ObjectFormer for Image Manipulation Detection and Localization

Junke Wang[1,2], Zuxuan Wu[1,2 †], Jingjing Chen[1,2], Xintong Han[3],
Abhinav Shrivastava[4], Ser-Nam Lim[5], Yu-Gang Jiang[1,2]
[1]Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University
[2]Shanghai Collaborative Innovation Center on Intelligent Visual Computing
[3]Huya Inc, [4]University of Maryland, [5]Meta AI

Abstract

Recent advances in image editing techniques have posed serious challenges to the trustworthiness of multimedia data, which drives the research of image tampering detection. In this paper, we propose ObjectFormer to detect and localize image manipulations. To capture subtle manipulation traces that are no longer visible in the RGB domain, we extract high-frequency features of the images and combine them with RGB features as multimodal patch embeddings. Additionally, we use a set of learnable object prototypes as mid-level representations to model the object-level consistencies among different regions, which are further used to refine patch embeddings to capture the patch-level consistencies. We conduct extensive experiments on various datasets and the results verify the effectiveness of the proposed method, outperforming state-of-the-art tampering detection and localization methods.

1. Introduction

With the rapid development of deep generative models like GANs [13, 25, 47] and VAEs [18, 33], a multitude of image editing applications have become widely accessible to the public [10,20,29,38]. These editing tools make it easy and effective to produce photo-realistic images and videos that could be used for entertainment, interactive design, *etc*., which otherwise requires professional skills. However, there are growing concerns on the abuse of editing techniques to manipulate image and video content for malicious purposes. Therefore, it is crucial to develop effective image manipulation detection methods to examine whether images have been modified or not and identify regions in images that have been modified.

Image manipulation techniques can be generally classified into three types: (1) splicing methods which copy regions from one image and paste to other images, (2) copy-move which shifts the spatial locations of objects within images, and (3) removal methods which erase regions from images and inpaint missing regions with visually plausible contents. As shown in Figure 1, to produce semantically meaningful and perceptually convincing images, these approaches oftentimes manipulate images at the object-level, *i.e*., adding/removing objects in images. While there are some recent studies focusing on image manipulation detection [16, 43, 46], they typically use CNNs to directly map input images to binary labels (*i.e*., authentic/manipulated) without explicitly modeling object-level representations. In contrast, we posit that **image manipulation detection should not only examine whether certain pixels are out of distribution, but also consider whether objects are consistent with each other**. In addition, visual artifacts brought by image editing that are no longer perceptible in the RGB domain are oftentimes noticeable in the frequency domain [6, 31, 39]. This demands a multimodal approach that jointly models the RGB domain and the frequency domain to discover subtle manipulation traces.

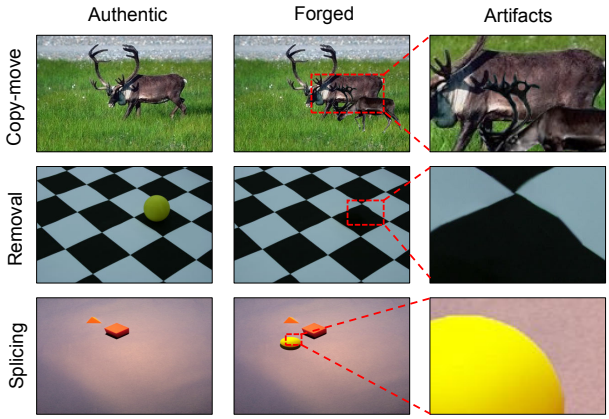

Figure 1. Tampering images usually contain manipulated objects. Thus, exploiting object-level consistency is crucial for manipulation detection.

In this paper, we introduce ObjectFormer, a multimodal transformer framework for image manipulation detection

†Corresponding author.

and localization. ObjectFormer builds upon transformers due to their impressive performance on a variety of vision tasks like image classification [12, 15, 24], object detection [5,49], video classification [4,23,40,41], *etc*. More importantly, transformers are natural choices to model whether patches/pixels are consistent in images, given that they explore the correlations between different spatial locations using self-attention. Inspired by object queries that are automatically learned [1, 49], we use a set of learnable parameters as object prototypes (serving as mid-level object representations) to discover the object-level consistencies, which are further leveraged to refine the patch embeddings for patch-level consistencies modeling.

With this in mind, ObjectFormer first converts an image from the RGB domain to the frequency domain using Discrete Cosine Transform and then extracts multimodal patch embeddings with a few convolutional layers. The RGB patch embeddings and the frequency patch embeddings are further concatenated to complement each other. Further, we use a set of learnable embeddings as object queries/prototypes, interacting with the derived patch embeddings to learn consistencies among different objects. We refine patch embeddings with these object prototypes with cross-attention. By iteratively doing so, ObjectFormer derives global feature representations that explicitly encode mid-level object features, which can be readily used to detect manipulation artifacts. Finally, the global features are used to predict whether images have been modified and the corresponding manipulation mask in a multi-task fashion. The framework can be trained in an end-to-end manner. We conduct experiments on commonly used image tampering datasets, including CASIA [11], Columbia [35], Coverage [42], NIST16 [27], and IMD20 [28]. The results demonstrate that ObjectFormer outperforms state-of-the-art tampering detection and localization methods. In summary, our work makes the following key contributions:

- We introduce ObjectFormer, an end-to-end multimodal framework for image manipulation detection and localization, combining RGB features and frequency features to identify the tampering artifacts.
- We explicitly leverage learnable object prototypes as mid-level representations to model object-level consistencies and refined patch embeddings to capture patch-level consistencies.
- We conduct extensive experiments on multiple benchmarks and demonstrate that our method achieves state-of-the-art detection and localization performance.

2. Related Work

Image Manipulation Detection / Localization Most early studies focus on detecting a specific type of manipulation, *e.g*., splicing [8, 17, 19], copy-move [7, 32], and removal [48]. However, in real-world scenarios, the exact manipulation type is unknown, which motivates a line of work focusing on general manipulation detection [3, 16, 43]. In addition, RGB-N [46] introduces a two-stream network for manipulation localization, where one stream extracts RGB features to capture visual artifacts, and the other stream leverages noise features to model the inconsistencies between tampered and untouched regions. SPAN [16] models the relationships of pixels within image patches on multiple scales through a pyramid structure of local self-attention blocks. PSCCNet [22] extracts hierarchical features with a top-down path and detects whether input image has been manipulated using a bottom-up path. In this work, we detect the manipulation artifacts by explicitly adopting a set of learnable embeddings as object prototypes for object-level consistency modeling and the refined patch embeddings for patch-level consistency modeling.

Visual Transformer The remarkable success of Transformers [37] and their variants in natural language processing has motivated a plethora of work exploring transformers for a variety of computer vision tasks due to their capabilities in modeling long-range dependencies. More specifically, ViT [12] reshapes an image into a sequence of flattened patches and inputs them to the transformer encoders for image classification. T2T [45] incorporates a Token-to-Token module to progressively aggregate local information before using self-attention layers. There are also some studies combining the self-attention blocks with classical convolutional neural networks. For instance, DETR [5] uses the pretrained CNN for extracting low-level features, which are then fed into a transformer-based encoder-decoder architecture for object detection. In contrast, we introduce the frequency information to facilitate the capture of subtle forgery traces, which are further combined with RGB modality features through a multi-modal transformer for image tampering detection.

3. Method

Our goal is to detect manipulated objects within images by modeling visual consistencies among mid-level representations, which are automatically derived by attending to multimodal inputs. In this section, we introduce ObjectFormer, which consists of a High-frequency Feature Extraction Module (Section 3.1), an object encoder (Section 3.2) that uses learnable object queries to learn whether mid-level representations in images are coherent, and a patch decoder (Section 3.3) that produces refined global representations for manipulation detection and localization. Figure 2 gives an overview of the framework.

More formally, we denote an input image as $X \in \mathbb{R}^{H\times W\times 3}$, where H and W are the height and width of the

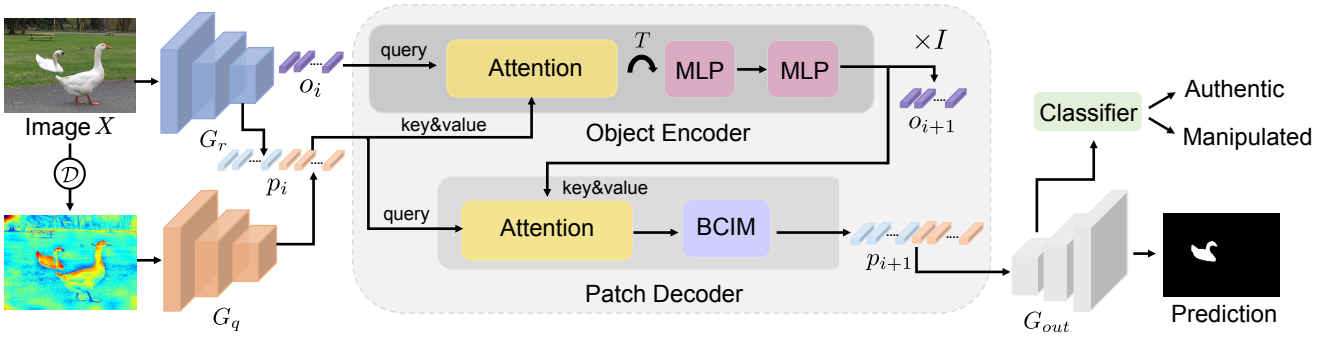

Figure 2. An overview of ObjectFormer. The input is a suspicious image ($H \times W \times 3$), and the output includes a tampering localization result and a predicted mask ($H \times W \times 1$), which localizes the manipulation regions.

image, respectively. We first extract the feature map $G_r \in \mathbb{R}^{H_s \times W_s \times C_s}$ and generate patch embeddings using a few convolutional layers, parameterized by g, for faster convergence as in [44].

3.1. High-frequency Feature Extraction

As manipulated images are generally post-processed to hide tampering artifacts, it is difficult to capture subtle forgery traces in the RGB space. Therefore, we extract features from the frequency domain to provide complementary clues for manipulation detection.

Taking the image X as input, ObjectFormer first transforms it from the RGB domain to the frequency domain using Discrete Cosine Transform (DCT):

$$X_q = \mathcal{D}(X), \tag{1}$$

where $X_q \in \mathbb{R}^{H \times W \times 1}$ is the frequency domain representation and $\mathcal{D}$ denotes DCT. Then we obtain the high-frequency component through a high pass filter, and transform it back to RGB domain to preserve the shift invariance and local consistency of natural images:

$$X_h = \mathcal{D}^{-1}(\mathcal{F}(X_q, \alpha)), \tag{2}$$

where $\mathcal{F}$ denotes the high pass filter and α is the manually-designed threshold which controls the low frequency component to be filtered out. After that, we input X_h to several convolutional layers to extract frequency features G_f, the size of which is the same with G_r.

We then generate spatial patches of the same sizes using G_r and G_f and further flatten them to a sequence of C-d vectors with the length of L. We concatenate the two sequences to obtain a multimodal patch vector $p \in \mathbb{R}^{2L \times C}$. Sinusoidal positional embeddings [5] are added to p to provide positional information.

3.2. Object Encoder

The object encoder aims to learn a group of mid-level representations automatically that attend to specific regions in G_r/G_f and identify whether these regions are consistent with each other. To this end, we use a set of learnable parameters $o \in \mathbb{R}^{N \times C}$ as object prototypes, which are learned to represent objects that may appear in images. N is a manually designed constant value indicating the maximum number of objects, which we empirically set to 16 in this paper.

Specifically, given the object representations o_i from the i-th layer, we first normalize it with Layer Normalization (LN) and use it as the query of the attention block. The patch embeddings p_i after the normalization serve as the key and value. Note that we set $p_0 = p, o_0 = o$, respectively. Then we calculate the object-patch affinity matrix $A_i \in \mathbb{R}^{N \times L}$ with matrix multiplication and a `softmax` function:

$$A_i = \texttt{softmax}\left(\frac{o_i W_{eq} \cdot (p_i W_{ek})^T}{\sqrt{C}}\right), \tag{3}$$

where W_{eq} and W_{ek} are learnable parameters of two linear projection layers. After that, we use another linear layer to project p_i into value embedding, and further compute its weighted average with A_i to obtain the attention matrix. Finally, the object representations are updated through a residual connection with the attention matrix to obtain $\hat{o}_i \in \mathbb{R}^{N \times C}$:

$$\hat{o}_i = o_i + A_i \cdot p_i W_{ev}, \tag{4}$$

where W_{ev} is the learnable parameter for the value embedding layer. With this, each object representation can be injected with global contextual information from all locations. Then we further enable the interaction among different objects using a single linear projection:

$$\widetilde{o}_i = \hat{o}_i + (\hat{o}_i^T W_c)^T, \tag{5}$$

where $W_c \in \mathbb{R}^{N \times N}$ is a learnable weight matrix. This essentially learns how different object prototypes interact with one another to discover object-level visual inconsistencies.

Since the number of objects within an image varies, we additionally use linear projection layers and an activation

function GELU [14] to enhance the object features. This process can be formulated as:

$$o_{i+1} = \widetilde{o}_i + \delta(\widetilde{o}_i W_{act_1}) W_{act_2}, \tag{6}$$

where W_{act_1} and W_{act_2} are learnable parameters, δ is the GELU function, o_{i+1} is the updated object representation.

3.3. Patch Decoder

The object encoder allows different objects within the images to interact with each other to model whether mid-level representations are visually coherent and attend to important patches. In addition to this, we use the updated object representations from the object encoder to further refine the patch embeddings. More specifically, we use p_i as query, o_{i+1} as key and value, and enhance the patch features following classic attention paradigm. With this, each patch embedding can further absorb useful information from the derived object prototypes.

More specifically, we first adopt Layer Normalization to normalize both p_i and o_{i+1}, and then feed them into an attention block for patch embeddings refinement. The complete process can be formulated as:

$$\begin{aligned} \hat{p}_i &= p_i + \texttt{softmax}\left(\frac{p_i W_{dq} \cdot (o_{i+1} W_{dk})^T}{\sqrt{C}}\right) \cdot o_{i+1} W_{dv}, \\ \overline{p}_i &= \hat{p}_i + \texttt{MLP}(\hat{p}_i), \end{aligned} \tag{7}$$

where W_{dq}, W_{dk}, and W_{dv} are the learnable parameters of three embedding layers, and MLP denotes a Multi-Layer Perceptron that has two linear mappings.

After aggregating the mid-level object features into each patch within the images, we further apply a boundary-sensitive contextual incoherence modeling (*BCIM*) module to detect pixel-level inconsistency for fine-grained feature modeling. In particular, we first reshape $\overline{p}_i \in \mathbb{R}^{2N \times C}$ to a 2D feature map $\widetilde{P}_i$ with the size $\mathbb{R}^{H_s \times W_s \times 2C_s}$. We then calculate the similarity between each pixel and surrounding pixels within a local window:

$$S_{i_j} = \frac{1}{k \times k} \sum_{j \in \kappa} \texttt{Sim}(\widetilde{P}_{i_j}, \widetilde{P}_{i_k}), \tag{8}$$

where κ denotes a small $k \times k$ window in the feature map $\widetilde{P}_i$, $\widetilde{P}_{i_j}$ is the central feature vector of the window, and $\widetilde{P}_{i_k}$ is its neighboring feature vector within κ. The similarity measurement function Sim that we use is cosine similarity. Then we compute the element-wise summation between $S_i \in \mathbb{R}^{H_s \times W_s \times 1}$ and $\widetilde{P}_i$ to obtain a boundary-sensitive feature map with the size of $\mathbb{R}^{H_s \times W_s \times 2C_s}$ to obtain a boundary-sensitive feature map, and finally serialize it to patch embeddings $p_{i+1} \in \mathbb{R}^{2N \times C}$.

Note that we use stacked object encoders and image decoders in a sequential order for I times (which we set to 8 in this paper) to alternately update the object representations and patch features. Finally, we obtain $p_{out} \in \mathbb{R}^{2N \times C}$ which contains visual consistency information at both the object-level and the patch-level. After that, we reshape it to a 2D feature map G_{out}, which is then used for manipulation detection and localization.

3.4. Loss Functions

For manipulation detection, we apply global average pooling on G_{out}, and calculate the final binary prediction $\hat{y}$ using a fully connected layer. While for manipulation localization, we progressively upsample G_{out} by alternating convolutional layers and linear interpolation operations to obtain a predicted mask $\hat{M}$. Given the ground-truth label y and mask M, we train ObjectFormer with the following objective function:

$$\mathcal{L} = \mathcal{L}_{cls}(y, \hat{y}) + \lambda \mathcal{L}_{seg}(M, \hat{M}), \tag{9}$$

where both $\mathcal{L}_{cls}$ and $\mathcal{L}_{seg}$ are binary cross-entropy loss, and λ_{seg} is a balancing hyperparameter. By default, we set $\lambda_{seg} = 1$.

4. Experiments

We evaluate our models on two closely related tasks: manipulation localization and detection. In the former task, our goal is to localize the manipulated regions within the images. In the latter task, the goal is to classify images as being manipulated or authentic. Below, we introduce the experimental setup in Sec. 4.1, and present results in Sec. 4.2 - Sec. 4.4, and finally we perform an ablation study to justify the effectiveness of different components in Sec. 4.5, and show the visualization results in Sec. 4.6.

4.1. Experimental Settings

Synthesized Pre-training Data We synthesize a large-scale image tampering dataset and pre-train our model on it. The synthesized dataset includes three subsets: 1) FakeCOCO, which is built on MS COCO [21]. Inspired by [46], we use the annotations provided by MS COCO to randomly copy and paste an object within the same image, or slice an object from one image to another. We also apply the Poisson Blending algorithm between source and target images to erase the slicing boundaries. 2) FakeParis, which is built on Paris StreetView [30] dataset. We erase a region from an authentic image, and adopt a state-of-the-art inpainting method Edgeconnnect [26] to restore visual contents within it. 3) Pristine images, *i.e.*, the original images from the above datasets. We randomly add Gaussian noise or apply the JPEG compression algorithm to the generated

data to resemble the visual quality of images in realistic scenarios.

Testing Datasets We follow PSCCNet [22] to evaluate our model on CASIA [11] dataset, Columbia [35] dataset, Carvalho [42], Nist Nimble 2016 (NIST16) dataset [27], and IMD20 [28] dataset.

- **CASIA** [11] provides spliced and copy-moved images of various objects. The tampered regions are carefully selected and some post-processing techniques like filtering and blurring are also applied. Ground-truth masks are obtained by binarizing the difference between tampered and original images.
- **Columbia** [35] dataset focuses on splicing based on uncompressed images. Ground-truth masks are provided.
- **Coverage** [42] dataset contains 100 images generated by copy-move techniques, the ground-truth masks are also available.
- **NIST16** [27] is a challenging dataset which contains all three tampering techniques. The manipulations in this dataset are post-processed to conceal visible traces. They provide ground-truth tampering mask for evaluation.
- **IMD20** [28], which collects 35,000 real images captured by different camera models and generates the same number of forged images using a large variety of Inpainting methods.

To fine-tune ObjectFormer, we use the same training/testing splits as [16, 22] for fair comparisons.

Evaluation Metrics We evaluate the performance of the proposed method on both image manipulation detection task and localization task. For detection results, we use image-level Area Under Curve (AUC) and F1 score as our evaluation metric, while for localization, the pixel-level AUC and F1 score on manipulation masks are adopted. Since binary masks and detection scores are required to compute F1 scores, we adopt the Equal Error Rate (EER) threshold to binarize them.

Implementation Details All images are resized to 256×256. For our backbone network, we use EfficientNet-b4 [36] pretrained on ImageNet [9]. We use Adam for optimization with a learning rate of 0.0001. We train the complete model for 90 epochs with a batch size of 24, and the learning rate is decayed by 10 times every 30 epochs.

Baseline Models We compare our method with various baseline models as described below:

- **J-LSTM** [2], which employs a hybrid CNN-LSTM architecture to capture the discriminative features between manipulated and non-manipulated regions within a tampered image.
- **H-LSTM** [3], which segments the resampling features extracted by a CNN encoder into patches and adopts an LSTM network to model the transition between different patches for tampering localization.
- **RGB-N** [46], which adopts an RGB stream and a noise stream in parallel to separately discover tampering features and noise inconsistency within an image.
- **ManTraNet** [43], which uses a feature extractor to capture the manipulation traces and a local anomaly detection network to localize the manipulated regions.
- **SPAN** [16], which leverages a pyramid architecture and models the dependency of image patches through self-attention blocks.
- **PSCCNet** [22], which employs features at different scales progressively for image tampering localization in a coarse-to-fine manner.

4.2. Image Manipulation Localization

Compared with binary tampering detection task, manipulation localization is more challenging because it requires the models to capture more refined forgery features. Following SPAN [16] and PSCCNet [22], we compare our model with other state-of-the-art tampering localization methods under two settings: 1). training on the synthetic dataset and evaluating on the full test datasets. 2) fine-tuning the pre-trained model on the training split of test datasets and evaluating on their test split.

Pre-trained Model For pre-trained model evaluation, we compare ObjectFormer with MantraNet [43], SPAN [16], and PSCCNet [22]. We report the AUC scores (%) in Table 1, from which we can observe ObjectFormer achieves the best localization performance on most datasets. Especially, ObjectFormer achieves 82.1% on the real-world dataset IMD20, and outperforms PSCCNet by 1.9%. This suggests our method owns the superior ability to capture tampering features, and can generalize well to high-quality manipulated images datasets. On Columbia dataset, we surpass SPAN and MaTraNet by 2.0% and 15.9%, but are 2.7% behind PSCCNet. We argue that the reason might be their synthesized training data closely resemble the distribution of the Columbia dataset. This can be further verified by the results in Table 2, which demonstrates ObjectFormer outperforms PSCCNet in both AUC and F1 scores if the model is fine-tuned on Columbia dataset. Moreover, it is worth pointing out ObjectFormer achieves decent results using less pre-training data compared with other methods.

Fine-tuned Model To compensate for the difference in visual quality between the synthesized datasets and standard datasets, we further fine-tune the pre-trained models on spe-

Method	#Data	Colombia	Coverage	CASIA	NIST16	IMD20
ManTraNet	64K	82.4	81.9	81.7	79.5	74.8
SPAN	96K	93.6	92.2	79.7	84.0	75.0
PSCCNet	100K	**98.2**	84.7	82.9	85.5	80.6
Ours	62K	95.5	**92.8**	**84.3**	**87.2**	**82.1**

Table 1. Comparisons of manipulation localization AUC (%) scores of different pre-trained models.

Method	Coverage		CASIA		NIST16	
	AUC	F1	AUC	F1	AUC	F1
J-LSTM	61.4	-	-	-	76.4	-
H-LSTM	71.2	-	-	-	79.4	-
RGB-N	81.7	43.7	79.5	40.8	93.7	72.2
SPAN	93.7	55.8	83.8	38.2	96.1	58.2
PSCCNet	94.1	72.3	87.5	55.4	99.6	81.9
Ours	**95.7**	**75.8**	**88.2**	**57.9**	**99.6**	**82.4**

Table 2. Comparison of manipulation localization results using fine-tuned models.

cific datasets, and compare with other methods in Table 2. We can observe significant performance gains, which illustrates that ObjectFormer could capture subtle tampering artifacts through the object-level and patch-level consistency modeling and the multimodal design.

4.3. Image Manipulation Detection

Since most previous studies do not consider tampering detection task, we evaluate our model on CASIA-D introduced by [22]. Table 3 shows the AUC and F1 scores (%) for detecting manipulated images. The results demonstrate that our model achieves state-of-the-art performance, *i.e.*, 99.70% in terms of AUC and 97.34% in F1, which demonstrates the effectiveness of our method to capture manipulation artifacts.

Method	AUC	F1
MantraNet	59.94	56.69
SPAN	67.33	63.48
PSCCNet	99.65	97.12
Ours	**99.70**	**97.34**

Table 3. AUC and F1 scores (%) of tampering detection results on CASIA-D dataset [22]. The best results are marked in bold.

4.4. Robustness Evaluation

Following PSCCNet [22], we apply different image distortion methods on raw images from NIST16 dataset and evaluate the robustness of our ObjectFormer. The distortion types include: 1) image scaling with different scales, 2) Gaussian blurring with a kernel size k, 3) Gaussian noise with a standard deviation σ, and 4) JPEG compression with a quality factor q. We compare the manipulation localization performance (AUC scores) of our pre-trained models with SPAN [16] and PSCCNet [22] on these corrupted data, and report the results in Table 4. ObjectFormer demonstrates better robustness against various distortion techniques, especially on compressed images (1.1% higher than PSCCNet when the quality factor is 100, and 1.0% higher when the quality factor is 50).

Distortion	SPAN	PSCCNet	Ours
no distortion	83.95	85.47	**87.18**
Resize (0.78×)	83.24	85.29	**87.17** ↓0.01
Resize (0.25×)	80.32	85.01	**86.33** ↓0.85
GaussianBlur (k=3)	83.10	85.38	**85.97** ↓1.21
GaussianBlur (k=15)	79.15	79.93	**80.26** ↓6.92
GaussianNoise (σ=3)	75.17	78.42	**79.58** ↓7.60
GaussianNoise (σ=15)	67.28	76.65	**78.15** ↓9.03
JPEGCompress (q=100)	83.59	85.40	**86.37** ↓0.81
JPEGCompress (q=50)	80.68	85.37	**86.24** ↓0.94

Table 4. Localization performance on NIST16 dataset under various distortions. AUC scores are reported (in %).

4.5. Ablation Analysis

The High-frequency Feature Extraction (*HFE*) module of our method is designed to extract the abnormal forgery features in the frequency domain, while the Boundary-sensitivity Contextual Incoherence Modeling (*BCIM*) module is utilized to improve the sharpness of the predicted tampering masks. To evaluate the effectiveness of *HFE* and *BCIM*, we remove them separately from ObjectFormer and evaluate the tampering localization performance on CASIA and NIST16 dataset.

The quantitative results are listed in Table 5. We can observe that without *HFE*, the AUC scores decrease by 14.6% on CASIA and 11.0% on NIST16, while without *BCIM*, the AUC scores decrease by 6.2% on CASIA and 2.4% on NIST16. The performance degradation validates that the use of *HFE* and *BCIM* effectively improves the performance of our model. Moreover, to illustrate the effectiveness of representations learned by ObjectFormer, we discard the object representations and replace the stacked object encoders and image decoders with vanilla self-attention blocks. We can observe a significant performance degradation in the third row of Table 5, *i.e.*, 5% in terms of AUC and 12.5% in terms of F1 on NIST16 dataset.

The object prototypes are deployed to represent the vi-

Variants	CASIA		NIST16	
	AUC	F1	AUC	F1
Baseline (EfficientNet-b4)	70.3	37.2	77.4	51.9
w/o *HFE*	75.3	51.5	88.6	63.9
w/o *BCIM*	82.7	40.1	97.2	74.5
vanilla self-attention	84.8	47.6	94.5	72.1
Ours	**88.2**	**57.9**	**99.6**	**82.4**

Table 5. Ablation results on CASIA and NIST16 dataset using different variants of ObjectFormer. Both AUC and F1 scores (%) are reported.

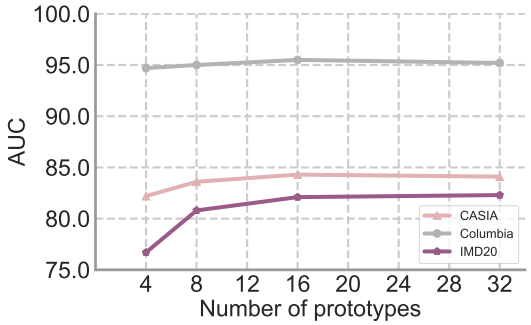

Figure 3. AUC scores (%) of ObjectFormer with different numbers of object prototypes.

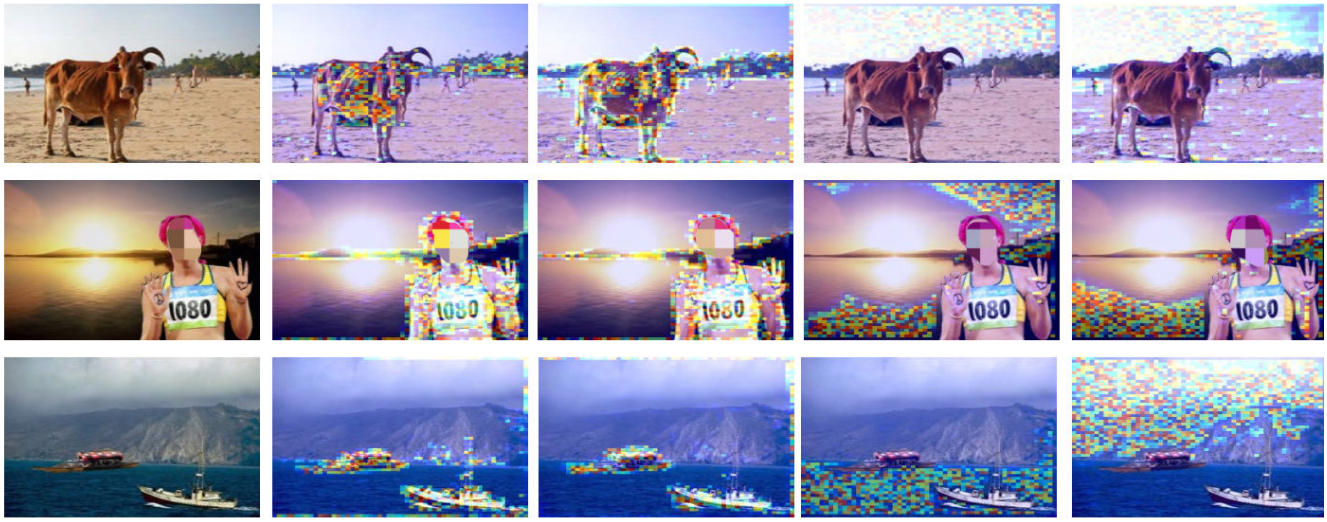

Figure 4. Visualization of the affinity matrix A_i in the first object encoder. From left to right, we display the forged images, two prototypes corresponding to two foreground objects, and two prototypes related to background objects.

sual elements that may appear in an image, which facilitates ObjectFormer to learn the mid-level semantic features for object-level consistencies modeling. We further conduct experiments to investigate the effect of the number of prototypes (N) on model performance. As shown in Figure 3, there is an overall incremental trend in the tampering location performance as the number of prototypes increases, and the best are achieved on Columbia and CASIA dataset when N is set to 16.

4.6. Visualization Results

Visualization of object encoder. We further investigate the behavior of ObjectFormer qualitatively. Specifically, we average all heads of the affinity matrix A_i (Eqn. 3) in the first object encoder, and then normalize it to [0, 255]. For each image, we visualize the pristine image (column 1), and regions that are attended to by different object prototypes, *e.g.*, column 2 and 3 are two prototypes correspond to two foreground objects while column 4 and 5 relate to background objects. The results in Figure 4 suggest that through iterative updates, the object representations correspond to meaningful regions in the images, thus contributing to object consistency modeling.

Qualitative results. We provide predicted manipulation masks of different methods in Figure 5. Since the source code of PSCCNet [22] is not available, their predictions are not available. The results demonstrate that our method could not only locate the tampering regions more accurately, but also develop more sharp boundaries, which benefits from the inconsistency modeling ability and boundaries sensitivity of ObjectFormer.

Visualization of high-frequency features. To verify the usefulness of frequency features for tampering detection, we visualize the high-frequency components and *HFE* features using GradCAM [34] (Sec. 3.1) in Figure 6. The results demonstrate that although the forged images are visually natural, the manipulated regions are distinguishable

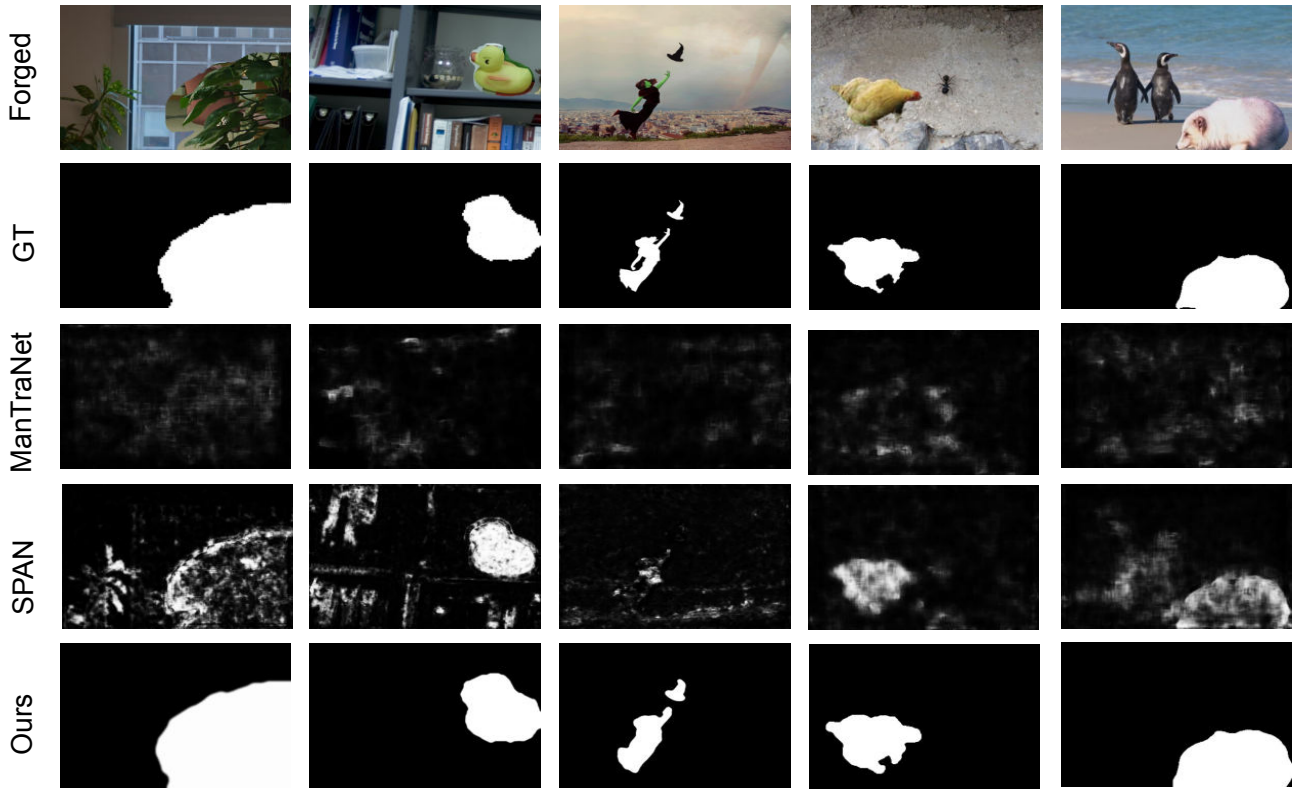

Figure 5. Visualization of the predicted manipulation mask by different methods. From top to bottom, we show forged images, GT masks, predictions of ManTraNet, SPAN, and ours.

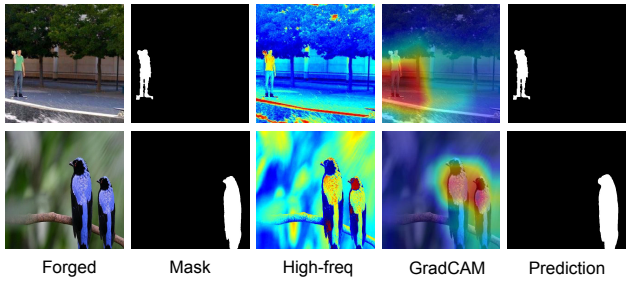

Figure 6. Visualization of the frequency features. From left to right, we display the forged images, masks, the high-frequency components, GradCAM of the feature maps after conv layer, and ObjectFormer predictions.

from the untouched areas in frequency domain.

4.7. Limitation

ObjectFormer faces one potential limitation: when using the pre-trained model to evaluate the performance of tampering localization on Columbia, ObjectFormer is 2.7% lower than PSCCNet [22] in terms of AUC score. The possible reason might be that the pre-training data they use closely resemble the data distribution in the Columbia dataset. Therefore, we believe this problem can be resolved by using more pre-training data.

5. Conclusion

We introduced ObjectFormer, an end-to-end multimodal framework for image tampering detection and localization. To detect subtle manipulation artifacts that are no longer visible in RGB domain, ObjectFormer extracts forgery features in the frequency domain as complementary information, which are further combined with the RGB features to generate multimodal patch embeddings. Additionally, ObjectFormer leverages learnable object prototypes as mid-level representations, and alternately updates the object prototypes and patch embeddings with stacked object encoders and patch decoders to model the object-level and patch-level visual consistencies within the images. Extensive experiments on different datasets demonstrate the effectiveness of the proposed method.

Acknowledgement This work was supported by National Natural Science Foundation of Project (62072116). Y.-G. Jiang was sponsored in part by "Shuguang Program" supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission (20SG01).

References

[1] Song Bai, Philip Torr, et al. Visual parser: Representing part-whole hierarchies with transformers. *arXiv preprint arXiv:2107.05790*, 2021. 2

[2] Jawadul H Bappy, Amit K Roy-Chowdhury, Jason Bunk, Lakshmanan Nataraj, and BS Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *ICCV*, 2017. 5

[3] Jawadul H Bappy, Cody Simons, Lakshmanan Nataraj, BS Manjunath, and Amit K Roy-Chowdhury. Hybrid lstm and encoder–decoder architecture for detection of image forgeries. *TIP*, 2019. 2, 5

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021. 2

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2, 3

[6] Shen Chen, Taiping Yao, Yang Chen, Shouhong Ding, Jilin Li, and Rongrong Ji. Local relation learning for face forgery detection. In *AAAI*, 2021. 1

[7] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Efficient dense-field copy–move forgery detection. *TIFS*, 2015. 2

[8] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *WIFS*, 2015. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[10] Helisa Dhamo, Azade Farshad, Iro Laina, Nassir Navab, Gregory D Hager, Federico Tombari, and Christian Rupprecht. Semantic image manipulation using scene graphs. In *CVPR*, 2020. 1

[11] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database. In *ChinaSIP*, 2013. 2, 5

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 1

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[15] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021. 2

[16] Xuefeng Hu, Zhihan Zhang, Zhenye Jiang, Syomantak Chaudhuri, Zhenheng Yang, and Ram Nevatia. Span: Spatial pyramid attention network for image manipulation localization. In *ECCV*, 2020. 1, 2, 5, 6

[17] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018. 2

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[19] Vladimir V Kniaz, Vladimir Knyaz, and Fabio Remondino. The point where reality meets fantasy: Mixed adversarial generators for image splice detection. 2019. 2

[20] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *CVPR*, 2020. 1

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 4

[22] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. Pscc-net: Progressive spatio-channel correlation network for image manipulation detection and localization. *arXiv preprint arXiv:2103.10596*, 2021. 2, 5, 6, 7, 8

[23] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 2

[24] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *CVPR*, 2022. 2

[25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[26] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. Edgeconnect: Structure guided image inpainting using edge prediction. In *ICCVW*, 2019. 4

[27] Nist. Nimble 2016 datasets. https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation, 2016. 2, 5

[28] Adam Novozamsky, Babak Mahdian, and Stanislav Saic. Imd2020: A large-scale annotated dataset tailored for detecting manipulated images. In *WACVW*, 2020. 2, 5

[29] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *NIPS*, 2020. 1

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 4

[31] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 2020. 1

[32] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *WIFS*, 2016. 2

[33] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-resolution images with vq-vae. In *ICLR Workshop*, 2019. 1

[34] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 7

[35] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000. 2, 5

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

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2

[38] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020. 1

[39] Junke Wang, Zuxuan Wu, Jingjing Chen, and Yu-Gang Jiang. M2tr: Multi-modal multi-scale transformers for deepfake detection. *arXiv preprint arXiv:2104.09770*, 2021. 1

[40] Junke Wang, Xitong Yang, Hengduo Li, Zuxuan Wu, and Yu-Gang Jiang. Efficient video transformers with spatial-temporal token selection. *arXiv preprint arXiv:2111.11591*, 2021. 2

[41] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. Bevt: Bert pretraining of video transformers. In *CVPR*, 2022. 2

[42] 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. 2, 5

[43] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *CVPR*, 2019. 1, 2, 5

[44] Tete Xiao, Piotr Dollar, Mannat Singh, Eric Mintun, Trevor Darrell, and Ross Girshick. Early convolutions help transformers see better. In *NIPS*, 2021. 3

[45] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021. 2

[46] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018. 1, 2, 4, 5

[47] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1

[48] Xinshan Zhu, Yongjun Qian, Xianfeng Zhao, Biao Sun, and Ya Sun. A deep learning approach to patch-based image inpainting forensics. *Signal Processing: Image Communication*, 2018. 2

[49] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2