Hindawi
Security and Communication Networks
Volume 2022, Article ID 9755509, 10 pages
https://doi.org/10.1155/2022/9755509

Research Article

Semantic Modeling and Pixel Discrimination for Image Manipulation Detection

Ziyu Xue[1,2] **Xiuhua Jiang,**[1,3] **and Qingtong Liu**[2]

[1]*School of Information and Communication Engineering, Communication University of China, Beijing, China*
[2]*Academy of Broadcasting Science, NRTA, Beijing, China*
[3]*Peng Cheng Laboratory, Shenzhen, China*

Correspondence should be addressed to Ziyu Xue; xzy_88@126.com

Received 27 January 2022; Revised 4 March 2022; Accepted 12 April 2022; Published 5 May 2022

Academic Editor: Beijing Chen

Image manipulation methods, such as the copy-move, splicing, and removal methods, have become increasingly mature and changed the common perception of “seeing is believing.” The credibility of digital media has been seriously damaged with the development of image manipulation methods. Most image manipulation detection methods detect traces of tampering pixel by pixel. As a result, the detected manipulation areas are separated, which results in insufficient consideration of content manipulation at the object level. In this paper, the detection of image manipulation areas based on forgery object detection and pixel discrimination is proposed. Specifically, the pixel-level detection branch resamples features and uses an LSTM to detect manipulations, such as resampling, rotation, and cropping. The goal of the forgery object detection branch, which is based on Faster R-CNN, is to extract the regions of interest and analyze the regions with high contrast as well as the forgery objects of the image. Furthermore, the fused heatmaps of the two branches are integrated with the object detection results. The noise in the heatmaps is shielded based on the forgery object information of the region proposal network. Experimental results on multiple standard forgery datasets have demonstrated the superiority of our proposed method compared with the state-of-the-art methods.

1. Introduction

With the rapid evolution of digital image manipulation, digital images can be tampered with or forged through various methods, i.e., the splicing, copy-move, and image removal methods. When splicing is implemented, a portion of the source image splits into the target image to form a new image, as shown in Figure 1(a). The copy-move method is used to paste an area of an image into the same image, as shown in Figure 1(b). In the removal method, an area in the image is removed and the area restored, as shown in Figure 1(c).

Early image manipulation detection methods employed deep neural networks to determine the type of manipulation in advance. A detection method can only solve one kind of manipulation problem. This kind of method, which is referred to as known manipulation-based detection, uses the Daubechies wavelet features to detect image patches [1] and edge reinforcement methods to build a multitask detection task [2]. Chen et al. [3] propose a parallel deep neural network scheme BusterNet for image copy-move forgery localization. With the development of manipulation methods, it has become easy to superimpose multiple image manipulations. Therefore, it is increasingly challenging to detect manipulated images based on unknown manipulation types.

The first line is splicing, the second line is copy-move, and the third line is removal.

Unknown manipulation-based detection has more significant applications than the above approach. Most existing detection approaches [4, 5] combine resampling and deep learning features to detect manipulated regions. Park et al. [6] utilized double JPEG compression features, which were merged with the additional information in the quantization

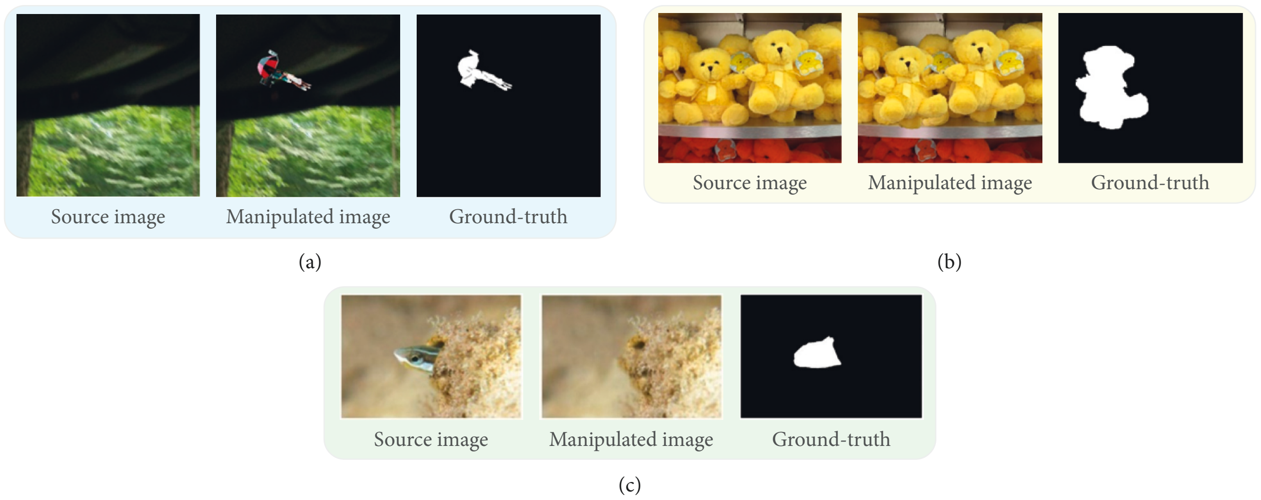

Figure 1: Image manipulation detection method. (a) Splicing. (b) Copy-move. (c) Removal.

table, to determine whether the image contains manipulated areas and locate them. Zhou et al. [7] proposed a framework that effectively combined a region proposal network (RPN) to localize the synthesis region at the object level and carry out a two-stream fine-grained bilinear pooling operation. Xiao et al. [8] proposed an approach that combines a coarse convolutional neural network (C-CNN) and a refined CNN (R-CNN) to learn the differences in the image properties to catch manipulated images. Chen et al. [9] utilize the dual-color spaces and improve the Xception architecture to detect GAN-generated faces. However, most of these approaches are pixel-level oriented. We find that image manipulation is more likely to occur at the object level. The image manipulation regions can be detected at the pixel level and object level to simultaneously contribute to improving the detection accuracy.

We have developed an approach, whose architecture is shown in Figure 2, which combines both pixel-level and object-level information. Our method consists of four parts, i.e., a forgery object-level branch, a pixel-level detection branch, a feature fusion module, and an integrated fusion module. First, the forgery object-level branch is used to extract the image features using CNNs and feed the features into the feature fusion module. Then, the ROIs are obtained based on Faster R-CNN [10]. Inspired by Bappy et al. [5], the pixel-level detection branch is designed to divide the image into 8×8 patches and resample them one by one.

Meanwhile, the LSTM learns to establish the temporal correlations between patches. Equipped with the outputs of two branches combined, the feature fusion module uses a decoder to reconstruct the features, and the heatmaps of the manipulation area are generated. The manipulation areas from the pixel-level branch are used to fine-tune the forgery object-level branch to achieve accurate forgery area labeling. The contributions of our work are summarized as follows:

(1) We propose a novel two-branch image manipulation detection framework consisting of a forgery object-level branch and a pixel-level branch. The image manipulation region is refined by two fusion modules, making our work significantly different from other state-of-the-art methods.

(2) We employ the region of interest in the forgery object-level branch to optimize the heatmap in the feature fusion module. The noise in the heatmap is masked by bounding boxes to decrease the error caused by the pixel-level detection branch and improve the detection precision of the manipulated image regions.

(3) Extensive experiments on four benchmarks have demonstrated the effectiveness of our proposed method.

The rest of this paper is organized as follows. In Section 2, we summarize the current image manipulation detection technologies. In Section 3, we elaborate on the details of our proposed method. The conducted experiments and analysis are presented in Section 4. Finally, in Section 5, we conclude the paper.

2. Related Work

Researches on detecting unknown manipulation type images consist of various approaches. Some earlier approaches [4, 6, 11] are based on manually designed features. Wu et al. [11] propose a method to divide the image into blocks and extract the resampling features for each block. A deep neural network is utilized to construct a classifier and a Gaussian conditional random field model to create a thermodynamic diagram. Meanwhile, they use the random walk method to locate the synthetic region.

Some approaches that are based on adaptive feature extraction are proposed to reduce the limitations of manual design features and improve the method's adaptability. Bappy et al. [5] construct a two-branch manipulation image detection architecture by combining the resampling feature, LSTM, and encoder-decoder architecture. A resampling detection model is utilized to extract the resampling feature of the image from each patch, and the LSTM establishes the

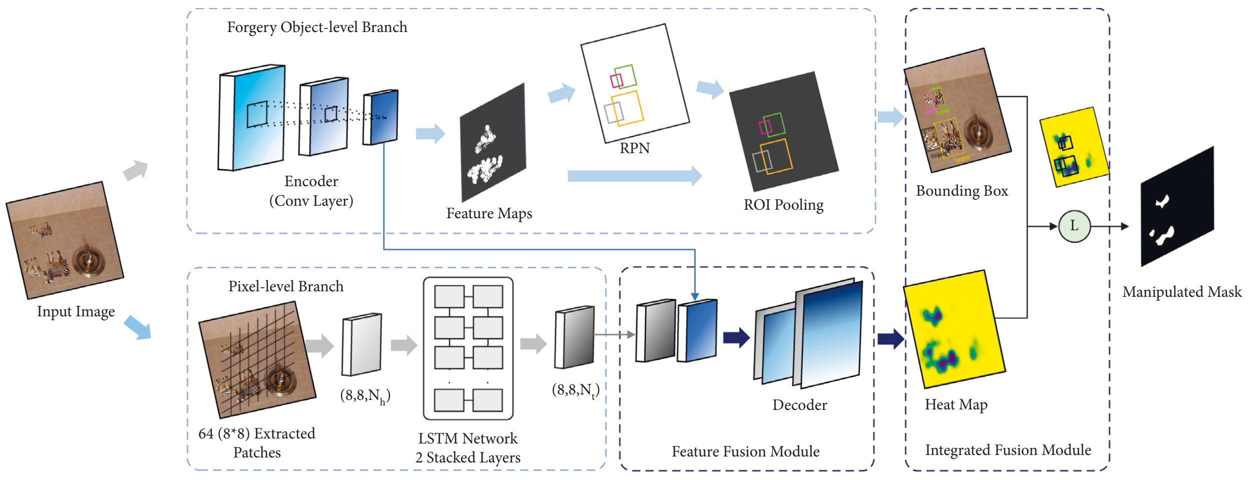

FIGURE 2: The overall architecture of our method. Forgery object-level branch: the features are extracted from the whole image using several convolutional layers. The forgery object information is extracted using encoder to form the feature maps to RPN. Pixel-level branch: the image is divided into 8×8 image patches, and the resampling features are extracted for each patch, combined with LSTM to build a temporal relationship. Feature fusion module: the forgery object-level features in the upper branch and the pixel-level features in the lower branch are integrated to produce the heat maps using a decoder. Integrated fusion module: according to the forgery object information of the RPN network, the noises in the heat maps are masked.

correlation between patches. The encoder is used to capture the spatial information of the image. After fusing the frequency and spatial features, the decoder is used to enlarge the feature to obtain the synthetic region located at the pixel level.

Mohammed et al. [12] use the CNN to obtain the abnormal image boundary features and the LSTM to establish the connection between image patches. A separate detection model is used to obtain more accurate detection results to enhance the detection effect of the copy-move manipulation image. Mazumdar et al. [13] present a two-stream encoder-decoder network. The first stream extracts the noise residuals to learn the low-level features through the encoder of the high-pass filter. The second stream extracts high-level features from the RGB values of the input image. The feature maps of both streams generated pixel-level predictions. Some other typical methods also detect pixel modifications such as resampling, copy-move, and removal.

By summarizing the above work, forgers mainly perform manipulation on objects, such as a car and sofa. Therefore, the forgery detection is based on object-level to explore the semantic information. Meanwhile, pixel-level detection is more accurate, especially at detecting edges of fake ones. It is necessary to combine both object- and pixel-level information to detect forgeries. Based on this, we present a novel framework to effectively detect the manipulation region, in which both object- and pixel-level features contribute to the detection results.

3. Network Architecture Overview

The purpose of the proposed framework is to detect image manipulations. A multitask framework is employed to model both object-level and pixel-level structures and consists of a forgery object-level branch and a pixel-level branch. A feature fusion module is further used to fuse both forgery object-level and pixel-level features. The generated heatmaps are merged with the forgery object information of the manipulated areas through an integrated fusion module to detect the manipulated areas.

3.1. Forgery Object-Level Detection Branch. We utilize Faster R-CNN in the forgery object-level branch to detect manipulation areas. A convolutional network is used to learn manipulation features, and the RPN is utilized to generate ROIs for bounding box regression.

3.1.1. Feature Extraction Network. It is essential in image manipulation detection to extract features using convolutional neural networks and ensure that the classifier can learn to discriminate the manipulated areas. In our work, we employ ResNet-101 to extract image features. Specifically, we utilize different convolution kernels to locate the manipulated area. In the first layer, the image is taken as input with dimensions of $256 \times 256 \times 3$. The network contains multiple convolutions, pooling layers, and activation functions. The residual module utilizes a parameter-free shortcut connection to optimize the residual mapping and model training.

Following [5], we utilize a convolution kernel size of 3×3 to generate 32, 64, 128, and 256 feature maps. Each residual unit in the network generates a set of feature maps normalized by batch processing in the convolutional layer. The rectified linear unit (ReLU) function is utilized as an activation function followed by a max-pooling layer with a stride of two at the end of each residual unit.

3.1.2. Region Proposal Network. The RPN in Faster R-CNN is used to extract the proposed regions. Compared with the selective search method, RPN is more efficient and easier to combine with Faster R-CNN. As the anchor is the center point of the sliding window in the original pixel space, we use the anchor as the center point to generate the proposed regions.

We first map the ROIs to the corresponding area on the feature map and shape the different ROIs to a fixed size. We take the maximum pixel value of each divided region. Then, each ROI will have a fixed size.

The loss of the forgery object-level branch is composed of two items, i.e., classification loss and regression loss, which are based on the RPN in Faster R-CNN. The *i*-th anchor can be represented as

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i, t_i^*), \tag{1}$$

where p is denoted as the probability of a manipulation area, p^* represents the ground-truth label, t is the 4-dimensional bounding box, N_{cls} and N_{reg} represent the RPN network batch and the number of anchors at each location, respectively, L_{cls} is the standard cross-entropy loss for the RPN network, L_{reg} is the smooth L_1 regression loss for the proposal bounding boxes, and the hyperparameter λ is utilized to balance the two loss items with a value of 10. The output of the forgery object-level branch is Z_{Object}, as shown in Figure 3.

3.2. Pixel-Level Detection Branch. The pixel-level detection branch is used to detect the natural statistics of copy-move, splicing, and removal, consisting of a resampling feature extraction network and an LSTM network. Following [5], we utilize the pixel-level features as the input of the feature fusion module.

3.2.1. Resampling Feature Extraction Network. Mahdian et al. [14] proposed a resampling detection approach using the Radon transform. They employed the Laplacian filter to resample each patch after the Radon transform. Bappy et al. [5] utilized the Radon transform to detect manipulations such as copy-move, splicing, and removal. They detected the tampered regions by distorting the natural statistics at the boundary. The Radon transform was proven to be effective in distinguishing manipulated and nonmanipulated patches.

Following [5], we set the size of the input image to $256 \times 256 \times 3$ and extracted 64 nonoverlapping patches from the images. The size of each block is $32 \times 32 \times 3$. Then, we utilize the square root of the 3×3 Laplacian filter to generate a linear prediction error for each patch. To prevent the periodic correlation of resampling features in linear prediction loss, the Radon transform is used to accelerate the gradient descent along with ten angles of projection. Ultimately, the fast Fourier transform (FFT) is applied to obtain the periodic signal. We balance the size of the patch and resize it to 32×32 to capture the resampling features of additional information.

3.2.2. LSTM Networks. The LSTM networks are utilized to establish the relationship between patches to analyze manipulations in the overall image. Following [5], we use a Hilbert curve to convert the multidimensional problem into a single dimension to capture the correlation between patches and guarantee the local spatial positioning for patches.

From Figure 4(a), we find the results of the Hilbert curve on the image. All the patches are connected in order. The Hilbert curve includes "cups" and "joins." A square with one open side represents a "cup." The vector connections of two "cups" are called "joins." Every cup has an entry point and an exit point. In Figure 4(b), a cup is marked with a dashed box from Figure 4(a). The curve starts at the entry point (red) and ends at the exit point (green). Meanwhile, the curve traverses four adjacent squares connected to the next cup through a dotted line. As a result, the order of the input that is fed into the LSTM network is established.

The LSTM network takes the patches associated with the Hilbert curve one by one as input and learns the relationship between adjacent patches by calculating the logarithmic interval. In our work, we employ 64 steps in the LSTM network, where each step represents a patch, and a 64-dimensional feature vector is obtained in the last layer of the LSTM. First, we denote the n-th feature of the LSTM as F_n ($F_n \epsilon R^{1\times N_h}$) and the feature map as N_t. The next feature from the LSTM network can be represented as

$$F_n' = F_n \cdot W_n + B_n. \tag{2}$$

where $W_n \in R^{N_h \times N_t}$ is a matrix and $B_n \in R^{1\times N_t}$ is the bias.

Following [5], we choose $N_h = 128$ and $N_t = 64$ in our experiment, and each patch can obtain the feature matrix of $64 \times N_t$ and is reshaped to $8 \times 8 \times N_t$. We set the cross-entropy loss as

$$L(\partial) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N}\varphi(Y^m = n)\log(Y^m = n|y^m;\partial). \tag{3}$$

Here, M represents the number of pixels, N represents the number of classes, y represents the input pixel, and $\varphi(.)$ is an indicator function. If $m = n$, the loss equals 1; otherwise, it equals 0.

3.3. Feature Fusion Module. The feature fusion module aims to synthesize forgery object and pixel features, as shown in Figure 2. Following [15], we utilize a decoder to reconstruct the fusion features and divide the manipulation area to replace the fully connected layer. We utilize multichannel filters to generate heatmaps for manipulating images for the convolutional operation. Each decoder upsamples the feature maps discovered in the previous layer and performs convolution and batch normalization operations. We employ a 3×3 size kernel [5] for the decoder and obtain 64 and 16 feature maps in the first and second layers, respectively. The output of the feature fusion module is a heatmap $Z_{\text{Fusion 1}}$ containing manipulation areas, as shown in Figure 3.

3.4. Integrated Fusion Module. The heatmaps generated by the pixel-level detection branch may contain many noisy areas, such as the blue circles in Figure 3. Generally, there are

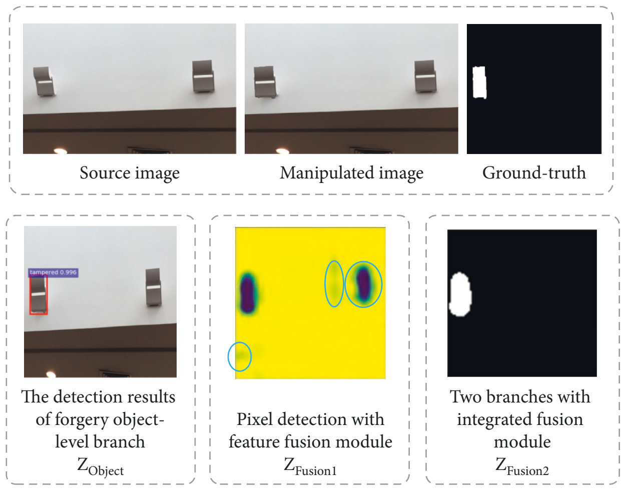
Source image
Manipulated image
Ground-truth
tampered 0.996
The detection results of forgery object-level branch Z_{Object}
Pixel detection with feature fusion module $Z_{Fusion1}$
Two branches with integrated fusion module $Z_{Fusion2}$

FIGURE 3: The outputs of our approach. The first row indicates the source image, the manipulated images, and the ground truth. The second row demonstrates the detection result of forgery object-level branch (Z_{Object}), the pixel-level branch with the feature fusion module ($Z_{\text{Fusion 1}}$), and the two branches with the integrated fusion module ($Z_{\text{Fusion 2}}$).

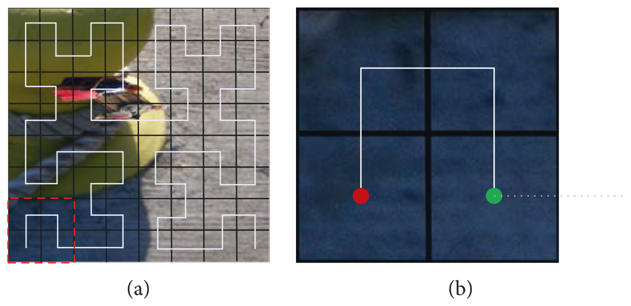

FIGURE 4: How the Hilbert curve works in an image.

two typical fusion methods when fusing Z_{Object} and $Z_{\text{Fusion 1}}$ to $Z_{\text{Fusion 2}}$, i.e., the AND operation and OR operation.

$$Z_{\text{Fusion 2}} = Z_{\text{Fusion 1}} \textcircled{\scriptsize L} Z_{\text{Object}}. \tag{4}$$

For the AND operation, we keep the heatmap in the bounding box from $Z_{\text{Fusion 1}}$. Meanwhile, the heatmap areas that are out of the bounding box are ignored, as shown in Figure 5(a). In the OR operation, we preserve all the areas in the bounding box and the heatmaps, as shown in Figure 5(b).

We devise a multiresolution fusion (MRF) strategy based on the bounding box from both the pixel and object levels, as shown in Figure 5(c). We follow the AND operation if the bounding box in the forgery object detection branch encompasses the pixel-level detection results. Meanwhile, we take all the areas in the bounding box if there are no pixel-level detection results, as described in Algorithm 1.

4. Experiments

4.1. Experimental Datasets and Evaluation Metrics. Implementation Details. We implement our proposed approach in TensorFlow. We utilize two NVIDIA Quadro RTX

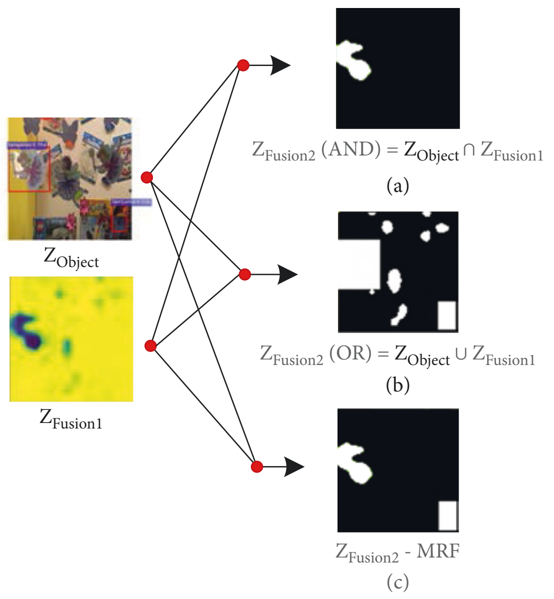
Z_{Object}
$Z_{Fusion1}$
$Z_{Fusion2}$ (AND) = $Z_{Object} \cap Z_{Fusion1}$
(a)
$Z_{Fusion2}$ (OR) = $Z_{Object} \cup Z_{Fusion1}$
(b)
$Z_{Fusion2}$ - MRF
(c)

FIGURE 5: Using integrated fusion method to choose the manipulation area.

5000 GPUs to expedite our computational load. We set the batch size to 16. The learning rate is set to 0.00003 in pixel level. We set 0.001 as the initial value in object level and then reduce it to 0.0001 after 40k steps.

Datasets: we compared our method with current state-of-the-art methods on NIST Nimble 2016 (NIST′16) [16], CASIA [17, 18], Coverage [19], and Columbia [20].

(1) NIST′16 dataset was released in 2016, and it is a standard dataset for image manipulation detection. It covers the three types of manipulations: splicing, copy-move, and removal.

(2) CASIA dataset was released in 2013, and it covers two types of manipulations: splicing and copy-move. Some postprocessing is used in manipulation regions, like filtering and blurring, to improve the difficulty of detection.

(3) Coverage dataset was released in 2016, and it covers six types of copy-move manipulation, such as copy only and shape change. The dataset includes the source and mask images and has a similarity measure for manipulating images.

(4) Columbia dataset focuses on splicing based on uncompressed images.

Evaluation metric: we utilize the F1 score in pixel- and object-level under the area under curve (AUC) as our evaluation metrics for performance comparison.

(1) F1 score is a pixel-level evaluation index for image manipulation detection, and it is used to estimate the similarity between predicted results and actual value.

(2) AUC is the area under the ROC curve, an essential indicator for measuring detection accuracy. Based on the intersections between ROC curves, we can evaluate the consequence of the model.

```
Input: Bounding boxes in the forgery object detection branch: $B_i$, $i = 1, 2, \ldots, N$. N: The number of bounding boxes. Pixel-level
detection results in pixel-level detection branch: $P_j$, $j = 1, 2, \ldots, M$. M: The number of results.
Output: forgery areas: $F_p$, $p = 1, 2, \ldots, N$
assign Intersection$_{ij}$ to $B_i$ AND $P_j$
  for each $B_i$ do
    if Intersection$_{ij}$ is not null:
      assign Intersection$_{ij}$ to $F_p$
    else
      assign $B_i$ to $F_p$
return $F_p$
```

Algorithm 1: The preprocessing of the multi-resolution fusion (MRF) strategy.

Baseline models: we compare our approach with different baseline models.

(1) ELA [21]: this approach detects quality loss caused by JPEG compression by calculating the error between the actual and manipulation areas.

(2) NOI1 [22]: this approach simulates local noise through wavelet coefficients and utilizes the inconsistency of the noise to detect the manipulation area.

(3) CFA1 [23]: this approach is based on the CFA estimation method. It employs adjacent pixels to simulate the filter array image from the camera and calculate the manipulation probability for each pixel.

(4) MFCN [2]: this approach is based on an edge-enhanced multitask complete convolutional network. Moreover, it is used to detect manipulation by predicting the area edge.

(5) J-LSTM [24]: this approach is based on the LSTM network, which judges the pixel-level manipulations by separating the image into blocks.

(6) RGB-N [7]: this approach is based on Faster R-CNN to establish a model for RGB and noise streams.

(7) LSTM-Encoder [5]: this approach employs a hybrid CNN-LSTM model to detect manipulation regions.

(8) C2RNet [8]: this approach includes C-CNN and R-CNN to distinguish the genuine and manipulation images.

Pretrained model: we train the forgery object-level and pixel-level branch separately. The forgery object-level branch, which needs to set the bounding box as the object label by the frame around the pixel-level ground truth, is trained first. Then, we train the pixel branch and employ the features extracted by the forgery object encoder to train the feature fusion module. Finally, we combine the results in an integrated fusion module using logical operations in Section 3.4.

We utilize the synthetic dataset created by Bappy et al. [5] in the pixel-level branch using the DRESDEN, COCO, and NIST′16 datasets. We follow [7] to set up the forgery object-level branch. Moreover, we utilize ResNet-101 in Faster R-CNN, which is pretrained on ImageNet, to extract the features. We train the pixel branch by using 90% of the images for training and 10% for validation.

4.2. Experimental Analysis

Experiment preparation: we test our proposed method on four datasets, NIST′16 [16], CASIA [17, 18], Coverage [19], and Columbia. Table 1 shows the comparison results of the pixel-level F1 score and AUC. We compare four sets of experiments. The first column and the second column are the results of the single pixel-level branch and single forgery object-level branch. The third column is the result of the double branch, which includes the AND operation, OR operation, and the MRF proposed in Section 3.4.

The MRF has a better performance, as shown in Table 1. Therefore, we choose the MRF as the follow-up experimental approach and set 0.2 as the manipulated threshold in the heatmap.

Result analysis: Table 2 lists the F1 score comparison between our method and the baselines. Table 3 provides the AUC comparison. We utilize the experimental results from [7, 13] and [5].

As shown in Table 2, the F1 score is a classification accuracy metric that combines precision and recall, which means the larger the F1 score is, the more robust the model is. The CASIA dataset has postprocessing methods, which affect our forgery object-level branch in detected forged objects. In comparison to our method, the LSTM-encoder [5] approach utilizes pixel-level detection and focuses on spatial cues. As a result, the LSTM-encoder [5] approach's F1 score is 0.3% higher than ours on CASIA. However, our approach combines both object- and pixel-level branches. Using our model, the F1 score on the other datasets is improved and the detection of postprocessing methods is considered. As a result, our approach performs better than other methods on the NIST′16 [16], Coverage [19], and Columbia [20] datasets.

Our approach outperforms the baselines on the CASIA, Coverage, and Columbia datasets for the pixel-level AUC comparison. Especially on the Coverage dataset, our approach has 1.1% improvement compared to the second-best

Table 1: The pixel-level F1 score/AUC comparison on four standard datasets.

	Pixel-level		Forgery object-level		Dual branch					
					"AND"		"OR"		"MRF"	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
NIST′16 [16]	0.780	0.796	0.717	0.912	0.739	0.854	0.828	0.892	0.807	0.929
CASIA [17, 18]	0.751	0.713	0.688	0.844	0.744	0.743	0.742	0.851	0.789	0.862
Coverage [19]	0.551	0.723	0.433	0.802	0.531	0.808	0.561	0.828	0.563	0.828
Columbia [20]	0.831	0.641	0.398	0.756	0.529	0.809	0.776	0.803	0.833	0.801

Table 2: The pixel-level *F*1 score comparison on the standard datasets.

	NIST′16 [16]	CASIA [17, 18]	Coverage [19]	Columbia [20]
ELA [21]	0.236	0.470	0.222	0.214
NOI1 [22]	0.285	0.574	0.269	0.263
CFA1 [23]	0.174	0.467	0.190	0.207
MFCN [2]	0.571	0.612	-	0.541
RGB-N [7]	0.722	0.697	0.437	0.408
LSTM-encoder [5]	0.789	0.792	—	0.823
C2RNet [8]	0.55	0.676	—	0.695
Ours	0.807	0.789	0.563	0.833

Table 3: The pixel-level AUC comparison on the standard datasets.

	NIST′16 [16]	CASIA [17, 18]	Coverage [19]	Columbia [20]
ELA [21]	0.429	0.581	0.583	0.613
NOI1 [22]	0.487	0.546	0.578	0.612
CFA1 [23]	0.501	0.720	0.485	0.522
J-LSTM [24]	0.764	—	0.614	—
RGB-N [7]	0.937	0.858	0.817	0.795
LSTM-encoder [5]	0.794	—	0.712	—
Ours	0.929	0.862	0.828	0.801

result. Moreover, RGB-N [7] outperforms our approach on the NIST′16 dataset. This is mainly because the forged image quality is low in NIST′16. The pixel-level branch has inaccurate pixel classification, leading to poor boundary box regression. However, RGN-N only uses forgery object-level information and performs better on NIST′16. Therefore, the manipulation of object detection at different scales is the next point of study.

Visualization results: we illustrate some visualization results in Figures 6 and 7 in comparison with the pixel-level, forgery object-level, and the dual branch (MRF). Images are selected from Coverage and NIST′16. We illustrate better results in Figure 6. As we can see, our approach can detect image manipulation accurately. The two branches can correct for each other. The pixel-level branch can segment the forged objects from the bounding box detected by the forgery object-level branch, which makes the results finer-grained, such as Line 1 in Figure 6. Meanwhile, the forgery object-level branch shields the noise points of the pixel-level branch and obtains better performance, such as Line 4 in Figure 6.

Meanwhile, in Figure 7, we select a few poor cases, in which the pixel branch causes the result in the first row, and the forgery object-level branch causes the result in the second row. Similar to Line 1 in Figure 7, the forgery object-level branch detects the manipulated region precisely, but the pixel-level branch does not detect the whole manipulated region, which causes the poor result. Similarly, in Line 2, the pixel-level branch successfully detects part of the image manipulation area, but the forgery object-level branch does not detect the bounding box. Both situations lead to detection failures. We will balance the detection results of the two branches as much as possible in future work to obtain better experimental results.

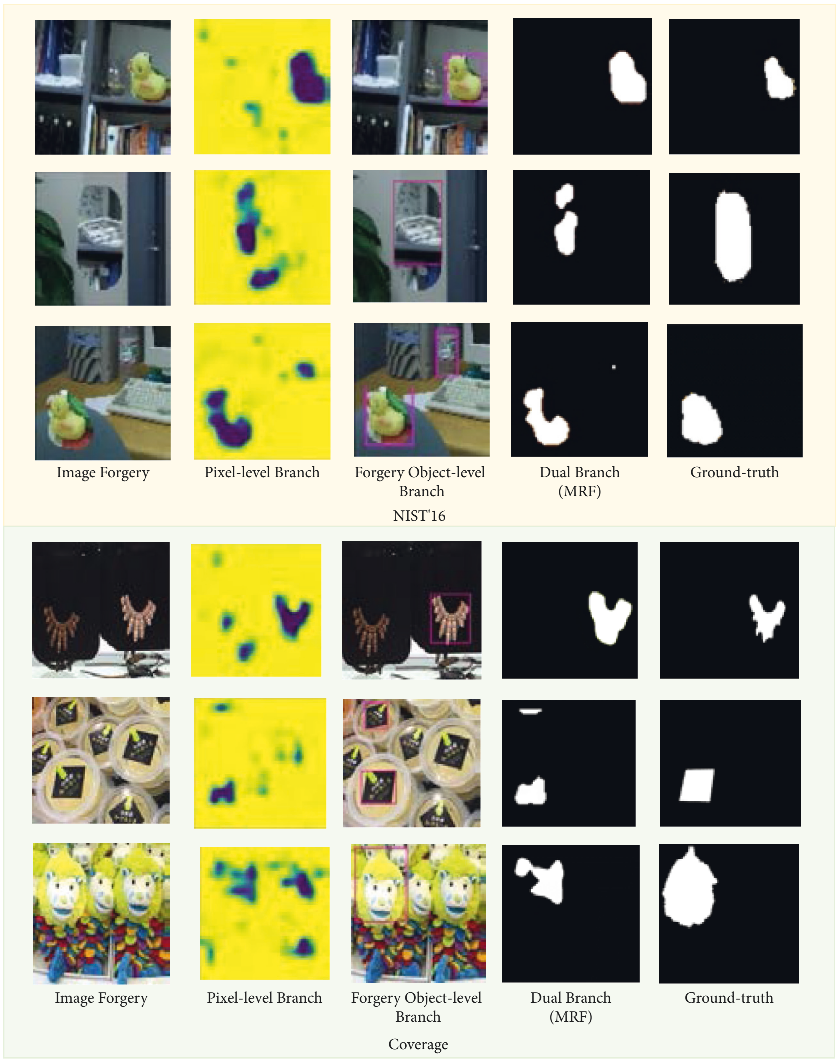

Figure 6: Visualization results from our approach.

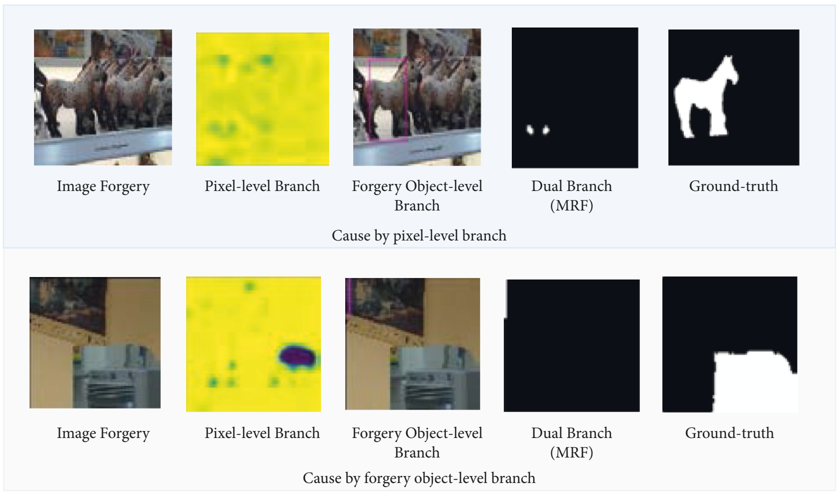

Figure 7: Poor visualization results from our approach.

5. Conclusion

We present a framework for image manipulation detection that combines a forgery object-level branch, a pixel-level branch, and two fusion modules. We utilize Faster R-CNN to detect manipulation areas on the forgery object-level branch. Meanwhile, we extract the resampling feature for each patch and utilize the Hilbert curve and LSTM network to detect the manipulated regions in the pixel-level branch. We fuse the two branches with the fusion modules and obtain a binary map of the manipulation regions. Experimental results show superior performance compared with the state-of-the-art methods. However, we also find that the two branches can affect each other when the manipulated object is of low quality. We will balance the two branches as much as possible in future work to obtain better experimental results.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Basic Scientific Research Operating Expenses of the Academy of Broadcasting Science, NRTA (No. JBKY2020006).

References

[1] Y. Zhang, J. Goh, L. L. Win, and V. Thing, "Image Region Forgery Detection: A Deep Learning Approach," *Proceedings of the Singapore Cyber-Security Conference (SG-CRC)*, vol. 2016, Singapore, 2016.

[2] R. Salloum, Y. Ren, and C.-C. Jay Kuo, "Image splicing localization using a multi-task fully convolutional network (MFCN)," *Journal of Visual Communication and Image Representation*, vol. 51, pp. 201–209, 2018.

[3] B. Chen, W. Tan, G. Coatrieux, Y. Zheng, and Y. -Q. Shi, "A serial image copy-move forgery localization scheme with source/target distinguishment," *IEEE Transactions on Multimedia*, vol. 23, pp. 3506–3517, 2020.

[4] J. Bunk, J. H. Bappy, T. M. Mohammed, A. Flenner, and B. S. Manjunath, "Detection and Localization of Image Forgeries Using Resampling Features and Deep learning," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1881–1889, IEEE, Honolulu, HI, USA, July 2017.

[5] J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury, "Hybrid LSTM and encoder-decoder architecture for detection of image forgeries," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3286–3300, 2019.

[6] J. Park, D. Cho, W. Ahn, and H. K. Lee, "Double JPEG detection in mixed JPEG quality factors using deep convolutional neural network Computer Vision (ECCV)," 2018.

[7] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,"pp. 1053–1061, Salt Lake City, UT, USA, June 2018.

[8] B. Xiao, Y. Wei, X. Bi, W. Li, and J. Ma, "Image splicing forgery detection combining coarse to refined convolutional neural network and adaptive clustering," *Information Sciences*, vol. 511, pp. 172–191, 2020.

[9] B. Chen, X. Liu, Y. Zheng, G. Zhao, and Y. -Q. Shi, "A robust GAN-generated face detection method based on dual-color spaces and an improved Xception," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 3506–3517, 2021.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[11] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Image Copy-Move Forgery Detection via an End-To-End Deep Neural network," in *Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1907–1915, IEEE, Lake Tahoe, NV, USA, March 2018.

[12] T. M. Mohammed, J. Bunk, L. Nataraj et al., "Boosting image forgery detection using resampling features and copy-move analysis," *Electronic Imaging*, vol. 30, no. 7, pp. 118-1–118-7, 2018.

[13] A. Mazumdar and P. K. Bora, "Two-stream Encoder-Decoder Network for Localizing Image Forgeries," 2020, https://arxiv.org/pdf/2009.12881.pdf.

[14] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 529–538, 2008.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] NIST, "Nist nimble 2016 datasets," 2016, https://www.nist.gov/itl/iad/mig/%20nimble-challenge-2017-evaluation/.

[17] J. Dong, W. Wang, and T. Tan, "Casia image tampering detection evaluation database," 2010, http://forensics.idealtest.org.

[18] J. Dong, W. Wang, and T. Tan, "Casia Image Tampering Detection Evaluation database," in *Proceedings of the 2013 IEEE China Summit and International Conference on Signal and Information Processing*, pp. 422–426, IEEE, Beijing, China, July 2013.

[19] B. Wen, Y. Zhu, R. Subramanian, T. Ng, X. Shen, and S. Winkler, "COVERAGE-A Novel Database for Copy-Move Forgery detection," in *Proceedings of the 2016IEEE International Conference on Image Processing (ICIP)*, pp. 161–165, IEEE, Phoenix, AZ, USA, September 2016.

[20] T. T. Ng, S. F. Chang, and Q. Sun, "A Data Set of Authentic and Spliced Image blocks," ADVENT Technical Report, pp. 203–2004, Columbia University, Columbia, USA, 2004.

[21] N. Krawetz and H. F. Solutions, "A picture's worth," *Hacker Factor Solutions*, vol. 6, no. 2, p. 2, 2007.

[22] B. Mahdian and S. Saic, "Using noise inconsistencies for blind image forensics," *Image and Vision Computing*, vol. 27, no. 10, pp. 1497–1503, 2009.

[23] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of CFA artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, 2012.

[24] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. S. Manjunath, "Exploiting spatial structure for localizing manipulated image regions," in *Proceedings of the IEEE international conference on computer vision*, pp. 4970–4979, Venice, Italy, October 2017.