Minimalist Preprocessing Approach for Image Synthesis Detection

Hoai-Danh Vo[1,2], Trung-Nghia Le*[1,2]

[1] University of Science, VNU-HCM, Ho Chi Minh City, Vietnam
[2] Vietnam National University, Ho Chi Minh City, Vietnam
22c15025@student.hcmus.edu.vn, ltnghia@fit.hcmus.edu.vn

Abstract. Generative models have significantly advanced image generation, resulting in synthesized images that are increasingly indistinguishable from authentic ones. However, the creation of fake images with malicious intent is a growing concern. Low-configured smart devices have become highly popular, making it easier for deceptive images to reach users. Consequently, the demand for effective detection methods is increasingly urgent. In this paper, we introduce a simple yet efficient method that captures pixel fluctuations between neighboring pixels by calculating the gradient, which highlights variations in grayscale intensity. This approach functions as a high-pass filter, emphasizing key features for accurate image distinction while minimizing color influence. Our experiments on multiple datasets demonstrate that our method achieves accuracy levels comparable to state-of-the-art techniques while requiring minimal computational resources. Therefore, it is suitable for deployment on low-end devices such as smartphones. The code is available at https://github.com/vohoaidanh/adof.

Keywords: Image synthesis detection · Lightweight model · Low-level computation

1 Introduction

In recent years, significant advancements in image generation have been achieved, particularly with Generative Adversarial Networks (GANs) [11] and Diffusion models [12, 14]. These approaches produce high-quality images that closely resemble real-world visuals [31] and have garnered attention in academic and societal circles. Generative models have found applications in various fields, including virtual try-ons and personalized fashion recommendations in the fashion industry [25], as well as in image editing [4, 39] and interior design [6].

Despite the valuable applications of image generation technology, significant drawbacks exist. According to a survey conducted by Bauer and Bindschaedlerr [2], generative models can create fake information, particularly deepfakes, which depict fabricated scenarios involving famous individuals. In response to these dangers, several US states [3, 20] have outlawed the malicious use of

* Corresponding author.

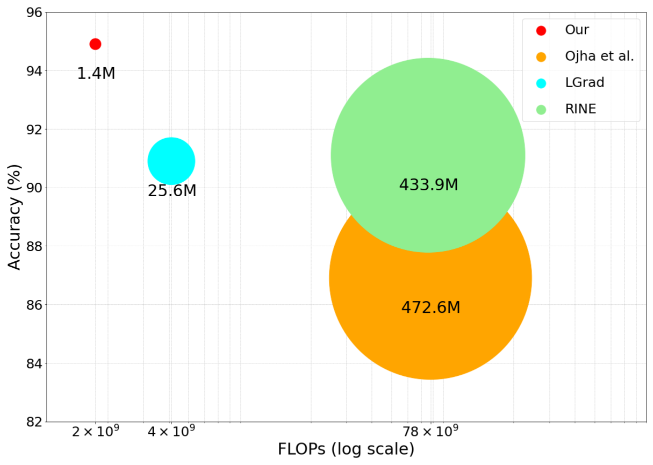

Fig. 1. Comparison of different synthetic image detection methods on the Ojha dataset [28]. Our proposed method is simple yet efficient, significantly reduces FLOPs, total parameters, while achieving comparable accuracy state-of-the-art methods.

deepfake technology, especially for harmful content like revenge and celebrity pornography. To address the threats posed by synthetic images on digital communication platforms and social media, it is essential to develop effective countermeasures for verifying image authenticity directly on mobile devices. Given the ubiquity and portability of these devices, real-time detection of generated images is crucial for preventing misinformation and preserving the integrity of visual content. However, the constrained computational capacity of mobile devices presents a significant challenge. This paper introduces a simple yet efficient solution for synthesized image detection, specifically the Adjacency Difference Orientation Filter (ADOF) for data preprocessing; this filter allows us to compute the gradient in both the x and y directions. The direction of the gradient reflects the behavior of grayscale variation among neighboring pixels, assisting in distinguishing between real and generated images. Focusing on extracting useful low-level features, our approach ensures generalization while utilizing a lightweight CNN architecture for detecting generated images, without demanding extensive computational resources. This strategy effectively reduces irrelevant information, enabling the model to concentrate on fine-grained variations, ultimately leading to improved performance and generalization. In contrast to existing methods [21, 28, 35] that require large deep learning architectures, such as CLIP [30], ViT [7], Resnet50 [13], and significant computational resources, our approach demands fewer resources while still ensuring generalization and achieving comparable accuracy. Fig. 1 presents a comparative overview of results, highlighting the advantages of this strategy.

Experiments on well-known datasets [28,36,37] demonstrate the effectiveness of our method, achieving impressive accuracy of 94.9% on the Ojha dataset [28] and 98.3 % on the DiffusionForensis [37]. Additionally, there is a reduction in computational load by 97.8% compared to RINE [21] and 57.8% compared to LGrad [35]. These results underscore the advantages of our approach, as illustrated in Fig. 1. The code for reproducing our results is publicly available at https://github.com/vohoaidanh/adof. Our contributions are as follow:

- We introduce a simple yet efficient approach for detecting synthetic images, and our approach is more generalized than existing methods.
- We present a filter-based method that computes pixel intensity gradients to capture pixel fluctuations and reduce color influence, leading to improved model performance with faster inference speed and lower complexity.
- Our proposed method reduces the number of parameters and FLOPs while maintaining accuracy compared to state-of-the-art methods.

2 Related Work

Various methods have been developed to address the challenge of distinguishing synthetic images from real ones, utilizing both traditional machine learning techniques and modern deep learning approaches. Durall *et al.* [8] applied a Fourier Transform [1] to grayscale images and used azimuthal averaging to convert the 2D frequency data into a 1D feature vector, retaining essential information for classification. They then employed either Support Vector Machines or K-means clustering to detect GAN-generated images. Alternatively, methods like RINE [21] and Ojha et al. [28], along with similar approaches, leverage pre-trained deep learning networks such as CLIP to enhance performance. This integration contributing to consistently high success rates in detecting synthesized images through the integration of these networks into their frameworks. Notably, the FatFormer [23] method focuses on the contrastive objectives between adapted image features and text prompt embeddings, providing valuable information that enables the deep learning models to learn more robust and discriminative representations, ultimately improving their ability to accurately classify real and generated images.

Frequency domain-based methods involve transforming images from the spatial domain to the frequency domain using transformations such as Fast Fourier Transform or Discrete Cosine Transform (DCT). By focusing on frequency characteristics, these methods effectively capture artifacts that might not be evident in the spatial domain. This allows classifiers to distinguish between real and fake images by analyzing the unique patterns that emerge in the spectral domain. Frank *et al.* [10] utilized the DCT to analyze images in the frequency domain, revealing unique spectral differences between real images and those produced by GAN models. Qian *et al.* introduced F^3-*Net* [29] to decompose the spectrum into various bands, enabling the analysis of these components to identify unusual distributions. This method effectively detects subtle artifacts, enhancing the ability to recognize synthetic image manipulations.

Tan *et al.* proposed FreqNet [33], which emphasizes high-frequency details and directs the detector to concentrate on these features across spatial and channel dimensions, rather than utilizing the full spectrum of frequency bands as is common in many other approaches. BiHPF [16], a method by Jeong *et al.*, amplifies frequency-level artifacts commonly found in images generated by generative models to tackle the challenge of identifying images from previously unobserved models. Jeong *et al.* [15] generated perturbation maps added to training images to prevent overfitting to frequency-specific features, reducing high-frequency noise and enhancing classifier generalization.

Spatial-based methods analyze images directly on pixel values, as seen in models like CNNDetection [36] and Gram-Net [24]. A key issue, however, is that raw images often contain excessive, irrelevant information, such as semantic content, which Nang *et al.* [40] identified as detrimental to image classification effectiveness. This extraneous information, unnecessary for distinguishing real from fake images, can disrupt the model's learning process and reduce its effectiveness. Wang *et al.* [36] developed a comprehensive detector to distinguish real images from CNN-generated ones [22]. Using a dataset of images from 11 CNN-based generators, they showed that, with effective pre-/post-processing and data augmentation, a classifier trained on ProGAN [18] generalizes well to other models, including StyleGAN2 [19]. By dividing images into small patches categorized as either rich texture or flat, PatchCraft [40] exploits the inter-pixel correlation contrast between these regions. This approach breaks the semantic coherence present in traditional methods, addressing a key limitation and enhancing the model's ability to generalize more effectively. Tan *et al.* introduced the concept of Neighboring Pixel Relationships (NPR) [34] to capture and characterize generalized structural artifacts that arise from up-sampling operations, which are commonly used in image generation models to enhance image quality. This method shows a significant improvement over other techniques within the same approach.

Frequency-based approaches achieve faster convergence with smaller models but may lose essential spatial information needed to distinguish real from generated images. In contrast, spatial-based methods often require larger models and may struggle with new domain data. Our method leverages the strengths of both approaches by focusing on pixel perturbations between neighboring pixels, effectively discarding much of the pixel color information. Additionally, our filter functions as a high-pass filter, removing low-frequency components to emphasize the relevant features.

3 Proposed Method

3.1 Overview

Generative models, such as GAN [11] and Diffusion [14], currently use CNN layers for image synthesis, meaning that neighboring pixel regions are correlated to a certain extent. We hypothesize that synthetic images exhibits a stronger correlation between adjacent pixels compared to real images. Furthermore, due to the design of neural networks, noise in synthetic images tends to be averaged out, whereas in real images, noise typically remains more prominent. To

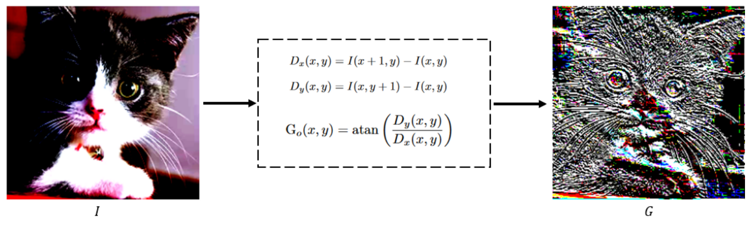
$D_x(x, y) = I(x+1, y) - I(x, y)$
$D_y(x, y) = I(x, y+1) - I(x, y)$
$G_o(x, y) = \text{atan}\left(\frac{D_y(x,y)}{D_x(x,y)}\right)$
I
G

Fig. 2. The left image represents the original image, the middle shows the gradient calculation applied, and the right image illustrates the resulting gradient map.

investigate and evaluate the impact of noise on adjacent pixels, we designed a simple yet effective filter to capture these variations. This filter aids the CNN in learning the differences in noise distribution between real and synthetic images. The overall architecture of ADOF is illustrated in Fig. 2. This approach captures noise information by calculating the differences between adjacent pixels and incorporating these differences into the gradient to account for variations in both the x and y directions.

3.2 Adjacency Difference Orientation Filter (ADOF)

Finite Difference. The finite difference technique [27, 38] is a mathematical approach for estimating intensity variations across neighboring points in a grid or matrix. In image processing, it calculates gradients by measuring differences in pixel values, thereby detecting changes in intensity in both horizontal and vertical directions. This technique aids in edge detection and texture analysis by highlighting contrasts between adjacent pixels.

The general formula for calculating the gradient in a given direction u is $\text{Gradient}_u = I(x + \Delta x, y + \Delta y) - I(x, y)$, where Δx and Δy define the direction of the difference.

Filter Construction. We have applied the *finite difference* to compute the gradients of intensity in our images. This helps in identifying intensity variations at each pixel and supports the detection of important geometric features in our approach. The formula that computes the difference between adjacent pixels along the x and y-direction is given by:

$$D_x(x, y) = I(x + 1, y) - I(x, y), \tag{1}$$

$$D_y(x, y) = I(x, y + 1) - I(x, y), \tag{2}$$

where I represents the image in which the difference is being calculated, $D_x(x, y)$ represents the difference between the pixel value at $(x+1, y)$ and the pixel value at (x, y). This filter captures variations in pixel intensity along the horizontal direction. Similarly, $D_y(x, y)$ captures variations in pixel intensity along the

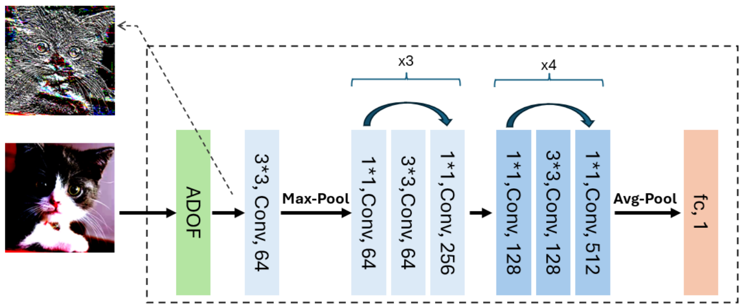

Fig. 3. Architecture of our lightweight model.

vertical direction. To determine the gradient magnitude and orientation, these values are computed from D_x and D_y.

$$\mathrm{G}_m(x,y) = \sqrt{D_x(x,y)^2 + D_y(x,y)^2}, \tag{3}$$

$$\mathrm{G}_o(x,y) = \arctan\left(\frac{D_y(x,y)}{D_x(x,y)}\right), \tag{4}$$

where $D_x(x,y)$ and $D_y(x,y)$ are as previously defined. The gradient orientation G_o, which represents the overall angle of gray-level changes at a pixel and indicates the direction of these combined intensity variations, is referred to as the ***Adjacency Difference Orientation Filter (ADOF)*** in this paper. Meanwhile, the gradient magnitude G_m quantifies the strength of intensity changes at that pixel. The result of this computational process is illustrated in Figure 2.

3.3 Lightweight Model Architecture

To evaluate the effectiveness of our filter ADOF on images, we use basic CNN architectures. Specifically, this work employs a modified ResNet50 [13] model with `layer3` and `layer4` removed. To capture information from 8-connected neighboring pixels more effectively, the kernel size of the `conv1` layer was adjusted from 7 to 3. The architecture is depicted in Fig. 3.

4 Experiments

4.1 Implementation Details

In practice, we are more concerned with the flat regions of an image rather than the edge areas where there is a significant variation in gray levels between the x and y directions. This is because, in regions with large changes in gray levels in one direction compared to the other, the gradient angles are close to $\pm\frac{\pi}{2}$. Although these angles both indicate edge regions in the image, the gradient angles at edges typically take values of $\pm\frac{\pi}{2}$, which are numerically distant from each other despite conveying similar edge information. To exclude these areas, we set the gradient values approaching $\pm\frac{\pi}{2}$ to 0 , the experimental process has demonstrated that this approach leads to higher accuracy for the model.

All experiments are conducted on a computing system using a NVIDIA RTX A4000 GPU with 16 GB of memory and an AMD Ryzen 5 5600X 6-Core CPU. We trained our model using parameters that are closely aligned with those used in common methods [34–36] to ensure a fair comparison and demonstrate the effectiveness of our method independent of specific hyperparameters. Furthermore, we utilized the source code provided by NPR [34] to streamline the training process and maintain consistency. The model was trained using the Adam optimizer with a learning rate of 2×10^{-4} and a batch size of 32. To accelerate the training process, we adjusted the learning rate every 5 epochs instead of every 10 epochs and utilized 4 out of the 20 classes (*car, cat, chair, horse*) for training, similar to the protocol used in existing works [15, 16, 34, 36].

4.2 Dataset

Training set. To facilitate comparison between methods, we used the same ForenSynths dataset with existing methods [17, 28, 34–36]. This dataset consists of 20 object classes selected from the LSUN dataset. Each class contains 18,000 real-world images, with corresponding generative images generated using the ProGAN [18] model. To verify the generalization of methods, all compared method was trained on a subset of the ForenSynths [36] dataset consisting of 4 classes: car, cat, chair, horse.

Evaluation set. To investigate the generalization of methods, our evaluation was conducted using the Self-Synthesis 9 GANs [34], which contains 36,000 images sourced from LSUN, ImageNet, CelebA, CelebA-HQ, COCO, and FaceForensics++, generated using models like AttGAN, BEGAN, and CramerGAN. The second dataset, DiffusionForensics [37], comprises 40,000 images from LSUN and ImageNet, utilizing models such as ADM, DDPM, and IDDPM. Lastly, the Ojha Test Set [28] includes 16,000 images from LAION and ImageNet, generated with ADM, Glide, DALL-E-Mini, and LDM.

4.3 Comparison with State-of-the-Art Methods

We conduct a performance comparison of our method with 10 State-of-the-Art methods, including CNNDetection [36], Frank [10], Durall [9], Patchfor [5], F3Net [29] , SelfBland [32], GANDetection [26], LGrad [35] , Ojha [28], NPR [34]. The experimental results in Tables 1, 2, and 3 demonstrate that our method exceeds existing approaches. On the 9-GAN dataset, ADOF delivers the highest accuracy at 94.2%, surpassing Ojha [28] with a mere 77.6% 1, and NPR [34] at 93.2% (see Table 1). Notably, our approach achieves a remarkable 98.3% accuracy on the DiffusionForensics [37] dataset, outperforming the NPR method [34],

Table 1. Evaluation results on the Self-Synthesis 9 GANs [34].

Method	**AttGAN**		**BEGAN**		**CramerGAN**		**InfoMaxGAN**		**MMDGAN**		**RelGAN**		**S3GAN**		**SNGAN**		**STGAN**		**Mean**	
	Acc.	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	A.P.	**Acc.**	**A.P.**
CNNDetection [36]	51.1	83.7	50.2	44.9	81.5	97.5	71.1	94.7	72.9	94.4	53.3	82.1	55.2	66.1	62.7	90.4	63.0	92.7	62.3	82.9
Frank [10]	65.0	74.4	39.4	39.9	31.0	36.0	41.1	41.0	38.4	40.5	69.2	96.2	69.7	81.9	48.4	47.9	25.4	34.0	47.5	54.7
Durall [9]	39.9	38.2	48.2	30.9	60.9	67.2	50.1	51.7	59.5	65.5	80.0	88.2	**87.3**	97.0	54.8	58.9	62.1	72.5	60.3	63.3
Patchfor [5]	68.0	92.9	97.1	100.0	97.8	**99.9**	93.6	98.2	97.9	**100.0**	99.6	100.0	66.8	68.1	**97.6**	**99.8**	92.7	99.8	90.1	95.4
F3Net	85.2	94.8	87.1	97.5	89.5	99.8	67.1	83.1	73.7	99.6	98.8	100.0	65.4	70.0	51.6	93.6	60.3	99.9	75.4	93.1
SelfBland [32]	63.1	66.1	56.4	59.0	75.1	82.4	79.0	82.5	68.6	74.0	73.6	77.8	53.2	53.9	61.6	65.0	61.2	66.7	65.8	69.7
GANDetection [26]	57.4	75.1	67.9	100.0	67.8	99.7	67.6	92.4	67.7	99.3	60.9	86.2	69.6	83.5	66.7	90.6	69.6	97.2	66.1	91.6
LGrad [35]	68.6	93.8	69.9	89.2	50.3	54.0	71.1	82.0	57.5	67.3	89.1	99.1	78.5	86.0	78.0	87.4	54.8	68.0	68.6	80.8
Ojha [28]	78.5	98.3	72.0	98.9	77.6	99.8	77.6	98.9	77.6	99.7	78.2	98.7	85.2	**98.1**	77.6	98.7	74.2	97.8	77.6	**98.8**
NPR [34]	83.0	96.2	**99.0**	99.8	**98.7**	99.0	**94.5**	98.3	**98.6**	99.0	99.6	100.0	79.0	80.0	88.8	97.4	**98.0**	**100.0**	93.2	96.6
ADOF(ours)	**99.5**	**100.0**	92.2	**100.0**	96.0	99.6	94.1	**99.1**	96.0	99.7	**100.0**	**100.0**	77.5	86.7	94.8	99.3	97.8	99.7	**94.2**	98.2

Table 2. Evaluation results on the test set of DiffusionForensics dataset [37].

Method	**ADM**		**DDPM**		**IDDPM**		**LDM**		**PNDM**		**VQ-Diffusion**		**Stable Diffusion v1**		**Stable Diffusion v2**		**Mean**	
	Acc.	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**
CNNDetection [36]	53.9	71.8	62.7	76.6	50.2	82.7	50.4	78.7	50.8	90.3	50.0	71.0	38.0	76.7	52.0	90.3	51.0	79.8
Frank [10]	58.9	65.9	37.0	27.6	51.4	65.0	51.7	48.5	44.0	38.2	51.7	66.7	32.8	52.3	40.8	37.5	46.0	50.2
Durall [9]	39.8	42.1	52.9	49.8	55.3	56.7	43.1	39.9	44.5	47.3	38.6	38.3	39.5	56.3	62.1	55.8	47.0	48.3
Patchfor [5]	77.5	93.9	62.3	97.1	50.0	91.6	99.5	100.0	50.2	99.9	100.0	100.0	90.7	99.8	94.8	100.0	78.1	97.8
F3Net [29]	80.9	96.9	84.7	99.4	74.7	98.9	100.0	100.0	72.8	99.5	100.0	100.0	73.4	97.2	99.8	100.0	85.8	99.0
SelfBland [32]	57.0	59.0	61.9	49.6	63.2	66.9	83.3	92.2	48.2	48.2	77.2	82.7	46.2	68.0	71.2	73.9	63.5	67.6
GANDetection [26]	51.1	53.1	62.3	46.4	50.2	63.0	51.6	48.1	50.6	79.0	51.1	51.2	39.8	65.6	50.1	36.9	50.8	55.4
LGrad [35]	86.4	97.5	**99.9**	100.0	66.1	92.8	99.7	100.0	69.5	98.5	96.2	100.0	90.4	99.4	97.1	100.0	88.2	98.5
Ojha [28]	78.4	92.1	72.9	78.8	75.0	92.8	82.2	97.1	75.3	92.5	83.5	97.7	56.4	90.4	71.5	92.4	74.4	91.7
NPR [34]	88.6	98.9	99.8	100.0	91.8	99.8	100.0	100.0	91.2	**100.0**	100.0	100.0	97.4	99.8	93.8	100.0	95.3	99.8
ADOF(ours)	**93.5**	**99.0**	99.6	**100.0**	**99.2**	**100.0**	99.9	**100.0**	**97.4**	99.9	97.1	99.8	**99.8**	**100.0**	**99.9**	**100.0**	**98.3**	**99.8**

Table 3. Evaluation results on the diffusion test set of Ojha [28].

Method	**DALLE**		**Glide_100_10**		**Glide_100_27**		**Glide_50_27**		**ADM**		**LDM_100**		**LDM_200**		**LDM_200_cfg**		**Mean**	
	Acc.	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**	**Acc.**	**A.P.**
CNNDetection [36]	51.8	61.3	53.3	72.9	53.0	71.3	54.2	76.0	54.9	66.6	51.9	63.7	52.0	64.5	51.6	63.1	52.8	67.4
Frank [10]	57.0	62.5	53.6	44.3	50.4	40.8	52.0	42.3	53.4	52.5	56.6	51.3	56.4	50.9	56.5	52.1	54.5	49.6
Durall [9]	55.9	58.0	54.9	52.3	48.9	46.9	51.7	49.9	40.6	42.3	62.0	62.6	61.7	61.7	58.4	58.5	54.3	54.0
Patchfor [5]	79.8	99.1	87.3	99.7	82.8	99.1	84.9	98.8	74.2	81.4	95.8	99.8	95.6	99.9	94.0	99.8	86.8	97.2
F3Net [29]	71.6	79.9	88.3	95.4	87.0	94.5	88.5	95.4	69.2	70.8	74.1	84.0	73.4	83.3	80.7	89.1	79.1	86.5
SelfBland [32]	52.4	51.6	58.8	63.2	59.4	64.1	64.2	68.3	58.3	63.4	53.0	54.0	52.6	51.9	51.9	52.6	56.3	58.7
GANDetection [26]	67.2	83.0	51.2	52.6	51.1	51.9	51.7	53.5	49.6	49.0	54.7	65.8	54.9	65.9	53.8	58.9	54.3	60.1
LGrad [35]	88.5	97.3	89.4	94.9	87.4	93.2	90.7	95.1	**86.6**	**100.0**	94.8	99.2	94.2	99.1	95.9	99.2	90.9	97.2
Ojha [28]	89.5	96.8	90.1	97.0	90.7	97.2	91.1	97.4	75.7	85.1	90.5	97.0	90.2	97.1	77.3	88.6	86.9	94.5
NPR [34]	94.5	**99.5**	98.2	99.8	97.8	99.7	98.2	99.8	75.8	81.0	**99.3**	99.9	**99.1**	99.9	**99.0**	99.8	**95.2**	97.4
RINE [21]	**95.0**	99.5	90.7	99.2	88.9	99.1	92.6	99.5	76.1	96.6	98.7	99.9	98.3	99.9	88.2	98.7	91.1	99.0
ADOF(ours)	92.1	98.3	**98.6**	**100.0**	**98.7**	**100.0**	**98.4**	**99.9**	75.9	87.6	98.8	**100.0**	98.6	**99.9**	98.5	**99.9**	94.9	**98.2**

which only reaches 95.3% (see Table 2). It also surpasses DIRE [37], which reports 97.9% accuracy on its own dataset, despite our model being trained on Forensynths [22], in contrast to DIRE's training on DiffusionForensics. Additionally, this approach exceeds both RINE [21] and Ojha [28] (see Table 3), with the latter achieving 91.1% on its dataset [28]. It is worth mentioning that both methods utilize a large CLIP model for their evaluations.

4.4 Computational Efficiency Evaluation

We conduct a comparative analysis of several representative methods based on their computational efficiency and resource requirements. Specifically, we evaluate and compare the following metrics:

- **Number of Parameters:** We quantify the total number of parameters for each model to assess its complexity.

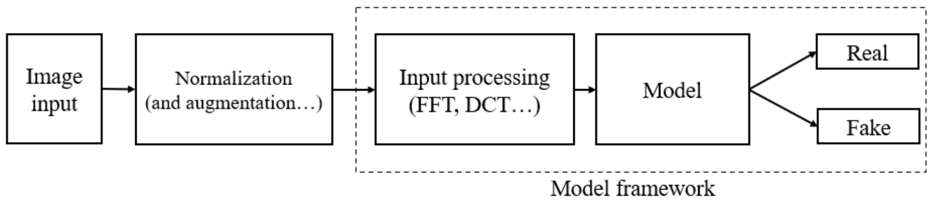
Image input
Normalization (and augmentation…)
Input processing (FFT, DCT…)
Model
Real
Fake
Model framework

Fig. 4. Standard pipeline for Image Synthesis Methods

Table 4. Resource usage and performance of synthetic image detection methods on the DiffusionForensics [37]. The method marked with † indicates it was trained on this dataset.

Method	Parameters	Processing (ms)	Inference Time (ms)	FLOPs	Means (acc/ap)
LGrad [35]	25.56×10^6	11.6	4.81	4.12×10^9	88.2/98.5
DIRE† [37]	25.56×10^6	4,502.7	4.81	4.12×10^9	97.9/**100**
Ojha [28]	427.62×10^6	None	29.19	77.83×10^9	74.4/91.7
ADOF(ours)	1.44×10^6	0.40	**2.43**	$\mathbf{1.74 \times 10^9}$	**98.3**/99.8

- **Input Processing Time:** We measure the time required for processing images before they are fed into the model, where these processing steps are tailored to the specific method used (See Fig. 4).
- **Inference Time:** We record the time taken for the model to process an image and produce a result.
- **FLOPs (Floating Point Operations Per Second):** We leveraged the `fvcore` library to estimate the FLOPs required by each model during inference, providing valuable insights into their computational demands.

Our method requires substantially fewer parameters and FLOPs while achieving faster inference and the highest mean accuracy (98.3%) compared to existing methods, including the DIRE [37] (see Table 4), which is trained on the same dataset but does not achieve comparable performance. This demonstrates its superior performance in synthetic image detection.

5 Conclusion

In this paper, we proposed a simple yet highly effective filter, namely ADOF, for capturing pixel-level variations. By treating an image as a discrete digital signal, this method eliminates the average components of the signal. These components typically carry semantic information, which is less helpful for distinguishing between real and synthetic images compared to the subtle traces that the proposed filter is designed to detect. Experimental results indicates that our proposed method significantly reduces model complexity while enhancing both accuracy and generalization, even on previously unseen data.

Acknowledgment

This research is funded by Vietnam National University - Ho Chi Minh City (VNU-HCM) under Grant Number C2024-18-25.

References

1. Arunachalam, S., Khairnar, S., Desale, B.: The fast fourier transform algorithm and its application in digital image processing. New J Chem **35**(5) (2013)
2. Bauer, L.A., Bindschaedler, V.: Generative models for security: Attacks, defenses, and opportunities. arXiv:2107.10139 (2021)
3. Cara Curtis: California makes deepfakes illegal to curb revenge porn and doctored political videos (2019), `https://bit.ly/4f40oaX`, accessed: 2024-09-24
4. Casteleiro-Pitrez, J.: Generative artificial intelligence image tools among future designers: A usability, user experience, and emotional analysis. Digital **4**(2), 316–332 (2024)
5. Chai, L., Bau, D., Lim, S.N., Isola, P.: What makes fake images detectable? understanding properties that generalize. In: Eur. Conf. on Computer Vision. (2020)
6. Chen, Z., Wang, X.: Application of ai technology in interior design **179** (2020)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. CoRR **abs/2010.11929** (2020)
8. Durall, R., Keuper, M., Pfreundt, F.J., Keuper, J.: Unmasking deepfakes with simple features. ArXiv **abs/1911.00686** (2019)
9. Durall, R., Keuper, M., Keuper, J.: Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions pp. 7890–7899 (2020)
10. Frank, J.C., Eisenhofer, T., Schönherr, L., Fischer, A., Kolossa, D., Holz, T.: Leveraging frequency analysis for deep fake image recognition. ArXiv (2020)
11. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**, 139–144 (2014)
12. Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., Guo, B.: Vector quantized diffusion model for text-to-image synthesis. CVPR pp. 10696–10706 (2022)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition pp. 770–778 (2016)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. ArXiv **abs/2006.11239** (2020)
15. Jeong, Y., Kim, D., Ro, Y., Choi, J.: Frepgan: Robust deepfake detection using frequency-level perturbations. In: AAAI Conference on Artificial Intelligence (2022)
16. Jeong, Y., Kim, D., Min, S., Joe, S., Gwon, Y., Choi, J.: Bihpf: Bilateral high-pass filters for robust deepfake detection. WACV pp. 48–57 (2022)
17. Ju, Y., Jia, S., Ke, L., Xue, H., Nagano, K., Lyu, S.: Fusing global and local features for generalized ai-synthesized image detection. In: ICIP. pp. 3465–3469 (2022)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. ArXiv **abs/1710.10196** (2017)

19. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan pp. 8110–8119 (2020)
20. Korosec, K.: Deepfake revenge porn is now illegal in virginia (2019), `https://techcrunch.com/2019/07/01/deepfake-revenge-porn-is-now-illegal-in-virginia/`, accessed: 24 Sep
21. Koutlis, C., Papadopoulos, S.: Leveraging representations from intermediate encoder-blocks for synthetic image detection. arXiv:2402.19091 (2024)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**, 84 – 90 (2012)
23. Liu, H., Tan, Z., Tan, C., Wei, Y., Wang, J., Zhao, Y.: Forgery-aware adaptive transformer for generalizable synthetic image detection. In: CVPR. pp. 10770–10780 (2024)
24. Liu, Z., Qi, X., Torr, P.H.: Global texture enhancement for fake face detection in the wild. In: CVPR. pp. 8060–8069 (2020)
25. Lomov, I., Makarov, I.: Generative models for fashion industry using deep neural networks. In: ICCAIS. pp. 1–6. IEEE (2019)
26. Mandelli, S., Bonettini, N., Bestagini, P., Tubaro, S.: Detecting gan-generated images by orthogonal training of multiple cnns pp. 3091–3095 (2022)
27. Mickens, R.E.: Difference equations: theory, applications and advanced topics. CRC Press (2015)
28. Ojha, U., Li, Y., Lee, Y.J.: Towards universal fake image detectors that generalize across generative models pp. 24480–24489 (2023)
29. Qian, Y., Yin, G., Sheng, L., Chen, Z., Shao, J.: Thinking in frequency: Face forgery detection by mining frequency-aware clues. ArXiv **abs/2007.09355** (2020)
30. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision pp. 8748–8763 (2021)
31. for Schools, B.: Spotting ai: Knowing how to recognise real vs ai images. `https://elearn.eb.com/real-vs-ai-images/` (2024), accessed: 2024-08-21
32. Shiohara, K., Yamasaki, T.: Detecting deepfakes with self-blended images pp. 18720–18729 (2022)
33. Tan, C., Zhao, Y., Wei, S., Gu, G., Liu, P., Wei, Y.: Frequency-aware deepfake detection: Improving generalizability through frequency space domain learning **38**(5), 5052–5060 (2024)
34. Tan, C., Zhao, Y., Wei, S., Gu, G., Liu, P., Wei, Y.: Rethinking the up-sampling operations in cnn-based generative network for generalizable deepfake detection pp. 28130–28139 (2024)
35. Tan, C., Zhao, Y., Wei, S., Gu, G., Wei, Y.: Learning on gradients: Generalized artifacts representation for gan-generated images detection pp. 12105–12114 (2023)
36. Wang, S.Y., Wang, O., Zhang, R., Owens, A., Efros, A.A.: Cnn-generated images are surprisingly easy to spot... for now pp. 8695–8704 (2020)
37. Wang, Z., Bao, J., Zhou, W., Wang, W., Hu, H., Chen, H., Li, H.: Dire for diffusion-generated image detection pp. 22445–22455 (2023)
38. Wikipedia Contributors: Finite difference (2024), `https://en.wikipedia.org/wiki/Finite_difference`, accessed: 2024-08-21
39. Wootaek Shin, P., Ahn, J.J., Yin, W., Sampson, J., Narayanan, V.: Can prompt modifiers control bias? a comparative analysis of text-to-image generative models. arXiv e-prints (2024)
40. Zhong, N., Xu, Y., Li, S., Qian, Z., Zhang, X.: Patchcraft: Exploring texture patch for efficient ai-generated image detection (2024)