sampling,” in *European conference on computer vision*. Springer, 2012, pp. 13–26.
- [77] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, “Seeds: Superpixels extracted via energy-driven sampling,” *International Journal of Computer Vision*, vol. 111, no. 3, pp. 298–314, 2015.
- [78] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” in *Advances in neural information processing systems*, 2011, pp. 109–117.
- [79] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
- [80] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic, “Comofod—new database for copy-move forgery detection,” in *Proceedings ELMAR-2013*. IEEE, 2013, pp. 49–54.