Zero-shot Detection of AI-Generated Image via RAW–RGB Alignment

Supplementary Material

Contents

A. ISP and Post-processing Operations

Tab. 7 illustrates the ISP and post-processing operations utilized in Sec. 3.1 of data generation. Specifically, demosaicing is a mandatory operation, encompassing 13 variants {AAHD, AFD, AHD, AMAZE, DCB, DHT, LINEAR, LMMSE, MAHD, PPG, VCD, VCD-MAHD, VNG}, with algorithm implementations referable to the RawPY [2] library. White balance with variants {Shot, Auto, User} is responsible for adjusting the color temperature of an image to ensure accurate color reproduction. Noise reduction with variants {Full, Light, Off} aims to suppress image noise while preserving details. Highlight mode with variants {Blend, Clip, Reconstruct} deals with the processing of overexposed areas in an image. Tone mapping with variants {Scale, Bright, Auto} is used to map high dynamic range image data to a displayable range. Thus, the total number of options for ISP operations is $13 \times 3 \times 3 \times 3 \times 3 = 1053$. For post-processing operations, JPEG Compression (48 variants), WEBP Compression (48 variants), Gaussian Noise (7 variants), Poisson Noise (81 variants), Blurring (4 variants), and Scaling (4 variants) contribute to a total of 192 combinations.

Table 7. ISP and post-processing operations used in Sec. 3.1.

Category	Operation	Parameter
ISP	Demosaicing	{AAHD, AFD, AHD, AMAZE, DCB, DHT, LINEAR, LMMSE, MAHD, PPG, VCD, VCD-MAHD, VNG}
	White Balance	{Shot, Auto, User}
	Noies Reduction	{Full, Light, Off}
	Highlight	{Blend, Clip, Reconstruct}
	Tone Mapping	{Scale, Bright, Auto}
Post-processing	JPEG Compression	QF $\in [50, 98]$ (step = 1)
	WEBP Compression	QF $\in [50, 98]$ (step = 1)
	Gaussian Noise	Variance $\in [3, 9]$ (step = 1)
	Poisson Noise	Intensity $\in [0.1, 0.5]$ (step = 0.05) & Color Shift $\in [0.01, 0.05]$ (step = 0.005)
	Blurring	Kernel Size $\in [3, 9]$ (step = 2)
	Scaling	{Linear, Nearest, Area, Cubic}

B. Zero-/Few-shot Scenario Details

B.1. Zero-shot Scenario

After training, our model implements zero-shot detection by extracting alignment traces from test images and using these traces for authenticity judgment. This process can be further divided into macro and micro cases.

As shown in Fig. 4(a), in the macro case with N test samples, we first extract their anchor features $\{F_t\}_{t=1}^{N}$, then perform unsupervised clustering like K-Means to obtain clustering results $\mathbf{C} \in \mathbb{R}^N$. The performance is evaluated by measuring the match between $\mathbf{C}$ and the ground-truth labels $\mathbf{L} \in \mathbb{R}^N$. For example, Normalized Mutual Information (NMI) can quantify this alignment by:

$$\mathrm{NMI}(\mathbf{C}, \mathbf{L}) = \frac{2 \cdot \mathrm{MI}(\mathbf{C}, \mathbf{L}}{H(\mathbf{C}) + H(\mathbf{L})}, \tag{17}$$

where $\mathrm{MI}(\cdot, \cdot)$ is the mutual information defined by:

$$\mathrm{MI}(\mathbf{C}, \mathbf{L}) = \sum_{i=1}^{|\mathbf{C}|} \sum_{j=1}^{|\mathbf{L}|} \frac{|\mathbf{C}_i \cap \mathbf{L}_j|}{N} \log \frac{N|\mathbf{C}_i \cap \mathbf{L}_j|}{|\mathbf{C}_i||\mathbf{L}_j|}. \tag{18}$$

NMI ranges from 0 to 1, with higher values indicating better separation of real and synthetic images in the clustering result.

As shown in Fig. 4(b), in the micro case with limited samples, we determine the authenticity of a test image by computing the similarity between its anchor feature $\mathbf{F}_t$ and reference real $\mathbf{F}_{\text{Real}}$ (or synthetic $\mathbf{F}_{\text{Syn}}$) feature. The authenticity probability can be concisely expressed as:

$$P(\text{Syn} \mid F_t) = \frac{\cos(\mathbf{F}_t, \mathbf{F}_{\text{Syn}})}{\cos(\mathbf{F}_t, \mathbf{F}_{\text{Real}}) + \cos(\mathbf{F}_t, \mathbf{F}_{\text{Syn}})}, \tag{19}$$

where values close to 1 indicate a high likelihood of being synthetic.

B.2. Few-shot Scenario

While prior GenAI data is not mandatory for our framework, its availability enables fine-tuning to boost detection performance further, as shown in Fig. 4(c). Specifically, given a labeled dataset of real and fake images, we introduce an additional trainable linear classifier $f_{\text{LC}} : \mathbb{R}^d \to \{0, 1\}$ while freezing all parameters of the pre-trained backbone network. In this linear probing setup, anchor features extracted by

Table 8. Zero-shot evaluation on modern generators.

Method	SD3.5		SDXL		FLUX		Kolors	
	NMI ↑	AUC ↑	NMI ↑	AUC ↑	NMI ↑	AUC ↑	NMI ↑	AUC ↑
ZED	.438	.527	.785	.819	.653	.749	.725	.816
FSD	.616	.554	.898	.886	.829	.843	.846	.860
MIB	.816	.833	.859	.864	.812	.808	.832	.902
Ours	**.885**	**.914**	**.956**	**.939**	**.901**	**.942**	**.922**	**.951**

Table 9. Comparison with pre-trained and RIGID.

#	Method	Backbone	Zero-shot			Few-shot
			NMI ↑	AUC ↑	AP ↑	AP ↑
#1	Pre-trained	CLIP	.626	.729	N/A	.914
#2		DINOv2	.660	.686	N/A	.935
#3	RIGID [21]	CLIP	.876	.844	.843	.914
#4		DINOv2	.909	.872	**.874**	.935
#5	**Ours**	CLIP	**.964**	**.925**	N/A	**.987**

the frozen backbone are fed into f_{LC}, which learns to map these features to binary authenticity labels. The fine-tuning process optimizes θ using cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\mathbb{E}\left[y_t \log(f_{\text{LC}}(\mathbf{F}_t)) + (1-y_t)\log(1-f_{\text{LC}}(\mathbf{F}_t))\right], \tag{20}$$

where $y_t \in \{0,1\}$ denotes the ground-truth label. This approach efficiently adapts the model to specific data distributions without retraining the entire backbone, balancing performance gains and computational cost.

C. Implementation Details

This work is implemented on the Linux system using the PyTorch framework, with H800 GPUs as the hardware. AdamW [23] with default parameters is adopted as the optimizer for training. In data generation, we sample $I = 8$ RAW $\mathbf{R}_i$ for each mini-batch and apply $J = 8$ pipelines P_j to these $\mathbf{R}_i$ respectively, generating a total of 64 $\mathbf{I}_{i,j}$. Considering the GPU memory constraint, we randomly crop a 224×224 region at the same spatial position from both $\mathbf{R}_i$ and $\mathbf{I}_{i,j}$ as the actual input. In trace definition, we set the dimension d of the alignment trace $\mathbf{F}_j$ to 512. In the RGB-Vision branch, f_{LVM} is selected as ViT-L/14 trained under the CLIP paradigm [43], and the extracted intermediate feature $\mathbf{H}$ contains 256 tokens. For f_{SEA}, the entropy binning is based on a patch size $s = 13$ and a split number $B = 10$; in stratified sampling, $M = 8$ tokens are uniformly selected from each distinct bin to form the subset $\hat{\mathcal{T}}^b$. In the RAW-Graph branch, continuous-value operations (*e.g.*, JPEG's QF ranging from 50 to 98) are normalized to the range [0, 1], while discrete-value operations (*e.g.*, 13 algorithms in Demosaicing) are one-hot encoded. In the RAW-Vision branch, the UNet from Stable Diffusion [48] is chosen as the encoder $\text{Enc}(\cdot)$. Finally, in the objective function Eq. (16), we set $\lambda_1 = 1$ and $\lambda_2 = 5$ to balance the contribution magnitudes from both the graph and vision aspects.

D. Additional Experiments

D.1. Evaluation on Modern Generators

Tab. 8 reports zero-shot results of competitors on DiffSeg30k [15] with four modern generators (SD3.5, SDXL, FLUX, Kolors). The competitor FSD achieves 0.886/0.843 AUC on SDXL/FLUX but only 0.554 on SD3.5 due to poor generalization. In contrast, our method shows superior stability, reaching over 0.9 AUC on all generators, verifying the effectiveness of RAW–RGB aligned traces.

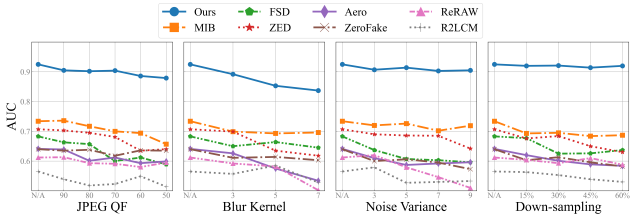

Figure 7. Robustness to common image operations.

D.2. Evaluation on More Baseline

#1/#2 in **Tab. 9** show pre-trained CLIP [43] and DINOv2 [42] achieve limited zero-shot synthetic detection with AUC < 0.68 as they focus on high-level semantics. Thus, methods targeting low-level artifacts/traces/perturbations (*e.g.*, RIGID [21]) may be more suitable for this task. To this end, #3/#4 in **Tab. 9** evaluate RIGID with CLIP/DINOv2 backbones in zero-/few-shot settings. Its noise-perturbed approach boosts zero-shot AUC from 0.686 to 0.872, yet our proposed RAW–RGB alignment achieves 0.925 AUC in the same scenario and also outperforms in other testing. However, RIGID's key strength lies in its training-free design, and its low-level noise patterns will be discussed and compared in detail in the revision.

D.3. Robustness

Fig. 7 evaluates competitors' robustness to JPEG, Gaussian blur/noise, and downsampling. Our method resists these operations well but is slightly weaker against blur, likely due to a lack of blur coverage in the RAW-to-RGB process, causing potential misclassification of blurred real images as synthetic. We will prioritize defining real/synthetic categories for post-processed images in future work.