In this appendix, we add some experiments and more details for a better understanding of our method. Section A, We investigate the generalization capability of fake image subsets produced by different generators and visualize them in the feature space in Figure 8, *highlighting the proper position of learned* α_0; Section B reveals the changing behavior of the baseline against the consistent behavior of our detector when dealing with various unseen generators' images. Section C provides comprehensive implementation details for our mixup training and voting inference strategies; Section D presents a qualitative visualization of our method better discriminated image samples; Section E conducts sensitivity analysis of hyperparameters; Section F visualizes the most similar image samples to learned artifacts/traits embeddings. Section G test our model on the benchmark Chameleon [48].

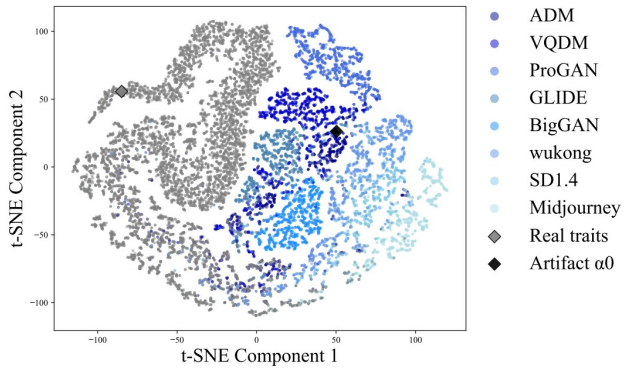


Figure 8. The t-SNE visualization of image features extracted from in-domain generators and real images within the training set using our approach. It also includes diamond-shaped markers for the learned generic artifact embedding and real trait embedding. Gray dots represent real images, while blue dots denote generated images. For better clarity, the depth of the blue color indicates the strength of the generalization ability according to Table 7.

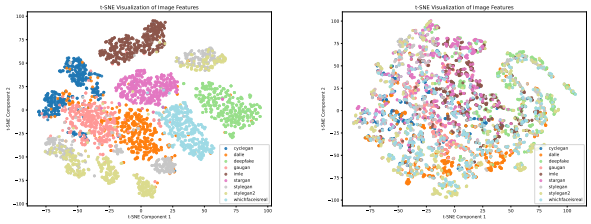
A. Generic Artifacts Embedding in a Proper Generalized Feature Space

We evaluated the generalization ability of each generator's fake image subset and visualized them in Figure 8. Specifically, we train a discriminator on each generator's image subset respectively and test the OOD accuracy on the validation set to measure the generalization ability in Table 7.

As depicted in Figure 8, we reveal that generators exhibiting robust generalization ability, such as ADM and VQDM, produce image features that cluster closely around the generic artifact labeled as α_0. In contrast, generators with poorer generalizations, like Midjourney, show features that diverge more significantly from α_0. This observation indicates that the learned α_0 represents a proper generalized artifact feature, in line with our initial design intentions.

Table 7. Results of out-of-domain (OOD) accuracy on the validation set for detectors trained on different generators' image subsets.

Generator	ADM	VQDM	ProGAN	Glide	BigGAN	WuKong	SD14	Midjourney
OOD acc.	0.782	0.769	0.736	0.713	0.700	0.618	0.592	0.524



(a) Baseline (b) Ours

Figure 9. The t-SNE visualization of the image features extracted from **out-of-domain** generators in the test set by baseline and Ours. **Our method extracts common generic artifacts from unseen generators regardless of different generator types or image semantics.** Features in different colors denote different generators.

B. Common Artifacts Extraction from Unseen Generators

Section 4.2 presents the detector's behavior on in-domain generators. To further investigate the detector's generalization capability, we visualize the features of images generated by unseen generators in Figure 9. Ideally, the detector should extract common generic artifacts from unseen generators' images as designed, unaffected by generator types or image semantic variations, thereby achieving stable discrimination. However, as shown in Figure 9 (a), the original detector exhibits varying mapping behaviors across different unseen generators, resulting in the highly fluctuating discrimination thresholds discussed in Section 4.5. In contrast, our method consistently extracts uniform features from different generator images as intended in Figure 9 (b). We attribute this capability as a key factor in our method's improved generalization performance.

C. Details of Mixup Training and Vote Inference

Mixup Training for Pretrain-based Methods. Conventional mixup [54] involves blending samples across different classes within the raw image distribution. Our approach, however, diverges in two key ways: 1. As the backbone of pretrain-based methods remains frozen, we apply mixup directly to the backbone's extracted image features $\tilde{x}_i$, $\tilde{x}_j$ from i-th, j-th generator, respectively:

$$\tilde{x}_{i,j} = \lambda \tilde{x}_i + (1-\lambda)\tilde{x}_j, \tag{3}$$

where λ is a randomly sampled value within the interval [0, 1]. 2. We independently perform mixup within the sets of real and synthetic images. In terms of specific implementation, the samples in a batch are reversed. If the pre- and post-reversal samples are both real or synthetic, they are subject to mixup with a certain probability.

Accordingly, our training objective is formalized by:

$$\mathcal{L}_{mixup} = \lambda \mathcal{L}(MLP(\tilde{x}_{i,j}), i) + (1-\lambda)\mathcal{L}(MLP(\tilde{x}_{i,j}), j), \tag{4}$$

where $\mathcal{L}$ is the proposed AC Loss as in Equation (1), MLP is the learnable head to extract the final image feature.

Vote Inference for Learning-based Methods. In this method, we utilize all the learned artifacts $\{\alpha_i\}_{i=0}^{I}$ during the evaluation process. Through Equation (2), we assess the input score s using both the learned artifacts $\{\alpha_i\}_{i=0}^{I}$ and the real traits α_{I+1}, obtaining $I+1$ scores denoted as $\{s_i\}_{i=0}^{I}$ for the input. By statistics of the in-domain and real image distribution in the training set, the scores can be converted to directly comparable probabilities. Specifically, we calculate scores, s_i^n, for 10,000 images generated by the i-th generator as well as 10,000 real images using α_i and α_{I+1}. The probability corresponding to a given score s under the bandwidth ϵ is approximated as

$$P(y|s) = \frac{|\{s_i^n \mid s_i^n \in (s-\epsilon, s+\epsilon) \wedge y_i = Fake\}|}{|\{s_i^n \mid \{s_i^n \in (s-\epsilon, s+\epsilon)\}|}. \tag{5}$$

If the number of samples within this score range is less than 10, we gradually increase ϵ. The adjustment rule for the t-th iteration is given by $\epsilon_t = \epsilon_0 + \epsilon_{t-1}$, with $\epsilon_0 = 0.001$.

D. Visualization of Results.

We visualize representative cases where our method succeeds in correct classification while UFD fails in Figure 10. The top row shows generated images that our method accurately identified as fake, spanning diverse categories including indoor scenes, vintage vehicles, animals, architecture, and everyday objects. These cases are particularly challenging as they exhibit high visual fidelity - for instance, the generated bedroom (leftmost) demonstrates photorealistic lighting and spatial composition that caused UFD to misclassify it as real. The bottom row shows natural images that our method correctly classified as real, highlighting our model's ability to preserve high true positive rates while reducing false positives. These results demonstrate our method's enhanced discrimination capability across varied content and lighting conditions, even in cases where traditional detectors struggle with highly realistic synthetic images.

E. Sensitivity of AC Loss to Hyperparameters.

Sensitivity of AC Loss to σ. The coefficient σ in AC Loss is a hyper-parameter to enhance mutual exclusivity among artifacts. We examined the sensitivity of the detector's performance to various values of σ. As shown in Table 8, the model achieved its highest out-of-distribution (OOD) accuracy of 0.861 when σ was set to 10. Since the mutual exclusivity exists in both terms in the AC Loss's denominator, the performance is relatively stable across different σ values.

Table 8. Results of Ours (UFD) under different values of σ on the validation set.

σ	Validataion					
	In-domain		Out-of-domain		Total	
	Mean acc.	AP	Mean acc.	AP	Mean acc.	AP
0	0.955	0.991	0.855	0.936	0.905	0.962
0.1	0.953	0.989	0.852	0.901	0.903	0.941
1	0.952	0.991	0.854	0.929	0.903	0.957
10	0.952	0.989	0.861	0.922	0.907	0.951
100	0.953	0.990	0.852	0.930	0.903	0.957

Table 9. Results of Ours (UFD) under different values of τ on the validation set.

τ	Validataion					
	In-domain		Out-of-domain		Total	
	Mean acc.	AP	Mean acc.	AP	Mean acc.	AP
0.05	0.945	0.984	0.830	0.846	0.887	0.918
0.1	0.946	0.986	0.835	0.888	0.891	0.938
0.5	0.952	0.989	0.861	0.922	0.907	0.951
1.0	0.931	0.982	0.840	0.937	0.885	0.955
1.5	0.925	0.980	0.832	0.931	0.878	0.950

Sensitivity of AC Loss to τ. In contrastive learning, the temperature coefficient τ significantly influences the model's ability to distinguish between positive and negative pairs. Results in Table 9 show that performance peaks at a temperature of 0.5. A lower temperature (e.g., 0.05) sharpens the similarity distribution, risking overfitting, while a higher temperature (e.g., 1.5) overly smooths it, reducing discriminative power. The optimal value of 0.5 balances these effects, enabling robust and effective learning.

F. Visualization of Representative Images to Artifacts/traits

While the optimized embeddings effectively represent corresponding artifacts/traits, their representation on images remains non-intuitive. To visualize this, we showcase the top 10 test-set images most similar to each embedding in Figure 11 and Figure 12. The results show that the real traits embedding associates with real images exhibiting non-uniform scaling, whereas the generic artifacts embedding correlates with abnormal blurred or flat backgrounds. Generator-specific embeddings further reflect unique characteristics. For instance, VQDM-specific artifacts' embedding links to smaller/blurred objects, and Wukong-specific artifacts' embeddings match highly saturated images. This shows that they tend to expose their specific artifacts in

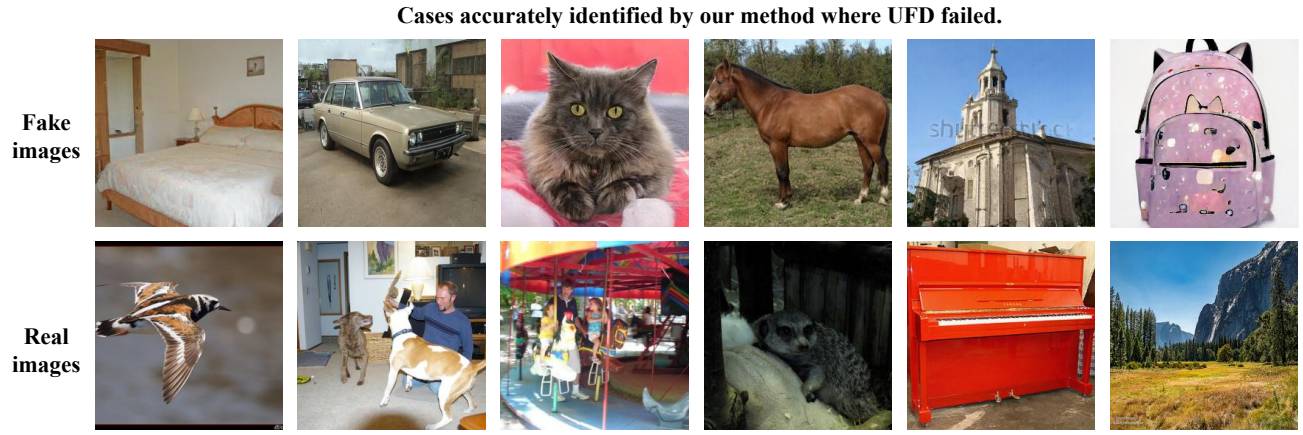


Figure 10. The visualization of fake (top row) and real (bottom row) images in the out-of-domain test set. These images represent cases accurately identified by our method where UFD failed.

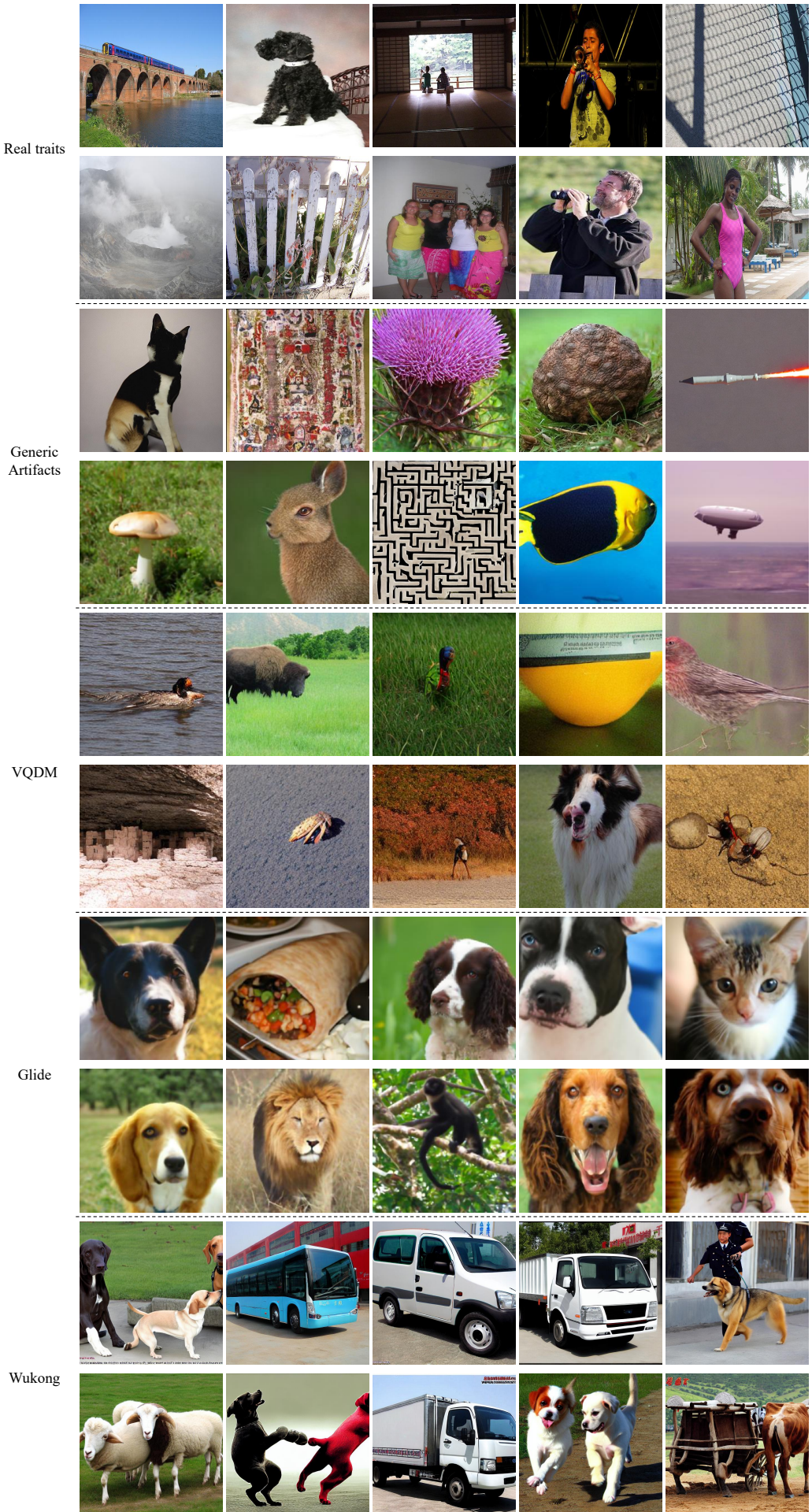
these images, respectively.

G. Cross Benchmark Validation.

We attain SOTA on AIDE's challenging benchmark Chameleon [48] in the Table 10.

Table 10. Cross-benchmark validation on Chameleon [48]. Best results are bolded for clarity.

Dataset	CNNDet	UFD	AIDE	Ours-CNN	Ours-UFD
Chameleon	0.609	0.604	0.658	0.647	**0.664**
Ours OOD	0.679	0.829	0.779	0.762	**0.855**


Real traits
Generic Artifacts
VQDM
Glide
Wukong


ProGAN
Midjourney
SD 1.4
ADM
BigGAN