HRGR: Enhancing Image Manipulation Detection via Hierarchical Region-aware Graph Reasoning

Xudong Wang[1], Jiaran Zhou[1], Huiyu Zhou[2], Junyu Dong[1], Yuezun Li[1,†]
[1] School of Computer Science and Technology, Ocean University of China
[2] School of Computing and Mathematical Sciences, University of Leicester

2025 IEEE International Conference on Multimedia and Expo (ICME) | 979-8-3315-9495-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICME59968.2025.11209303

***Abstract*—Image manipulation detection is to identify the authenticity of each pixel in images. One typical approach to uncover manipulation traces is to model image correlations. The previous methods commonly adopt the grids, which are fixed-size squares, as graph nodes to model correlations. However, these grids, being independent of image content, struggle to retain local content coherence, resulting in imprecise detection. To address this issue, we describe a new method named Hierarchical Region-aware Graph Reasoning (HRGR) to enhance image manipulation detection. Unlike existing grid-based methods, we model image correlations based on content-coherence feature regions with irregular shapes, generated by a novel Differentiable Feature Partition strategy. Then we construct a Hierarchical Region-aware Graph based on these regions within and across different feature layers. Subsequently, we describe a structural-agnostic graph reasoning strategy tailored for our graph to enhance the representation of nodes. Our method is fully differentiable and can seamlessly integrate into mainstream networks in an end-to-end manner, without requiring additional supervision. Extensive experiments demonstrate the effectiveness of our method in image manipulation detection, exhibiting its great potential as a plug-and-play component for existing architectures. Codes and models are available at https://github.com/OUC-VAS/HRGR-IMD.**

***Index Terms*—image manipualtion detection, feature clustering, graph reasoning**

I. Introduction

Image manipulation involves various techniques such as copy-move, splicing, and object removal that can subtly alter the primary content of images. As image editing tools and AI technology continue to advance, image manipulation has become effortless, leading to significant societal issues such as misinformation spread, ethical dilemmas, and potential legal problems [1].

To combat this, countermeasures have been proposed and gained increasing attention in recent years. Deep learning-based methods have emerged as the dominant method for detecting manipulations *e.g.*, [2]–[7],. Leveraging the impressive learning capabilities of deep learning models. Based on deep neural networks, feature enhancement strategies have been explored to further emphasize manipulation traces. Particularly, exploiting image correlations is a recent feature enhancement solution that has proven effective in enhancing detection ability [2], [3]. The intuition is that manipulated regions can be

This work is supported in part by the National Natural Science Foundation of China (No.62402464), Shandong Provincial Natural Science Foundation (No.ZR2024QF035), and China Postdoctoral Science Foundation (No.2021TQ0314; No.2021M703036).
† Corresponding author

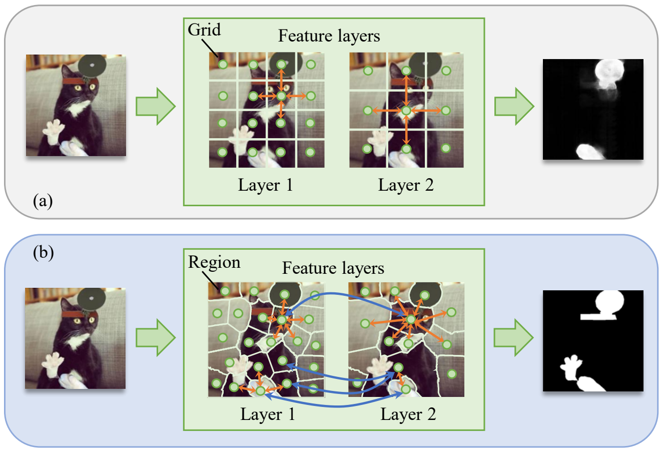


Fig. 1. The difference between (a) previous methods and (b) ours. Instead of using regular grids in previous methods, we model image correlations across different scales (orange and blue arrows) using connected local content-coherent feature regions.

better perceived when compared to pristine regions. To capture the correlations, these methods usually employ the mechanism of either attention mechanism [3], [4] or graphs [2], [8]. For example, PSCC-Net [3] proposed a spatial-channel correlation module to capture the correlations using attention. ObjectFormer [4] learned the correlation between intermediate features in objects and patches through cross-attention. HGCN [8] and ERMPC [2] sought the relationship between pristine and manipulated regions by building a graph structure.

Despite their promising results, these methods share a common limitation: They all rely on the grid-based modeling of correlations. It is important to note that these grids are regular squares with fixed sizes that are independent of the content. Typically, these methods partition the features into grids of size $n \times n$ and encapsulate the features within each grid as respective units for correlation modeling (see Fig. 1 (top)). However, as manipulations usually happen on the semantic-integrity objects (*e.g.*, copy-move/splicing a whole person), the use of grids can hardly retain the local content coherence. This limitation can potentially hinder their representation and subsequently restrict the extraction of manipulation traces (*e.g.*, leading to ambiguous detection boundaries (see Fig. 1 (top)).)

To overcome the limitation, we introduce a novel framework called *Hierarchical Region-aware Graph Reasoning (HRGR)* to enhance image manipulation detection. Instead of using regular grids to model correlations, we propose a Hierarchical Region-aware Graph that captures the correlations based on content-coherent **feature regions** (see Fig. 1 (bottom)).

To achieve this, we first describe a Differentiable Feature Partition strategy that divides the features into non-overlapped coherent regions while maintaining differentiability for end-to-end utilization. These feature regions are subsequently encapsulated as graph nodes using a Weighted Feature Aggregation strategy. In contrast to previous methods that typically construct graphs on single feature layers, we propose a Hierarchical Graph Modeling strategy to construct a graph within the single feature layers as well as across different feature layers. This is motivated by that the same manipulated instance at different scales should exhibit consistent manipulation traces. By modeling across different feature layers, we can identify this relation, thus further improving the identification of manipulation traces. After modeling the correlations, we describe a Structural-agnostic Graph Reasoning strategy to refine the representation of graph nodes, which are then mapped back to feature layers for feature enhancement. The overview of our method is illustrated in Fig. 2 (left).

It is noteworthy that our method is designed as a plug-and-play framework that can seamlessly integrate into mainstream networks without requiring additional supervision. Experiments are conducted on commonly used datasets, and the results demonstrate the efficacy of the proposed method.

The contribution of this paper can be summarized as follows: **1)** To the best of our knowledge, we are the first to explore image correlations based on content-coherent irregular feature regions rather than regular grids, facilitated by a novel Differentiable Feature Partition strategy. **2)** Leveraging the feature regions, we describe a new Hierarchical Region-aware Graph Reasoning strategy that seeks correlations within the same feature layer and across different feature layers. **3)** Our framework is fully differentiable and can seamlessly integrate into mainstream networks in an end-to-end manner without requiring additional supervision.

II. Method

A. Hierarchical Region-aware Graph

With feature maps, the gist of constructing this graph lies in two main steps: 1) formulating graph nodes and 2) modeling the correlations among these graph nodes. To ensure the end-to-end fashion of our method, accomplishing these two steps is not trivial. In the following, we first outline the overview of our method and then elaborate on the details of each step.

1) Differentiable Feature Partition: To formulate the graph nodes, we propose a new Differentiable Feature Partition (DFP) strategy to create feature regions. This strategy can partition the features into several non-overlapped and content-coherent regions in an unsupervised way, and can be integrated into the training in an end-to-end fashion.

Intuitively, the conventional clustering algorithms (*e.g.*, K-means) can be applied to feature layers by measuring the similarity among feature elements. However, this algorithm suffers from two difficulties. One is that local connectivity can not be guaranteed and the other is the clustering process is not differentiable. Inspired by the superpixel extraction methods [9], [10] and etc., we construct a soft feature element-region association matrix to make the clustering process differentiable.

Denote a feature map from the encoder $\mathcal{E}$ as $f \in \mathbb{R}^{h \times w \times c}$, where h, w, c are the corresponding number of feature height, width, and channel. To ensure region connectivity, we append the position (*i.e.*, x- and y- coordinates) of each feature element on its channel dimension as $f' \in \mathbb{R}^{h \times w \times c'}, c' = c + 2$. Note that the time cost in clustering is proportional to feature dimensions. To reduce the computational overhead while refining the features, we design a convolutional block ϕ that transforms the features $f'' = \phi(f'), f'' \in \mathbb{R}^{h \times w \times c''} (c'' < c')$. Based on f'', we initialize the region centers. Denote the number of regions as m. We evenly divide f'' into m grids, and average the features inside each grid as the initial representation of region centers. Let the region centers be a matrix $\mathbf{R}'' = [r''_1; ...; r''_m] \in \mathbb{R}^{m \times c''}$. Each region center r''_i has a size of $1 \times c''$. For clustering, we build the feature element-region association matrix to store the probability of assigning a feature element to a region, denoted as $\mathbf{D} \in \mathbb{R}^{n \times m}, n = h \times w$.

For the i-th feature element $e''_i \in f''$, we calculate the Euclidean distance between it and other region centers in $\mathbf{R}''$ and perform a softmax operation to normalize the probability, as

$$\mathbf{D}(i, j) = \frac{\exp^{-\|e''_i - r''_j\|^2}}{\sum_{j=1}^{m} \exp^{-\|e''_i - r''_j\|^2}}. \tag{1}$$

After obtaining the matrix $\mathbf{D}$, the intuitive way of obtaining regions is to assign each element to its nearest center, as

$$\mathbf{D}(i, j) = \begin{cases} 1 & \text{if} \quad j = \underset{j \in \{1,...,m\}}{\arg\max} \left(\mathbf{D}(i, j)\right), \\ 0 & \text{otherwise}. \end{cases} \tag{2}$$

Then the region center $r''_j \in \mathbf{R}''$ can be updated by

$$r''_j = \frac{\sum_{i=1}^{n} e''_i \cdot \mathbf{D}(i, j)}{\sum_{i=1}^{n} \mathbf{D}(i, j)}. \tag{3}$$

However, this assignment of Eq. (2) is not differentiable due to the `argmax` operation, which thus can not be incorporated into our model.

🔊: *"How to make this process differentiable?"*

To address this issue, we use soft assignments, *i.e.*, an element can be assigned to all regions with various probabilities. Specifically, we abandon Eq. (2) and update the region centers using the probabilities in matrix $\mathbf{D}$, directly following Eq. (3). The update process can be formulated as the operations between matrices as

$$\begin{aligned} &\mathbf{F}'' \in \mathbb{R}^{n \times c''} \leftarrow \text{Reshape}(f'' \in \mathbb{R}^{h \times w \times c''}), \\ &\mathbf{R}'' = \frac{1}{\mathbf{N}_\text{D}} \circ (\mathbf{D}^\top \mathbf{F}''), \end{aligned} \tag{4}$$

where $\mathbf{N}_\text{D} = [\sum_{i=1}^{n} \mathbf{D}(i, 1), ..., \sum_{i=1}^{n} \mathbf{D}(i, m)]$ is the summation of each column of $\mathbf{D}$, $\circ$ denotes the element-wise product.

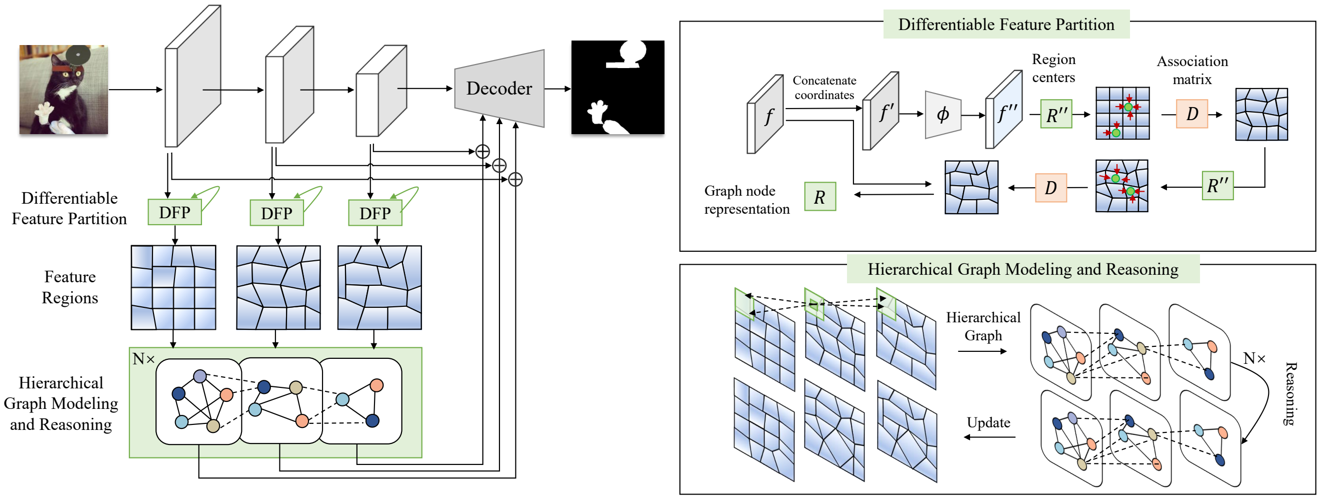


Fig. 2. Overview of our method. We first extract the feature regions on feature layers from the encoder using a Differentiable Feature Partition (DFP) strategy. Then we build the hierarchical region-aware graph over all features and perform graph reasoning to update the features. The process of Differentiable Feature Partition and Hierarchical Graph Reasoning and Reasoning are shown in the right part.

Note the feature regions are updated iteratively. Denote the current iteration as t. Once the region centers $\mathbf{R}''_t$ are updated, we can re-calculate the matrix $\mathbf{D}_{t+1}$ and then update $\mathbf{R}''_{t+1}$. This process is repeated until the maximum iteration T is reached. The overview of this process is shown in Fig. 2 (right-top).

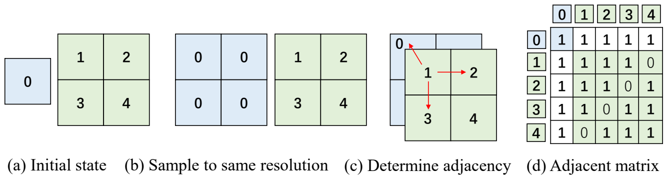
2) Weighted Feature Aggregation for Node Representation: After obtaining the feature regions, we encapsulate the feature elements inside regions as the representation of graph nodes using a Weighted Feature Aggregation strategy. Denote the matrix of node representations as $\mathbf{R} = [r_i; ...; r_m] \in \mathbb{R}^{m\times c}$. The encapsulation process can be defined with similar formulas as Eq. (4), with the difference that here the encapsulation is applied to the original encoder features f, as

$$\begin{aligned}\mathbf{F} &\in \mathbb{R}^{n\times c} \leftarrow \text{Reshape}(f \in \mathbb{R}^{h\times w\times c}),\\ \mathbf{R} &= \frac{1}{\mathbf{N}_{\text{D}}} \circ (\mathbf{D}^{\top}\mathbf{F}).\end{aligned} \tag{5}$$

3) Hierarchical Graph Modeling: With the obtained graph nodes, we propose a Hierarchical Graph Modeling strategy to construct a graph on feature regions within the same and across different feature layers. Denote a set of feature maps from different layers as $\{f^1, ..., f^k\}$. We first obtain the feature element-region matrix for each feature as $\{\mathbf{D}^1, ..., \mathbf{D}^k\}$ and their corresponding node representations as $\{\mathbf{R}^1, ..., \mathbf{R}^k\}$. Denote the adjacent matrix for this graph as $\mathbf{A} \in \mathbb{R}^{M\times M}$, where M is the total number of nodes, *i.e.*, $M = \sum_{i=1}^{k} |\mathbf{R}^i|$ (see Fig. 3 for illustration).

🔊: *"How to seek the hierarchical neighborhood relationships?"*

Since the regions corresponding to nodes are irregular, modeling the relationship is more challenging than the conventional grid-based methods. As such, we describe a 3D sliding-window based strategy to seek the correlations. It is noteworthy that in this step, we require an assignment



	0	1	2	3	4
0	1	1	1	1	1
1	1	1	1	1	0
2	1	1	1	0	1
3	1	1	0	1	1
4	1	0	1	1	1

Fig. 3. Illustration of adjacent matrix $\mathbf{A}$. We build a hierarchical graph that establishes adjacency relationships within the same layer (intra-layer, orange arrows in Fig. 1 (b)) and also across different layers (inter-layer, blue arrows in Fig. 1 (b)), resulting in a single adjacent matrix $\mathbf{A}$. For simplicity, we use an example with two layers (highlighted in blue with one node and green with four nodes, see Fig 3 (a)). First, we resample them to the same resolution (see Fig 3 (b)). Then we stack them along channel dimension and apply the 3D sliding window strategy to obtain an adjacency matrix (see Fig 3 (c,d)).

of feature elements to regions. Therefore, we reuse Eq. (2) to divide each feature map into non-overlapped regions and assign each region a unique index from $\{1, ..., M\}$. Then we can create the region index maps $\{J^1, ..., J^k\}$ corresponding to $\{f^1, ..., f^k\}$. We resize each index map to the same spatial size and concatenate them as a matrix $\mathbf{J} \in \mathbb{R}^{h\times w\times k}$. Then we perform the 3D sliding window with a size of $3 \times 3 \times 3$ in order. The criterion is that $\mathbf{A}(\mathbf{J}(p_0), \mathbf{J}(p_0 + p_n)) = 1$, if $\mathbf{J}(p_0) \neq \mathbf{J}(p_0 + p_n)$, where p_0 is the position of the window center and p_n is the offset within window. To accelerate this process, we formulate it into CUDA code.

Besides the adjacent matrix $\mathbf{A}$, we construct the node representation matrix for all regions. Since $\mathbf{R}^1, ..., \mathbf{R}^k$ has different channel numbers, we perform a linear transform for each matrix to uniform their channel number as C. Denote the node representation matrix as $\mathbf{R}_{\text{G}} \in \mathbb{R}^{M\times C}$, which is obtained by

$$\mathbf{R}_{\text{G}} = [\mathbf{R}_{\text{G}}^1; ...; \mathbf{R}_{\text{G}}^k] = [\mathbf{R}^1\mathbf{W}^1; ...; \mathbf{R}^k\mathbf{W}^k], \tag{6}$$

where $\mathbf{W}^1, ..., \mathbf{W}^k$ are learnable linear matrics.

B. Structure-agnostic Graph Reasoning

In the design of our method, the feature regions can be dynamically updated during training, which subsequently results in different graph structures. To process the dynamic graphs, we adapt structure-agnostic graph reasoning strategies [11] and etc. to our method. Generally, this strategy consists of two steps: aggregation and update. The aggregation step gathers the information from surrounding nodes and the update step transforms the node representations based on the aggregated information with a learnable weight matrix.

For each node, we simply average the representations of surrounding nodes and employ linear transformation to update the current node representations. Since both of the steps are independent of specific graph structures, our method can be applied even if node adjacency and the number of nodes vary. Denote the learnable linear transformation matrix as $\mathbf{W_G} \in \mathbb{R}^{C\times C}$. The message passing process can be formulated as

$$\hat{\mathbf{R}}_{\mathrm{G}} = \left(\frac{1}{\mathbf{N_A}} \circ (\mathbf{A}\mathbf{R}_{\mathrm{G}})\right)\mathbf{W}_{\mathrm{G}}, \tag{7}$$

where $\mathbf{N_A} = [\sum_{i=1}^{M}\mathbf{A}(1,i);...;\sum_{i=1}^{M}\mathbf{A}(M,i)]$ is the summation of each row of $\mathbf{A}$.

To mitigate the over-smoothing effect, we employ a regularization on $\hat{\mathbf{R}}$ following [12], which is defined as

$$\mathbf{R}_{\mathrm{G}} = \sigma(\hat{\mathbf{R}}_{\mathrm{G}}\mathbf{W}_{\alpha})\mathbf{W}_{\beta} + \hat{\mathbf{R}}_{\mathrm{G}}, \tag{8}$$

where $\mathbf{W}_{\alpha}, \mathbf{W}_{\beta}$ are two learnable linear matrics and σ is an activation function (*e.g.*, GeLu).

Note that this process is only a prototype version to demonstrate the effectiveness of the proposed hierarchical region-aware graph. It can be upgraded with other advanced graph reasoning approaches, which may result in additional improvement.

After graph reasoning, we remap the updated representations back to the feature elements for feature enhancement. The essence is to revert Eq. (5). For the feature f^i, the inverse mapping can be defined as

$$\mathbf{F}^i_{\mathrm{G}} = \mathbf{D}^i\mathbf{R}^i, \quad \mathbf{R}^i = \mathbf{R}^i_{\mathrm{G}}\mathbf{W}^i_{\mathrm{inv}}, \tag{9}$$

where $\mathbf{R}^i_{\mathrm{G}}$ denotes the updated representations of graph nodes at i-th feature layer, $\mathbf{W}^i_{\mathrm{inv}}$ is a linear matrix to match the channel number of f^i, $\mathbf{D}^i$ is the feature element-region association matrix at i-th feature layer. Then we integrate $\mathbf{F}^i_{\mathrm{G}}$ with the features f^i as

$$f^i_{\mathrm{G}} \leftarrow \mathrm{Reshape}(\mathbf{F}^i_{\mathrm{G}}), f^i = \mu f^i_{\mathrm{G}} + f^i, \tag{10}$$

where μ is a learnable parameter that controls the feature integration.

C. Objective Functions

Note that the proposed method does not require extra supervision. Thus we only employ a task-specific loss function for training. Considering that the area of the manipulated regions in images is usually less than the authentic surroundings, we employ focal loss [13] for balancing the importance of authentic and manipulated pixels, which is defined as

$$\begin{aligned}\mathbf{L}(\boldsymbol{y},\hat{\boldsymbol{y}}) = -\sum &(\alpha\cdot(1-\boldsymbol{y})^{\gamma}\cdot\hat{\boldsymbol{y}}\log(\boldsymbol{y}) \\ &+(1-\alpha)\boldsymbol{y}^{\gamma}(1-\hat{\boldsymbol{y}})\log(1-\boldsymbol{y}))\,,\end{aligned} \tag{11}$$

where $\hat{\boldsymbol{y}}$ represents the ground truth mask, α is the weight for balancing, and γ is the difficulty adjustment factor. The larger value of γ indicates a focus on more challenging samples.

III. Experiments

A. Experimental Settings

Datasets. Following previous methods, we evaluate our method on five datasets, CASIA [14], Coverage [15], NIST16 [16], Columbia [17], and IMD20 [18]. The CASIA dataset contains 921 images in CASIAv1 and 5123 images in CASIAv2. We use CASIAv2 for training and CASIAv1 for testing. The coverage dataset comprises 100 images, with 75 images for training and 25 for testing. NIST16 has 564 images, where 404 images for training and 160 images for testing. The Columbia dataset consists of 180 images, and IMD20 dataset consists of 2010 images. Note that these two datasets do not provide training sets. Thus we only use them for testing.

Evaluation Metrics. Following previous work [2]–[4], [19], we utilize the widely used F1 score for evaluation. The F1 score is calculated using the Equal Error Rate (EER) as the threshold. Moreover, we also employ pixel-level Area Under Curve (AUC) as complementary.

Implementation Details. Our method is implemented by PyTorch 11.3 with a Nvidia 3090 GPU. In the training phase, we utilize the AdamW optimizer with a learning rate of 6×10^{-5} and a weight decay of 0.05. We employ a linear warm-up learning strategy, where the learning rate is linearly increased during warm-up and then linearly decreased based on the iteration until it reaches 0. The initial learning rate for warm-up is set to 1×10^{-6}, and warm-up is completed within $1.5k$ iterations. The training image size is set to 512×512 and the training iterations are set to $160k$. For feature integration, each layer's μ_i is learned separately with initialization of 1. In the objective function, we set the hyperparameters as follows: $\alpha = 0.5, \gamma = 2$. We employ a plain feature pyramid detector [20] as the decoder. The detection performance can be further improved if a more advanced decoder is employed.

B. Comparing with State-of-the-arts

Following previous works [3], [19], we explore the Fine-tuned scenario, that firstly pre-trains the methods using synthetic dataset in [3] and fine-tunes the pre-trained model on the training set of each dataset. Then we evaluate the performance on its testing set. This scenario represents the capacity to capture manipulation traces in a specific dataset.

Following previous methods, the comparison is conducted on CASIA, Coverage, and NIST16 datasets since they provide training sets. Table I shows the results of our method compared to other state-of-the-art methods and graph-based methods under the fine-tuned scenario. We would like to clarify that EVP and NCL were trained without using synthetic dataset as others, while TransF. was tested on the Coverage dataset without fine-tuning. These three methods are marked by $(*)$. Moreover, the methods marked with $(\dagger)$ are the ones that we have reproduced. Specifically, PSCC-Net and HiFi-Net

TABLE I
COMPARISON OF DIFFERENT METHODS. THE **BEST** RESULT IS HIGHLIGHTED IN BOLD WHILE THE SECOND-BEST RESULT IS MARKED BY UNDERLINING.

Method	CASIA		Coverage		NIST16		avg	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
HGCN [8]	-	40.8	-	39.3	-	71.0	-	50.4
PSCC-Net [3]	87.5	55.4	94.1	72.3	99.1	74.2	93.7	69.8
ObjFormer [4]	88.2	57.9	95.7	75.8	99.6	82.4	94.5	72.0
EVP* [21]	86.2	63.6	-	-	-	-	86.2	63.6
PCL [22]	75.1	46.7	91.7	62.0	94.6	78.0	87.1	62.2
NCL* [23]	86.4	59.8	92.8	80.1	91.2	83.1	90.1	74.3
HiFi-Net [19]	88.5	61.6	96.1	80.1	98.9	85.0	94.6	75.5
ERMPC [2]	90.4	58.6	**98.4**	77.3	**99.7**	83.6	**96.1**	73.1
ACBG. [6]	-	66.9	-	71.1	-	**93.3**	-	77.1
PSCC-Net† [3]	85.2	53.6	91.1	71.6	96.4	70.1	90.3	65.1
HiFi-Net† [19]	85.9	59.0	86.2	63.4	87.4	62.8	86.5	61.7
CAT-Net† [5]	77.3	40.6	79.2	47.8	89.2	61.5	81.9	50.0
MVSS-Net† [24]	86.4	60.2	92.7	76.3	96.1	69.3	91.7	68.6
ObjFormer† [4]	71.9	42.6	68.4	41.3	91.4	68.2	77.2	50.7
TruFor† [7]	92.4	67.2	96.2	79.0	98.6	82.6	95.7	76.2
Res-50 [25]	87.0	54.0	90.5	69.8	97.1	64.7	91.5	62.8
+ HRGR	88.5	56.2	92.7	71.3	97.2	64.1	92.8	63.9
HRNet [26]	90.1	62.2	90.3	68.6	97.4	66.5	92.6	65.8
+ HRGR	91.4	64.2	90.9	69.5	98.4	67.1	93.6	66.9
EffNet-b4 [27]	92.5	66.2	93.3	76.1	97.5	61.7	94.4	68.0
+ HRGR	92.9	66.7	93.9	74.8	97.2	63.4	94.7	68.3
Intern-tiny [28]	91.4	66.0	94.5	78.7	99.3	79.3	95.1	74.7
+ HRGR	**93.4**	**69.9**	94.8	**82.1**	99.3	84.6	95.8	**78.9**

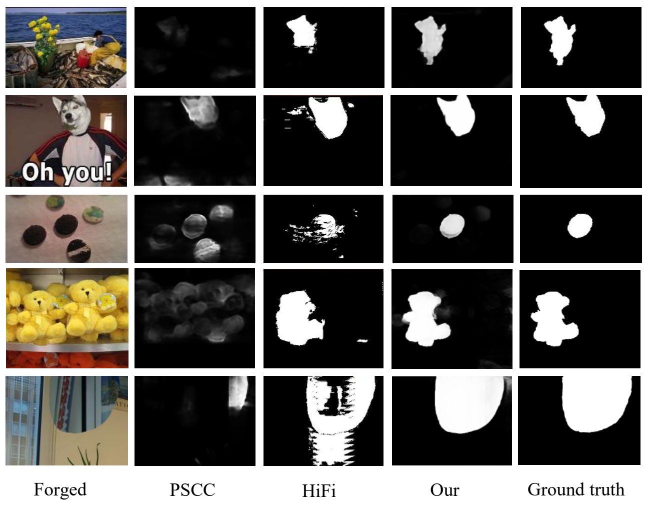


Fig. 4. Qualitative visualization on CASIA, IMD20, NIST16, Coverage, and Columbia datasets (from top to bottom).

were reproduced using the code provided by the respective authors, while the remaining methods were reproduced with IMDLBenCo [29].

We can observe that our method improves the performance on these base networks by 0.8% in AUC and 1.7% in the F1 score on average, demonstrating the efficacy of our method to improve the capacity of learning specific manipulation traces. Compared to state-of-the-arts, our method achieves an average F1 score of 78.9% across the three datasets, surpassing the highest-performing HiFi-Net by 3.3%. Particularly on the CASIA dataset, our method outperforms HiFi-Net with a 3.0% improvement in the AUC score and a 8.3% increase in the F1 score. These results demonstrate the superiority of the proposed method in capturing subtle manipulation traces. Moreover, our method only introduces a few additional parameters (1.45M) and FLOPS (1.49G). Fig. 4 shows several visualizations of different methods on five datasets.

TABLE II
ABLATION STUDY OF DIFFERENT SETTINGS ON CASIA DATASET.

Method	w/o Fine-tune		w/ Fine-tune	
	AUC	F1	AUC	F1
Baseline	84.1	48.3	91.4	66.0
Feature region $\rightarrow$ Grid	85.1	49.6	92.1	67.0
Graph reasoning $\rightarrow$ Linear	85.8	50.8	92.1	67.3
Graph reasoning $\rightarrow$ MHSA	85.3	49.1	91.7	66.8
Hierarchical $\rightarrow$ Layer-wise	85.6	50.6	92.5	68.0
DFP w/o x- and y- coords	85.4	49.5	93.3	68.9
Full	**86.3**	**52.4**	**93.4**	**69.9**

C. Analysis

Ablation Study. This part investigates the effect of different settings: 1) **Baseline** indicates solely using Intern-tiny network. 2) **Feature region** $\rightarrow$ **Grid** indicates replacing the feature partition module with the original regular grid. 3) **Graph reasoning** $\rightarrow$ **Linear** refers to replacing graph reasoning with several linear layers (equivalent to fully connected graph reasoning). 4) **Graph reasoning** $\rightarrow$ **MHSA** refers to replacing graph reasoning with several Mutil-Head Self Attention layers. 5) **Hierarchical** $\rightarrow$ **Layer-wise** means replacing the hierarchical graph modeling with layer-wise independent graph modeling and performing graph reasoning. 6) **DFP w/o x- and y- coords** means that we do not additionally concatenate the x and y coordinates in the clustering process of the DFP module. Table II displays the results of ablation experiments. It can be observed that when any module is replaced, the performance drops significantly, *e.g.*, F1 scores decreasing by 2.9%, 2.6%, 3.1%, 1.9%, 1.0% in the finetune stage. Nevertheless, only replacing one component still has an improvement compared to the baseline, which confirms the effectiveness of each component.

Performance without Fine-tune. This scenario represents that the methods are pre-trained on synthetic images and directly tested on other datasets, aiming to exhibit the generalizability of detection. Table III shows the performance of the different base networks before and after using our method HRGR. These methods are pre-trained on the synthetic dataset from [3] and directly tested on all datasets. For a fair comparison, we use the same configuration for each experimental group. It can be seen that the performance of base networks generally improves after using our method.

Robustness. To evaluate the robustness of our method, we follow the perturbation settings in [3], which degrades manipulated images obtained from CASIA under different preprocesses, including resizing images with different scales, Gaussian Blurring with kernel size, adding Gaussian Noise with standard deviation “Sigma”, and performing JPEG Compression with quality factor. The evaluation results in comparison

TABLE III
PERFORMANCE WITHOUT FINE-TUNE.

Base network	CASIA		Coverage		NIST16		Columbia		IMD20		avg	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
ResNet-50	82.0	41.2	78.3	42.5	80.6	28.1	**92.8**	**80.3**	78.6	26.3	82.4	43.6
+ HRGR	**82.3**	**42.6**	**78.9**	**43.0**	**80.7**	**29.0**	92.1	80.1	**79.2**	**27.1**	**82.6**	**44.3**
HRNet	82.7	44.9	80.3	45.6	81.7	31.5	95.1	87.2	79.6	28.4	83.8	47.5
+ HRGR	**84.0**	**46.3**	**81.0**	**46.0**	**82.3**	**34.6**	**95.7**	**88.2**	**80.4**	**29.9**	**84.6**	**49.0**
EffNet-b4	82.1	43.3	74.3	35.6	**83.2**	30.1	85.8	68.5	**81.3**	31.5	81.3	41.8
+ HRGR	**83.5**	**44.9**	**75.8**	**37.8**	81.9	**32.2**	**86.4**	**68.7**	80.9	**31.7**	**81.7**	**43.0**
Intern-tiny	84.1	48.3	80.3	49.3	**86.6**	39.3	94.7	88.8	79.5	29.5	85.0	51.0
+ HRGR	**86.3**	**52.4**	**83.9**	**54.7**	86.2	**39.4**	**96.4**	**91.2**	**79.8**	**30.4**	**86.5**	**53.6**

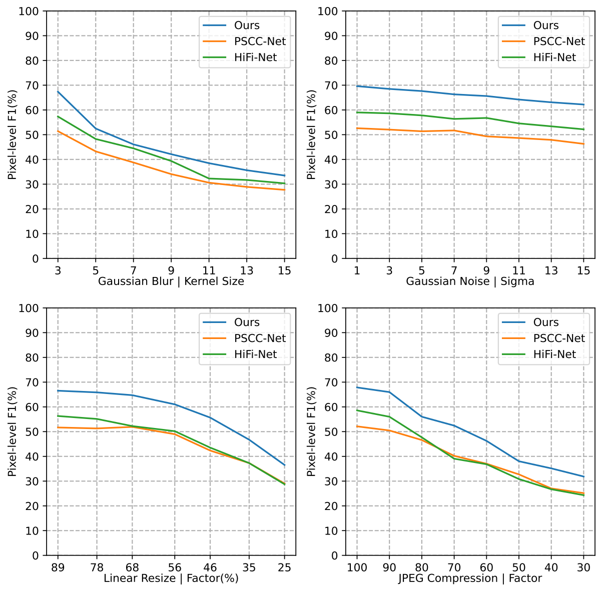


Fig. 5. Robustness analysis under various perturbations on CASIA dataset.

to PSCC-Net and HiFi-Net are shown in Fig. 5, which reveals that our method can resist certain perturbations.

IV. Conclusion

This paper introduces a new method, Hierarchical Region-aware Graph Reasoning (HRGR), to enhance image manipulation detection. Specifically, we describe a differentiable feature partition strategy to formulate feature regions and construct a hierarchical region-aware graph on these regions. Then we introduce a structure-agnostic graph reasoning to improve the feature effectiveness. These processes are iteratively updated during training, leading to mutual boosting. Extensive experiments demonstrate the effectiveness of our method and its potential as a plug-and-play component for classical models.

References

[1] Rahul Thakur and Rajesh Rohilla, "Recent advances in digital image manipulation detection techniques: A brief review," *Forensic science international*, 2020. 1

[2] Dong Li, Jiaying Zhu, Menglu Wang, Jiawei Liu, Xueyang Fu, and Zheng-Jun Zha, "Edge-aware regional message passing controller for image forgery localization," in *CVPR*, 2023. 1, 4, 5

[3] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu, "Pscc-net: Progressive spatio-channel correlation network for image manipulation detection and localization.," *TCSTV*, 2022. 1, 4, 5

[4] Junke Wang, Zuxuan Wu, Jingjing Chen, Xintong Han, Abhinav Shrivastava, Ser-Nam Lim, and Yu-Gang Jiang, "Objectformer for image manipulation detection and localization," in *CVPR*, 2022. 1, 4, 5

[5] Myung-Joon Kwon, Seung-Hun Nam, In-Jae Yu, Heung-Kyu Lee, and Changick Kim, "Learning jpeg compression artifacts for image manipulation detection and localization," *IJCV*, 2022. 1, 5

[6] Wenxi Liu, Hao Zhang, Xinyang Lin, Qing Zhang, Qi Li, Xiaoxiang Liu, and Ying Cao, "Attentive and contrastive image manipulation localization with boundary guidance," *TIFS*, 2024. 1, 5

[7] Fabrizio Guillaro, Davide Cozzolino, Avneesh Sud, Nicholas Dufour, and Luisa Verdoliva, "Trufor: Leveraging all-round clues for trustworthy image forgery detection and localization," in *CVPR*, 2023. 1, 5

[8] Wenyan Pan, Zhili Zhou, Miaogen Ling, Xin Geng, and QM Wu, "Learning hierarchical graph representation for image manipulation detection," *arXiv preprint arXiv:2201.05730*, 2022. 1, 5

[9] Fengting Yang, Qian Sun, Hailin Jin, and Zihan Zhou, "Superpixel segmentation with fully convolutional networks," in *CVPR*, 2020. 2

[10] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz, "Superpixel sampling networks," in *ECCV*, 2018. 2

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, 2017. 4

[12] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu, "Vision gnn: An image is worth graph of nodes," *Advances in Neural Information Processing Systems*, 2022. 4

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *ICCV*, 2017. 4

[14] Jing Dong, Wei Wang, and Tieniu Tan, "Casia image tampering detection evaluation database," in *ChinaSIP*, 2013. 4

[15] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler, "Coverage — a novel database for copy-move forgery detection," in *ICIP*, 2016. 4

[16] HY Guan, YY Lee, A Yates, A Delgado, D Zhou, D Joy, and A Pereira, "Nist nimble 2016 datasets," 2016. 4

[17] Tian-Tsong Ng, Jessie Hsu, and Shih-Fu Chang, "Columbia image splicing detection evaluation dataset," *DVMM lab. Columbia Univ CalPhotos Digit Libr*, 2009. 4

[18] Adam Novozamsky, Babak Mahdian, and Stanislav Saic, "Imd2020: A large-scale annotated dataset tailored for detecting manipulated images," in *WACVW*, 2020. 4

[19] Xiao Guo, Xiaohong Liu, Zhiyuan Ren, Steven Grosz, Iacopo Masi, and Xiaoming Liu, "Hierarchical fine-grained image forgery detection and localization," in *CVPR*, 2023. 4, 5

[20] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun, "Unified perceptual parsing for scene understanding," in *ECCV*, 2018. 4

[21] Weihuang Liu, Xi Shen, Chi-Man Pun, and Xiaodong Cun, "Explicit visual prompting for low-level structure segmentations," in *CVPR*, 2023. 5

[22] Yuyuan Zeng, Bowen Zhao, Shanzhao Qiu, Tao Dai, and Shu-Tao Xia, "Towards effective image manipulation detection with proposal contrastive learning," *TCSVT*, 2023. 5

[23] Jizhe Zhou, Xiaochen Ma, Xia Du, Ahmed Y Alhammadi, and Wentao Feng, "Pre-training-free image manipulation localization through non-mutually exclusive contrastive learning," in *ICCV*, 2023. 5

[24] Chengbo Dong, Xinru Chen, Ruohan Hu, Juan Cao, and Xirong Li, "Mvss-net: Multi-view multi-scale supervised networks for image manipulation detection," *TPAMI*, 2022. 5

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 5

[26] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al., "Deep high-resolution representation learning for visual recognition," *TPAMI*, 2020. 5

[27] Mingxing Tan and Quoc Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019. 5

[28] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, and Hongsheng Li, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," in *CVPR*, 2023. 5

[29] Xiaochen Ma, Xuekang Zhu, Lei Su, Bo Du, Zhuohang Jiang, Bingkui Tong, Zeyu Lei, Xinyu Yang, Chi-Man Pun, Jiancheng Lv, and Jizhe Zhou, "Imdl-benco: A comprehensive benchmark and codebase for image manipulation detection and localization," in *NeurIPS*, 2024. 5