AdaParse: Personalized Fingerprinting for Visual Generative Model Reverse Engineering

Yu Zheng, Zhuoxun Li, Bingyao Yu, Jie Zhou, *Fellow, IEEE*, Jiwen Lu, *Fellow, IEEE*

***Abstract*—In this paper, we propose a hyperparameter-specialized adaptive fingerprinting framework named AdaParse for model reverse engineering, which aims at predicting hyperparameters of interest in generative models from the given AI-generated images. Existing methods rely on a single coarse model fingerprint that is originally designed for model-level attribution, which makes it difficult to distinguish fine-grained traces corresponding to different hyperparameter configurations in a multitude of generative models. To address this, our AdaParse dynamically responds to instance-level variations by estimating hyperparameter-specific fingerprints via personalizing estimation networks tailored for each input image. Specifically, our approach simultaneously learns two-branch hypernetworks that balance instance-aware and model-agnostic prior knowledge for fingerprint generation. To enable efficient network personalization, we further propose a Broadcasted Fusion module that transforms condensed feature codes into adaptive parameters through factorized weight generation with enhanced representative capacity. Extensive experiments on the large-scale public dataset across 123 generative models demonstrate that our approach outperforms previous state-of-the-art methods. Code available at https://github.com/lizhuoxun/AdaParse/.**

***Index Terms*—Generative model parsing, reverse engineering, dynamic network**

I. Introduction

Advancements in deep learning have led to the explosive development of various generative models such as Variational Auto-Encoders (VAEs), Generative Adversarial Networks (GANs) and Diffusion Models (DMs) [1]–[9], which generate increasingly authentic visual data that raise concerns about their potential misuse by malicious attackers. Although the deepfake detectors [10]–[16] are designed to effectively distinguish real from synthetic content, this capability alone does not address the need to identify source generative models for potential legal measures. Therefore, understanding the sources of such AI-generated visual content, referred to as model parsing, is crucial for subsequent in-depth analysis such as style attribution [17] and ownership identification [18].

Emerging researches have explored model parsing by identifying the original generative model types, either through active watermark embedding during the training phase [20]–[24] or passive analysis of the given AI-generated images [20], [25]–[35]. However, these model attribution approaches merely classify generated content into predefined model-level categories, which provide limited insights into the underlying generative process. Therefore, a comprehensive reverse engineering approach [19], [36]–[45] that recovers critical hyperparameters[1] of interest such as loss functions and network architectures is essential for thorough forensic analysis. These parameters not only reveal the fundamental design principles of the target models, but also provide technical specificity that strengthens attribution evidence beyond basic model classification. In the field of deep learning, the reverse engineering technique has been actively explored for conventional visual understanding models [36], [37], [39], [42], [43], [46] and text-based generation models [38], [40], [41], [47]. For example, [39] demonstrates that the structural information (layers and hyperparameters) of a DNN model can be inferred remotely by programming ring oscillators on FPGA. [47] successfully reveals the hidden dimensions of production language models (e.g. `ada` and `babbage` models of OpenAI) with less than

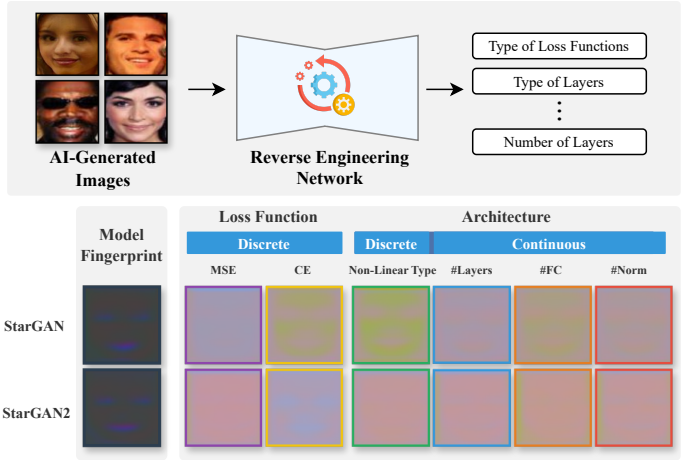


Fig. 1: **Upper:** Our problem setup. Given AI-generated images, we aim to conduct a reverse engineering to infer model hyperparameters from the corresponding generative models. **Bottom:** Comparison between the model fingerprint [19] and our hyperparameter fingerprint–while model-level fingerprints [19] (left column) provide only coarse-level signals that fail to distinguish between different hyperparameter configurations, our proposed design enables the generation of multiple hyperparameter-specific fingerprints (right columns) that isolate the subtle artifacts associated with different architectural choices and training objectives.

Yu Zheng and Zhuoxun Li contributed equally to this work.
Corresponding author: Jiwen Lu.
The authors are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: yu-zheng@tsinghua.edu.cn, li-zx22@mails.tsinghua.edu.cn, yuby@tsinghua.edu.cn, jzhou@tsinghua.edu.cn, lujiwen@tsinghua.edu.cn.

[1]In generative models, hyperparameters refer to the configuration settings that are predefined before model training and control aspects of the architecture or learning process. They are not learned from data but are set by the designer. In contrast, model parameters (e.g., weights and biases in neural networks) are internal variables learned during training.

$20 USD. However, in the rapidly expanding domain of visual generative models, such in-depth analysis on source models remains underexplored.

Recently, Asnani et al. [19] pioneered the reverse engineering of visual generative models by inferring critical model hyperparameters from generated images (as displayed by the problem setup in the upper part of Fig. 1) and constructing a corresponding dataset benchmark. Based on the observation that many generative models inherently leave stable fingerprints associated with specific model types [12], [20], [25], [27], [28], [31], [48], they extract generative model patterns through a Fingerprint Estimation Network (FEN) with physical constraints, and cluster these patterns to predict the network architectures and loss functions of interest. While groundbreaking, their approach relies on a single global fingerprint extraction mechanism that struggles to capture the nuanced relationships between specific hyperparameters and their corresponding visual artifacts. As shown in the bottom part of Fig. 1, model fingerprints (left column) provide only coarse-level signals that fail to distinguish between different hyperparameter configurations, especially when generative models leave similar fingerprints (StarGAN [49] and StarGAN2 [50] in this case). This limitation becomes even more significant when analyzing the diverse architectures and training objectives across a multitude of generative models, resulting in inferior parameter estimation accuracy. Recent studies [27], [51] have shown that hyperparameters in generative models leave identifiable visual traces in generated images. For instance, Yang et al. [27] demonstrated that network architectures leave globally consistent fingerprints. Tan et al. [51] analyzed how up-sampling operations introduce artifacts. These artifacts are often consistent across images from the same model configuration, enabling hyperparameter-level fingerprinting.

In this paper, we propose a hyperparameter-specialized adaptive fingerprinting framework named AdaParse for model reverse engineering, which dynamically personalizes the hyperparameter estimation process for each input image. Unlike previous approaches that rely on a single global fingerprint, our method captures fine-grained hyperparameter-specific traces by generating tailored fingerprint estimation networks for each image sample. Specifically, we design a two-branch encoder architecture that balances instance-aware and model-agnostic information through complementary pathways, i.e., an Instance-Aware (IA) Encoder that captures unique characteristics of each image and a Model-Agnostic (MA) Encoder that isolates hyperparameter-specific signals through hierarchical latent subspaces. These representations are integrated via a Broadcasted Fusion module, which efficiently transforms the condensed codes into personalized network parameters and prevents the excessive computational overhead of generating entirely distinct networks for each hyperparameter-image pair. This design enables the generation of multiple hyperparameter-specific fingerprints (as shown in the right columns of Fig. 1) that isolate the subtle artifacts associated with different architectural choices and training objectives, which promotes to reveal distinctive patterns for each target hyperparameter rather than merged global signatures. Extensive experiments on the large-scale benchmark [19] including 123 generative models demonstrate that our AdaParse significantly outperforms previous state-of-the-art methods, with particularly notable improvements in the reverse engineering of hyperparameters of interest including loss function and network architecture predictions.

II. Related Work

In this section, we briefly review the research topics on : 1) Visual Content Generation, 2) AI-generated Visual Content Detection, 3) Model Parsing and Reverse Engineering, and 4) Sample-specific Dynamic Networks.

A. Visual Content Generation

The advancements in deep learning have enabled the generative models to create abundant and authentic visual data. For example, Variational Auto-Encoders [1]–[3] (VAE) encode the input data into a compressed latent space and then decode it back to the original space. The latent space usually follows the Gaussian distribution allowing for smooth interpolation and meaningful sample generation. Generative Adversarial Networks [4]–[6] (GAN) consist of a generator that creates fake images and a discriminator that tries to distinguish them. Diffusion models [7]–[9] (DM) generate images by reversing a gradual noise-adding process, which have gained attention for the ability to produce high-quality images with fewer artifacts. Other generative models include Auto-Regressive [3], [52], [53] (AR) and Normalizing Flow [54], [55] (NF) models, etc.

B. AI-generated Visual Content Detection

With the proliferation of the AI-generated contents (AIGC) in recent years, preventing its misuse and abuse has become increasingly critical to safeguard societal security. In computer vision, deepfake detection is typically formulated as a binary classification problem aimed at distinguishing between authentic and artificially generated visual content. Early detection methods focus on identifying anomalies in biological signatures, such as unnatural blinking patterns [10] or facial movements [56]. In the era of deep learning, researchers developed various approaches for extracting spatial-domain features using specialized neural architectures. These include convolutional networks [11], [14], [57], auto-encoders that learn compressed representations of authentic images [58], capsule networks that capture spatial relationships [59], [60], and attention-based architectures that focus on discriminative regions [61]–[63]. While spatial methods dominate the field, several studies highlight the effectiveness of detecting deepfake artifacts in the frequency domain [12], [64]–[66], where manipulated images often exhibit distinctive spectral signatures. Besides, researchers have expanded the focus from face forgeries to comprehensive natural image forgery detection, developing advanced frameworks that address diverse manipulation techniques across various domains to enhance generalization and robustness [67], [68]. Complementary research has explored optimal training strategies, including data augmentation techniques, adversarial training approaches, and continual learning frameworks to improve detection robustness [69]–[72]. Beyond simply identifying manipulated content, a subset of research aims to precisely localize forgery regions

within images. These forgeries typically result from specific manipulation techniques such as copy-move operations [73]–[75], splicing content from multiple sources [76]–[78], and in-painting to fill removed areas [79], [80]. The information revealed by these localization methods is particularly valuable yet straightforward, as they typically produce binary segmentation masks that precisely indicate manipulated regions, as demonstrated in several benchmark studies [58]–[62], [81], [82]. In recent years, advanced generation models including DM and AR have received increasing interest [83]. Leveraging large multimodal models, researchers have developed more comprehensive and interpretable methods for detecting synthetic content produced by such advanced models [84].

C. Model Parsing and Reverse Engineering

To facilitate ownership identification and potential legal measures, researches have developed model parsing techniques by identifying the origins of AI-generated content, which can be divided into two categories. Active methods embed artificial fingerprints during the training phase [20]–[22], [24], which requires direct access to the generative model itself and therefore limits their usability in black-box scenarios. In contrast, passive methods identify source models solely from their outputs [28], [30]–[32], [35]. While these approaches effectively classify content by model type, a more comprehensive forensic analysis requires model reverse engineering to infer critical hyperparameters of interest such as loss functions and network architectures [36]–[43], [45], [47], either through hardware-based techniques exploiting information leaks from side-channel [39] and unencrypted PCIe buses [85], or software-based model-stealing attacks [47], [86] where the stolen models achieve the tradeoff between output accuracy and functional fidelity. More relevant to our work, Asnani et al. [19] pioneered reverse engineering of the rapidly-growing visual generative models based on model-level fingerprints and correspondingly constructed a comprehensive benchmark. Recently, Guo et al. [44] introduced LGPN, which formulates model parsing as a graph node classification problem to capture hyperparameter dependencies explicitly. In contrast to these approaches, our proposed AdaParse dynamically responds to instance-level variations by estimating hyperparameter-specific fingerprints through personalized networks tailored to each input image.

D. Sample-specific Dynamic Networks

Given the abundant and unpredictable variations among input samples beyond the inter-category differences, researches have developed approaches to process inputs in a sample-specific manner, which can be divided into two groups. The first strategy dynamically adjusts network architectures to allocate computational resources based on sample complexity. For instance, CALPA-NET [87] implements adaptive pruning schemes that selectively reduce convolutional layers according to data-driven requirements. The second strategy maintains consistent computational graphs while adapting network parameters for each input sample. For comprehensive coverage of related work, we refer readers to the literature review in [88]. Notably, Hypernetworks [89] have been utilized to generate target network weights conditioned on specific input embeddings, with successful applications across image classification [89], semantic segmentation [90], 3D reconstruction [91], and natural language processing [89]. Our work aligns with the second strategy of adapting sample-specific network parameters, where we design a series of hyperparameter-aware Hypernetworks [89] to personalize the fingerprint estimation network in the parameter space with stronger representative ability.

III. Approach

In this section, we demonstrate our proposed approach by firstly introducing our problem formulation and then elaborating the technical details of AdaParse. TABLE I summarizes the key mathematical symbols used in this section for reference.

TABLE I: Definitions of key mathematical symbols.

Symbol	Definition
λ	Loss weight balancing $\mathcal{L}_{cons}$ and $\mathcal{L}_{pred}$
N^l	Number of loss function hyperparameters
N^a	Number of network architectural hyperparameters
N_{hp}	Total number of hyperparameters ($N^l + N^a$)
$C.$	Channel dimension of some variable
$\mathbf{y}$	Hyperparameters to predict
$\mathbf{X}$	Input image generated by a generative model
$\mathbf{F}_{latent}$	Latent fingerprint representation output from shared encoder
$\mathbf{F}^i_{hyp}$	Fingerprint for the i-th hyperparameter
z_{it}	Instance encoding capturing instance-specific characteristics
z_b	Base encoding invariant to models or hyperparameters
z^i_{hp}	Hyperparameter encoding for the i-th hyperparameter
$\mathbb{Z}_{MA}$	The hierarchical set of model-agnostic learnable encodings
$\mathcal{H}_0$, $\mathcal{H}_1$	The root and specialized levels of $\mathbb{Z}_{MA}$
$\mathbf{Z}$	Fused encoding after broadcasted fusion
$\mathbf{W}^i$	Personalized weight parameters for the i-th FEN
$\mathbf{U}^i$, $\mathbf{V}^i$	Projection matrices in weight generation
$\mathcal{L}_{cons}$	Fingerprint constraint loss
$\mathcal{L}_{pred}$	Hyperparameter prediction loss
$f(\cdot)$	Shared convolutional feature extractor
$\mathcal{E}_{shared}(\cdot)$	Shared encoder for preliminary fingerprint extraction
$\mathcal{H}_{IA}(\cdot)$	Instance-aware hypernetwork for instance-specific encoding
$\mathcal{H}_{WG}(\cdot)$	Weight generation hypernetwork for transformation of $\mathbf{Z}$
$\mathcal{E}_{\cdot,\mathbf{W}^i}(\cdot)$	Personalized FEN parameterized by W^i
$\mathcal{D}_{parse}(\cdot)$	Fingerprint parsing network

A. Problem Formulation and Overview

Problem Formulation: Given an image $\mathbf{X} \in \mathbb{R}^{h\times w\times 3}$ generated by a source visual generative model, we aim to predict the hyperparameters $\mathbf{y}$ of interest in the source model [19]:

$$\mathcal{M}_\Theta : \mathbf{X} \longrightarrow \mathbf{y} \tag{1}$$

where Θ denotes the learnable parameters of the predicting network. The hyperparameters $\mathbf{y}$ of interest in our reverse engineering task typically include both loss function hyperparameters (e.g., whether L1 or MSE loss was used during training) and architectural hyperparameters (e.g., number of layers, types of normalization and non-linear layers) of the generative model. Following the pioneering framework established in [19], we consider $N^l = 10$ loss function hyperparameters and $N^a = 15$ network architectural hyperparameters. The architectural hyperparameters consist of both continuous variables

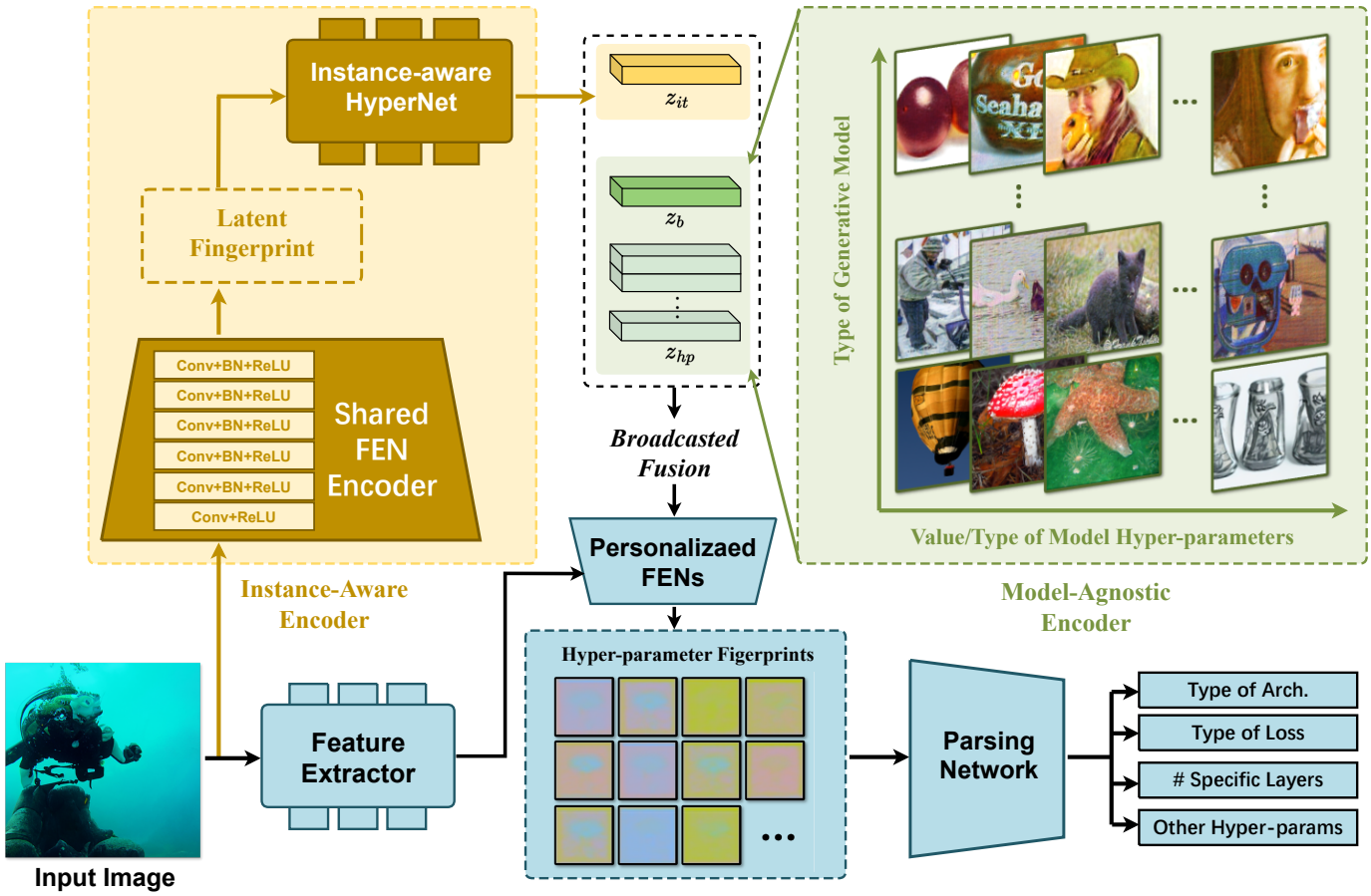


Fig. 2: **Framework of our proposed AdaParse.** AdaParse comprises an Instance-Aware (IA) Encoder and a Model-Agnostic (MA) Encoder. The outputs of the two encoders are integrated into a series of fingerprint estimation networks (FENs) personalized for each input image via a designed Broadcasted Fusion module. The resulting FENs consume the extracted features of the input image and generate the hyperparameter-specific fingerprints, which are then parsed into the estimated hyperparameters such as number of layers and type of network architecture, loss function, etc.

(such as normalized layer counts in $[0, 1]$) and discrete categorical variables (such as the specific normalization function employed). This comprehensive hyperparameter estimation reveals the underlying configuration and design choices of visual generative models, thus enabling more precise reverse engineering of how specific AI-generated images were created.

Our AdaParse: To address this challenging problem, a classic solution is to examine the noise patterns in generated images to identify model-specific fingerprints. Previous work [19] employs a shared fingerprint estimation network followed by a parsing network to predict target hyperparameters $\mathbf{y}$. However, as already displayed in Fig. 1, its reliance on global fingerprint estimation [19], [27] limits the ability to distinguish subtle variations caused by different hyperparameter configurations. In contrast, our method dynamically responds to instance-level variations by personalizing the hyperparameter-specific fingerprint estimation network for each input image, which uncovers the fine-grained traces critical for in-depth model reverse engineering.

As shown in Fig. 2, our method achieves the aforementioned personalized hyperparameter-specific fingerprint estimation through a dual-encoder architecture: an Instance-Aware (IA) Encoder and a Model-Agnostic (MA) Encoder. The IA Encoder attends to image-specific information unique to each sample, while the MA Encoder extracts sharable model traces across hierarchical subspaces that enjoy greater flexibility and expressiveness. These complementary representations are integrated via a Broadcasted Fusion module to create personalized fingerprint estimation networks (FENs) for each input image. The resulting FENs process the extracted image features to generate image-specific hyperparameter fingerprints, which are subsequently parsed to determine target hyperparameters such as network architecture type, loss function, and layer count.

B. Instance-Aware and Model-Agnostic Encoders

Our framework employs a complementary dual-encoder architecture to capture the nuanced variations in generative model hyperparameter configurations across different input samples.

Instance-Aware (IA) Encoder: As displayed in the left-up part of Figure 2, the Instance-Aware (IA) Encoder follows a coarse-to-fine extraction strategy. Initially, we leverage a conventional approach by employing a shared encoder to extract a preliminary fingerprint representation:

$$\mathbf{F}_{latent} = \mathcal{E}_{shared}(I), \tag{2}$$

where $\mathcal{E}_{shared}(\cdot)$ is implemented as the encoder part of a Denoising CNN (DnCNN) [92] architecture. A standard DnCNN consists of an initial Conv+ReLU layer, followed by multiple convolutional blocks (Conv+BN+ReLU), and a final convolutional layer. Each block contains Conv(3×3)-BatchNorm-ReLU operations with 64 channels and padding to maintain spatial dimensions. In our implementation, we utilize the first 5 blocks of this architecture to extract a high-dimensional latent representation, which produces $\mathbf{F}_{latent} \in \mathbb{R}^{H\times W\times C_F}$, where H and W match the input dimensions and $C_F = 64$ represents the channel dimension.

While this shared extraction design aligns with established methods [19], [20], [25], the resulting $\mathbf{F}_{latent}$ only captures attribution signals at a coarse model level. The inflexibility of using fixed network parameters during inference may fail to distinguish subtle instance-specific variations critical for precise hyperparameter estimation. To address this limitation, we introduce dynamic network personalization where FEN parameters are adaptively conditioned on each input image, which is implemented through an Instance-Aware (IA) Hypernetwork $\mathcal{H}_{IA}(\cdot)$ that transforms the latent fingerprint into an instance-specific encoding vector:

$$z_{it} = \mathcal{H}_{IA}(\mathbf{F}_{latent}) \tag{3}$$

where the compact encoding $z_{it} \in \mathbb{R}^{1\times C_{IA}}$ efficiently captures the instance-specific characteristics that modulate the downstream fingerprint estimation. The Instance-Aware (IA) Hypernetwork $\mathcal{H}_{IA}(\cdot)$ contains three convolutional layers followed by three fully-connected layers (see Experiment Section for implementation details).

Model-Agnostic (MA) Encoder: To prevent model-level characteristics from dominating the latent fingerprints and better exploit the flexibility of the weight parameter space, we develop a complementary Model-Agnostic (MA) encoder that isolates hyperparameter-specific signals through hierarchical latent subspaces. This encoder initializes a structured set of learnable encodings hierarchically:

$$\mathbb{Z}_{MA} = \begin{cases} \mathcal{H}_0: & z_b \in \mathbb{R}^{1\times C_{MA}} \\ \mathcal{H}_1: & \{z_{hp}^i\}_{i=1}^{N_{hp}} \in \mathbb{R}^{N_{hp}\times C_{MA}} \end{cases} \tag{4}$$

where $\mathcal{H}_0$ represents the root level of our hierarchy containing the base code $z_b \in \mathbb{R}^{1\times C_{MA}}$ invariant to both model type and hyperparameter configurations, which captures fundamental generative characteristics. $\mathcal{H}_1$ represents the specialized level containing hyperparameter-specific codes $z_{hp}^i \in \mathbb{R}^{1\times C_{MA}}$, where each $z_{hp}^i \in \mathbb{R}^{1\times C_{MA}}$ is associated with a specific target hyperparameter (e.g., network architecture type, loss function), so that it focuses exclusively on discriminative features for that particular hyperparameter while remaining model-agnostic. We set $C_{MA} = C_{IA} = 256$ to maintain consistency in representation dimensions across both encoders. All latent encodings are initialized using Xavier initialization [93] with a gain factor of 1.0 to promote proper gradient flow during training.

During training, the set of learnable $\mathbb{Z}_{MA}$ in the MA encoder is iteratively updated through backpropagation to discover optimal representations for different hyperparameter configurations. When used for inference, these learned hierarchical encodings serve as a structured prior knowledge base that isolates hyperparameter-specific signals from model-type variations. Through the complementary nature of the MA encoder and the IA encoder, AdaParse is aimed at capturing both instance-specific artifacts and generalizable hyperparameter patterns across different generative processes.

C. Broadcasted Fusion

To generate the personalized fingerprint estimation networks for each input instance, as illustrated in Fig. 3, we accordingly design a Broadcasted Fusion module which firstly concatenates the learnable instance-aware, model-agnostic and hyperparameter-specific codes from both encoders:

$$\mathbf{Z} = \left[z_{it}, z_b, z_{hp}^i\right]_{i=1}^{N_{hp}}, \tag{5}$$

where $[\cdot]$ denotes the concatenation operation which results in $\mathbf{Z}^i \in \mathbb{R}^{1\times(C_{IA}+2C_{MA})}$. For the ith type of target hyperparameter, we assume its dedicated FEN requires weight parameters $\mathbf{W}^i \in \mathbb{R}^{C_{in}\times C_{out}}$, where C_{in} and C_{out} represent the fan-in and fan-out dimensions. The elements in the broadcasted code $\mathbf{Z}$ are transformed into these weight parameters through a shared Weight Generation Hypernetwork $\mathcal{H}_{WG}(\cdot)$:

$$\mathbf{W}^i = \mathcal{H}_{WG}(\mathbf{Z}^i) = \mathrm{Sigmoid}(\mathbf{U}^i(\mathrm{ReLU}(\mathbf{V}^i\mathbf{Z}^i))) \tag{6}$$

where $\mathbf{V}^i \in \mathbb{R}^{C_{hidden}\times(C_{IA}+2C_{MA})}$ and $\mathbf{U}^i \in \mathbb{R}^{C_{out}\times C_{hidden}}$ are learnable projection matrices with $C_{hidden} = 2048$ as a bottleneck dimension. This design enhances representational capacity by capturing complex non-linear mappings, which is crucial for modeling the intricate dependencies in hyperparameter-specific fingerprint generation, while maintaining computational efficiency through a compact bottleneck structure.

For each hyperparameter-specific FEN $\mathcal{E}_{\cdot,\mathbf{W}^i}(\cdot)$, we implement a focused personalization approach in a standard DnCNN network. Rather than replacing entire convolutional blocks, we strategically inject personalized linear transformation between selected feature layers. Specifically, we apply the generated weights $\mathbf{W}^i$ as a personalized linear transformation between the outputs of intermediate blocks. In this way, an adaptive information pathway is created to maintain architectural stability while enabling effective hyperparameter-specific fingerprint extraction.

D. Hyperparameter-specific Fingerprint Parsing

Given the ith FEN network parameterized by $\mathbf{W}^i$ each associated with the ith type of target hyperparameter $\mathbf{y}^i$, AdaParse estimates the hyperparameter-specific fingerprint via:

$$\mathbf{F}_{hyp}^i = \mathcal{E}_{\cdot,\mathbf{W}^i}\big(f(\mathbf{X})\big), \tag{7}$$

where $\mathbf{F}_{hyp}^i \in \mathbb{R}^{H\times W\times 3}$ represents the extracted fingerprint for the ith hyperparameter. $f(\cdot)$ is a convolutional feature extractor shared by all input images. The set of hyperparameter-specific fingerprints is consumed by a parsing network to finally predict the hyperparameters of the generative model:

$$\{\mathbf{y}^i\}_{i=1}^{N_{hp}} = \mathcal{D}_{parse}\big(\{\mathbf{F}_{hyp}^i\}_{i=1}^{N_{hp}}\big) \tag{8}$$

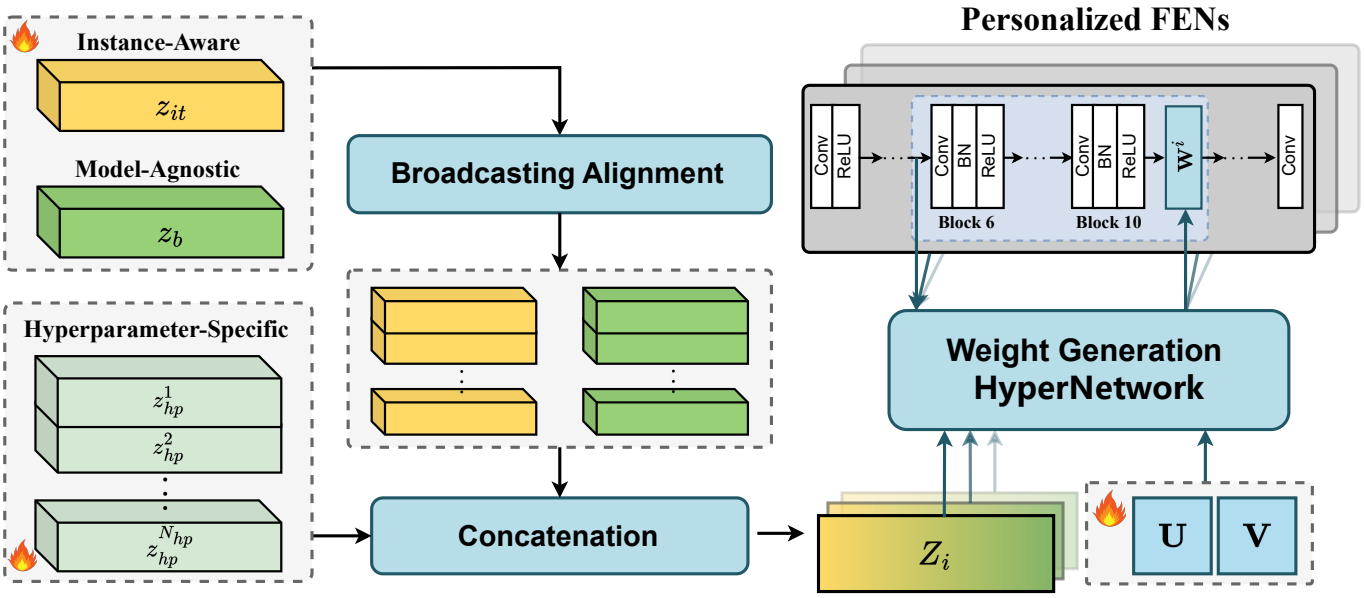


Fig. 3: **Illustration of the Broadcasted Fusion module in AdaParse**. The module combines instance-aware (z_{it}), model-agnostic (z_b), and hyperparameter-specific (z_{hp}) codes through broadcasting and concatenation to form Z_i, which is used by the Weight Generation HyperNetwork to generate personalized parameters via factorized weight generation ($\mathbf{U}, \mathbf{V}$). These parameters define an adaptive transformation layer that processes features from block 10 before they enter block 11, with the network receiving feature inputs from block 5 for hyperparameter-specific adaptation. The 🔥 icon denotes the vectors or matrices that are learned during training.

where $\mathcal{D}_{parse}$ is similar to the parsing network of RE-Net [19]. It processes hyperparameter fingerprints by separating their features before the final linear layers, unlike model fingerprints in RE-Net which share identical inputs to these layers.

During training, AdaParse jointly optimizes the fingerprint constraint losses $\mathcal{L}_{cons}$ upon $\{\mathbf{F}_{hyp}^i\}_{i=1}^{N_{hp}}$ and hyperparameter prediction loss $\mathcal{L}_{pred}$ upon $\{\mathbf{y}^i\}_{i=1}^{N_{hp}}$:

$$\mathcal{L}_{total} = \mathcal{L}_{cons} + \lambda \mathcal{L}_{pred} \tag{9}$$

where λ is a pre-set loss weight. Following [19], $\mathcal{L}_{cons}$ consists of four fingerprint constraint losses: magnitude loss to constrain fingerprint amplitude; spectrum loss to minimize low-frequency content; repetitive loss to maximize high-frequency information for encouraging spatial repetitive patterns; and energy loss to balance the Fourier spectrum energy in vertical and horizontal directions.

For the hyperparameter prediction loss $\mathcal{L}_{pred}$, we employ a hierarchical approach that leverages similarities between different generative models. We cluster models based on their hyperparameters and perform both coarse-level (cluster) and fine-level (instance) predictions. For the network architecture prediction, we use:

$$\mathcal{L}_{pred}^{arch} = \mathcal{L}_c + \mathcal{L}_d \tag{10}$$

where $\mathcal{L}_c$ is an L2 loss for continuous parameters (e.g., number of layers, filter size) and $\mathcal{L}_d$ is a weighted cross-entropy loss for discrete parameters (e.g., normalization type, activation function).

For loss function type prediction, we use weighted cross-entropy loss to handle class imbalance:

$$\mathcal{L}_{pred}^{loss} = -\sum_{m=1}^{N^l} \sum_i w_m^i \cdot y_m^i \cdot \log(S(\hat{y}_m^i)) \tag{11}$$

where w_m^i is the weight for class i of the m-th loss type, y_m^i is the ground-truth, $\hat{y}_m^i$ is the prediction, and S is the softmax function.

IV. Experiment

In this section, we conduct extensive experiments to validate our proposed AdaParse and analyze the corresponding results. We also show the ablation studies to validate our design choices and selection of hyper-parameters. We have further validated the generalization ability of AdaParse through cross-model evaluation.

A. Dataset and Configurations

Dataset: We experimented on the large-scale dataset released by [19] for the Reverse Engineering of common visual generative models, denoted as RE-116, which comprises 116,000 images generated from 116 publicly available generative models. Each model category contains 1,000 images it generated. Each image instance is annotated with the hyperparameters of its corresponding generative model, i.e., 15 hyperparameters on configurations of network architectures and 10 hyperparameters on usages of loss functions. We accordingly set N_{hp} as 25. In response to emerging technological paradigms, RE-116 is augmented with 7,000 images generated by seven diffusion models with 1,000 images each, denoted as

RE-123 benchmark [19]. We further extend RE-123 by adding 1,000 images generated by an additional diffusion model, resulting in RE-124.

Evaluation Metrics: We followed [19] to report L1 errors for continuous hyperparameters and F1 scores for discrete ones. The F1 score calculates the harmonic mean of precision and recall for each classification label to obtain the F1 score for that label, and then takes the average of all labels to obtain the macro F1 score.

Experimental Settings: For the RE-116 benchmark, we followed [19] to use 12 generative models for testing, while the remaining 104 models were used for training. For the RE-124 benchmark, we followed a similar protocol where 4, including the newly added one, out of 8 diffusion models were held out for testing, with the remaining 4 diffusion models included in the training set alongside the aforementioned 116 generative models. To conduct robust evaluation, four different random splits of training/testing sets were created while maintaining the abovementioned training v.s. testing ratio. The final performance metrics were averaged across these four splitting configurations.

Implementation Details: The loss weight λ in Eqn.(9) is set as 10. w_m^i in Eqn.(11) is set as $\frac{N}{N_m^i}$ where N_m^i is the number of training examples for the *i*-th class of *m*-th loss type, and N is the number of total training examples. For Instance-Aware (IA) Hypernetwork $\mathcal{H}_{IA}(\cdot)$, the network architecture consists of three convolutional layers (64→128→256→512, all with 3×3 kernels and stride 1) followed by three fully connected layers (input dimensions 4096→1024→256). During training, we used the Adam optimizer with the learning rate of 0.0001 and the weight decay of 0.0001. Our AdaParse is trained with a batch size of 16 for 20 epochs. All the experiments were conducted using a single NVIDIA GeForce RTX 3090 card.

B. Quantitative Results and Analysis

Following official configuration [19], we firstly experimented on loss function prediction and network architecture prediction using the RE-116 benchmark. Since the prediction of the discrete hyperparameters resembles the conventional image classification task, we mainly compare with three groups of baseline methods in such scenarios: (1) The architectures of the first group are mainly based on widely-used classic image recognition networks, including VGGNet [94], ResNet-18 [95], Inception v3 [96], Inception v4 [97] and Xception [98]. The classification heads in these methods were modified to adapt to our conducted task. (2) The architectures of the second group are based on popular deepfake detectors [51], [61], [98] where we also adapted their binary classifiers as in (1). (3) The third group mainly contains the state-of-the-art RE-Net [19] which estimates coarse-level model fingerprints.

The results of group (3) across all experiments are collected from our reproduced training and evaluation following the officially-released codes[2]. To determine whether the model is truly learning meaningful relationships rather than just statistical patterns in the data, we followed [19] to construct a Random GroundTruth (Random G.T.). Specifically, to break the correct associations, we shuffled the actual hyperparameter values across different generative models. Therefore, a significant performance gap between normal results and random groundtruth results indicates that the model is actually learning to associate images with their true properties, rather than just recognizing patterns that are learned from a given data distribution.

Loss Function Prediction: TABLE II demonstrates the quantitative results on prediction of loss function types in generative models, evaluated by F1 scores. Overall, our method achieves the highest F1 score averaged on all target loss functions, which outperforms the second-best approach by an obvious margin of 4.6%. AdaParse also performs the best on nearly half of them in terms of individual F1 scores. This verifies the effectiveness of our proposed method through exploiting fine-grained dynamic hyperparameter fingerprints. The second-best approach is the RE-Net [19] method which conducts coarse-level model fingerprint estimation. Moreover, we can see that despite the specially-designed network architectures in identifying AI-generated visual content, the deepfake detectors [51], [61], [98] are inadequate of capturing the fine-grained traces for inferring model hyperparameters, which also underperform the general image classifier backbones. We also observe that while the adapted version of a vanilla image classification network, i.e., ResNet-18 [95], achieves the highest F1 score among the image classifier backbones, the group it belongs to performs consistently inferior to our AdaParse as well as the RE-Net baseline in terms of average performance. However, we note that for certain loss functions, AdaParse performs worse than some other methods. For example, ResNet-18 achieves highest F1 scores on L2. This can be attributed to the strategy that other methods essentially rely on a single coarse-grained representation inherently closer in the feature space to specific hyperparameters, thus excelling in those specific cases at the expense of generality. In contrast, our fine-grained approach reduces the coupling between different classifications by generating hyperparameter-specific fingerprints, leading to better average performance.

Taking a closer look at the prediction performances, the substantial gap between our full model (0.662) and its Random G.T. version (0.555) demonstrates that our approach genuinely learns meaningful relationships rather than statistical patterns. The larger performance drop in our method (0.107 decrease) versus RE-Net (0.092 decrease) suggests our approach is more effectively capturing such relationships between images and the loss functions in their corresponding generative architectures.

Network Architecture Prediction: For the prediction on discrete hyperparameters of network architecture configurations in visual generative models, we follow TABLE II to compare our proposed method with three groups of baseline methods, as shown in TABLE III. The results are evaluated by F1 scores as in TABLE II. It can be seen that similar to TABLE II, our method achieves the best average performance compared with the three groups of baseline methods, outperforming the second-best method by 1.7% on F1 score. AdaParse also attains superior performance in half of the six discrete architectural hyperparameters. Counterintuitively, we

[2]https://github.com/vishal3477/Reverse_Engineering_GMs

TABLE II: Quantitative results on the reverse engineering of loss function configurations in visual generative models, experimented on the RE-116 benchmark. The results were evaluated by F1 scores.

Method	Loss Function Types										Average ↑
	L1	L2	MSE	MMD	LS	WGAN	KL	Adv.	Hinge	CE	
CNN Backbone Based Methods											
Xception [98]	0.581	0.508	0.573	0.466	0.627	0.508	0.756	0.469	0.540	0.741	0.577
VGG19 [94]	0.564	0.546	0.600	0.465	0.640	0.534	0.805	0.479	0.553	0.743	0.593
ResNet18 [95]	0.580	**0.616**	0.607	0.484	0.648	0.478	0.818	0.479	0.568	0.765	0.604
Inception v3 [96]	0.564	0.558	0.522	0.477	0.620	0.504	**0.826**	0.465	0.529	0.729	0.580
Inception v4 [97]	0.557	0.554	0.544	0.476	0.614	0.527	0.819	0.476	0.530	**0.781**	0.588
DeepFake Detection Specialized Methods											
CNNDetection [69]	0.564	0.547	0.554	0.471	0.609	0.523	0.771	0.471	0.525	0.760	0.580
MAT [61]	0.571	0.537	0.569	0.471	0.612	0.508	0.773	0.469	0.558	0.754	0.582
NPR [51]	0.533	0.506	0.619	0.484	**0.671**	0.481	0.795	0.459	0.615	0.761	0.592
Reverse Engineering Methods											
RE-Net (Random G.T.) [19]	0.455	0.497	0.502	0.789	0.447	0.451	0.472	0.564	0.552	0.510	0.524
RE-Net [19]	**0.604**	0.540	0.554	0.619	0.647	0.526	0.818	0.604	0.478	0.766	0.616
Ours (Random G.T.)	0.397	0.459	0.462	**0.958**	0.488	0.606	0.489	0.605	0.597	0.487	0.555
Ours (Full)	0.592	0.480	**0.749**	0.738	0.614	**0.620**	0.813	**0.611**	**0.678**	0.725	**0.662**

* **Loss functions**: L1 = Absolute Error Loss, L2 = Squared Error Loss, MSE = Mean Squared Error, MMD = Maximum Mean Discrepancy, LS = Least Squares, WGAN = Wasserstein GAN Loss, KL = Kullback-Leibler Divergence, Adv. = Adversarial Loss, CE = Cross Entropy
* **Method variants**: G.T. = Ground Truth
* Best results per column are in **bold**. Best average result is **highlighted**.

observe that the groups of deepfake detectors and RE-Net both perform inferior to several image classification based backbones, i.e., Xception, VGG-19 and ResNet-18. We infer that the prediction on basic building blocks of network layers can be partially addressed by the rich semantic clues extracted by the basic CNN backbone based methods. In contrast, our proposed method can better cope with the subtle traces to layers in generative models via personalizing hyperparameter fingerprint estimation network for each input image.

On the other hand, the larger performance gap between our full model and its Random G.T. variant (0.263 decrease) versus the gap in RE-Net (0.259 decrease) demonstrates that our approach better captures true architectural relationships. Our AdaParse also shows stronger baseline performances even with randomized data.

Despite the overall strong performance, a notable case emerges from the scores. While our method achieves a high F1 score of 0.716 for Last-Ly Nonlin., the score for Block Nonlin. remains significantly lower at 0.398. Despite both involving nonlinearity classification, this difference suggests that the position of a component within the architecture critically affects fingerprint extraction difficulty. The activation patterns introduced by the last-layer nonlinearity have a more direct and less transformed impact on the generated image. In contrast, the effect of a nonlinearity within blocks is heavily modulated and potentially obscured by other layers, making its unique fingerprint far more difficult to isolate.

As for the prediction on continuous hyperparameters of network architectures of generative models, the comparisons with the RE-Net baseline and the corresponding results are summarized in TABLE IV. We can see that AdaParse further achieves the lowest L1 error compared with RE-Net [19] along with its variants. Moreover, the performance gap between our method and its Random G.T. variant is slightly larger than that in RE-Net, which keeps consistent with the precedences of the learned genuineness in both TABLE II and TABLE III.

Comparison with an Advanced Method: To provide a more comprehensive comparison, we conducted a comparative experiment between our AdaParse and LGPN [44] on the RE-116 benchmark. In this experiment, we created a new test split for a feasible evaluation of LGPN. As shown in TABLE V, AdaParse achieves F1 scores of 0.443 for network architecture prediction and 0.554 for loss function prediction, outperforming LGPN, which scores 0.418 and 0.526, respectively. The superiority can be attributed to the adaptive fingerprinting framework of AdaParse, which dynamically personalizes feature extraction for each input image. This approach effectively mitigates bias from class imbalance, whereas LGPN relies on a static graph-based model that may overfit to major classes. This results underscores the efficacy of adaptive techniques, highlighting the capability of AdaParse in handling class imbalance.

Generalization to Diffusion-based Generative Models: To keep pace with the development trend and further demonstrate the generalization ability of our proposed method, we have evaluated our method on the benchmark comprised of diffusion models, including ADM [8], ADM-G [8], DDPM [99], DDIM [100], vanilla LDM [9], Stable-Diffusion [101], GLIDE [102] and SDXL [103]. Specifically, the dataset includes 8 different diffusion models. Each diffusion model generates 1,000 images. For comprehensive evaluation, we create 4 distinct test splits, each containing images from 4 diffusion models. The remaining diffusion models, together with the complete dataset, are used for training.

TABLE III: Quantitative results on the reverse engineering of discrete hyperparameters of network architecture configurations in visual generative models, experimented on the RE-116 benchmark. The results were evaluated by F1 scores.

Method	**Architectural Components**						**Average ↑**
	Norm.	Last-Ly Nonlin.	Block Nonlin.	Upsamp.	Skip Conn.	Downsamp.	
CNN Backbone Based Methods							
Xception [98]	0.614	0.574	0.364	0.680	0.747	0.780	0.627
VGG19 [94]	**0.662**	0.576	0.373	0.638	0.746	**0.791**	0.631
ResNet18 [95]	0.630	0.589	0.382	0.693	0.738	0.746	0.630
Inception v3 [96]	0.625	0.520	0.347	0.687	0.709	0.760	0.608
Inception v4 [97]	0.629	0.568	0.381	0.650	0.743	0.763	0.622
DeepFake Detection Specialized Methods							
CNNDetection [69]	0.612	0.527	0.353	0.667	**0.767**	0.777	0.617
MAT [61]	0.592	0.577	0.352	0.657	0.746	0.774	0.616
NPR [51]	0.599	0.576	0.392	0.645	0.688	0.788	0.615
Reverse Engineering Methods							
RE-Net (Random G.T.) [19]	0.211	0.234	0.214	0.466	0.554	0.502	0.363
RE-Net [19]	0.614	0.555	0.373	0.671	0.739	0.781	0.622
Ours (Random G.T.)	0.208	0.236	0.275	0.532	0.555	0.507	0.385
Ours (Full)	0.620	**0.716**	**0.398**	**0.693**	0.754	0.708	**0.648**

* **Abbreviations**: Norm. = Normalization type, Last-Ly Nonlin. = Nonlinearity type in last layer, Block Nonlin. = Nonlinearity type in blocks, Upsamp. = Upsampling type, Skip Conn. = Skip connection, Downsamp. = Downsampling method
* **Method variants**: G.T. = Ground Truth
* Best results per column are in **bold**. Best average result is **highlighted**.

TABLE IV: Quantitative results on the reverse engineering of continuous hyperparameters of network architecture configurations in visual generative models, experimented on the RE-116 benchmark. The results were evaluated by L1 error.

Method	**Continuous Hyperparameters**									**Average ↓**
	Lys	Conv. lys	FC lys	Pool. lys	Norm. lys	Filters	Blocks	Lys/block	Params.	
RE-Net (Random G.T.) [19]	0.249	0.230	0.105	0.041	0.144	0.277	0.149	0.183	0.091	0.163
RE-Net [19]	0.187	0.187	0.095	0.040	0.103	0.184	0.115	0.152	0.075	0.126
Ours (Random G.T.)	0.250	0.225	0.097	0.041	0.132	0.288	0.152	0.177	0.087	0.161
Ours (Full)	0.171	0.161	0.101	0.051	0.107	0.177	0.119	0.142	0.079	**0.123**

* **Hyperparameters**: Lys = Total layers, Conv. lys = Convolutional layers, FC lys = Fully connected layers, Pool. lys = Pooling layers, Norm. lys = Normalization layers, Filters = Number of filters, Lys/block = Layers per block, Params. = Number of parameters
* **Method variants**: G.T. = Ground Truth
* Lower is better for L1 error. Best average result is **highlighted**.

TABLE V: Quantitative results compared with LGPN [44] on the reverse engineering of discrete hyperparameters configurations in visual generative models, experimented on the RE-116 benchmark. The results were evaluated by F1 scores.

Method	**Network Architecture**	**Loss Function**
LGPN [44]	0.418	0.526
Ours	**0.443**	**0.554**

TABLE VI: Quantitative results on the reverse engineering on diffusion models, experimented on the RE-124 benchmark.

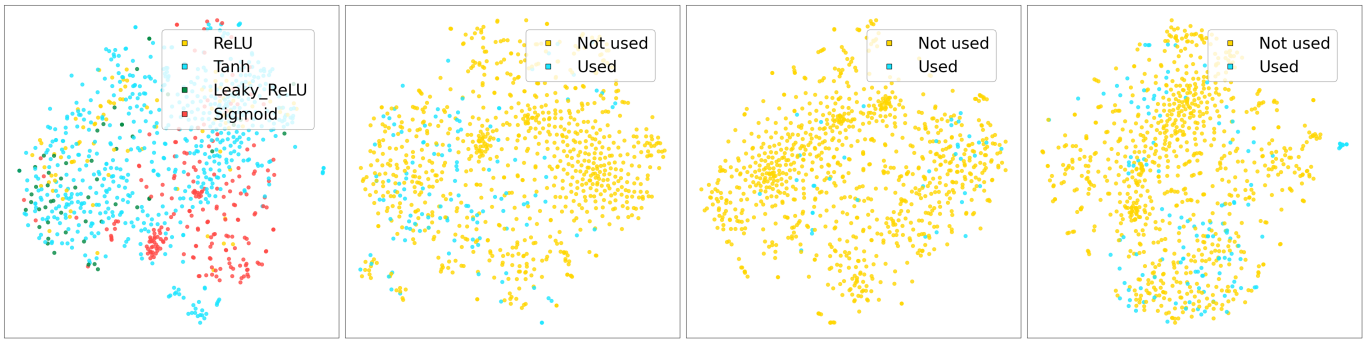
Method	**Network Architecture Parameters**		**Loss Function**
	Continuous (L1↓)	**Discrete** (F1↑)	
RE-Net [19]	0.226	0.372	0.472
Ours	**0.221**	**0.377**	**0.481**

TABLE VI presents the performance comparison between our method and the RE-Net [19] baseline. Our approach demonstrates consistent superiority across all evaluation metrics. In loss function type prediction, our method achieves a 1.9% relative improvement in F1 score over RE-Net. For network architecture parameters, we observe meaningful gains in both continuous parameter regression (reduced L1 error) and discrete parameter classification (improved F1 score). These improvements are particularly valuable given the complex architectural diversity among modern diffusion models with varying attention mechanisms and normalization techniques. The effectiveness on this benchmark suggests the potential of our method for generalizing to emerging visual generative models.

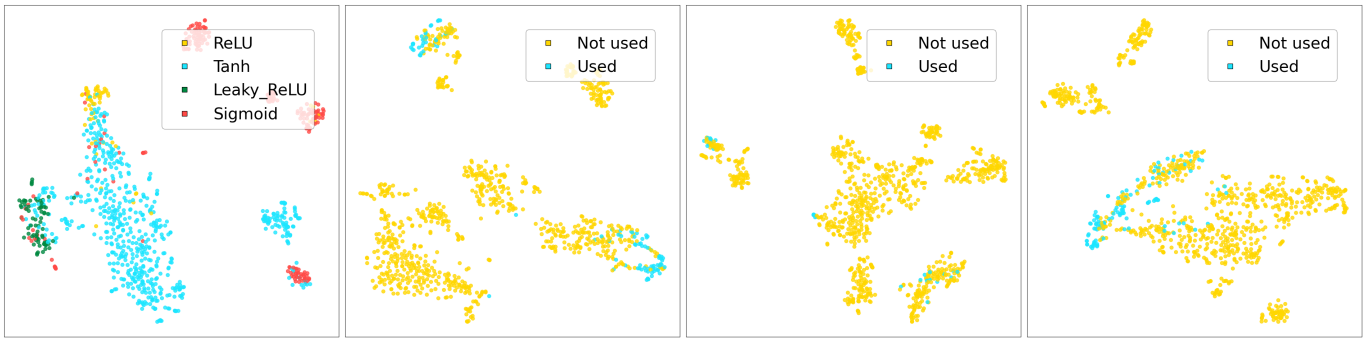
TABLE VII: Inference runtime comparison (per sample).

Method	**Inference Time (ms)**	**Relative Time**
RE-Net [19]	0.492	1×
Ours	0.522	1.06×

Inference Runtime: To evaluate the computational efficiency of our framework, we conducted an inference runtime experiment under identical experimental settings. As shown in TABLE VII, our AdaParse requires 0.522 ms per sample, which is 6% slower than the RE-Net [19] (0.492 ms). This marginal increase in runtime is a reasonable trade-off given the gains in hyperparameter prediction accuracy. The lightweight



(a) Non-linearity of last layer (b) MSE loss (c) MMD loss (d) WGAN loss



(e) Non-linearity of last layer (f) MSE loss (g) MMD loss (h) WGAN loss

Fig. 4: T-SNE visualization of discrete hyperparameter prediction features. Top row (a-d): Results from RE-Net baseline showing less distinct clustering patterns. Bottom row (e-h): Results from our method showing clearer separation between different hyperparameter configurations. All visualizations are based on the RE-116 benchmark.

design of our Broadcasted Fusion module and personalized FENs ensures that the computational overhead remains minimal, maintaining practical applicability for real-world reverse engineering tasks.

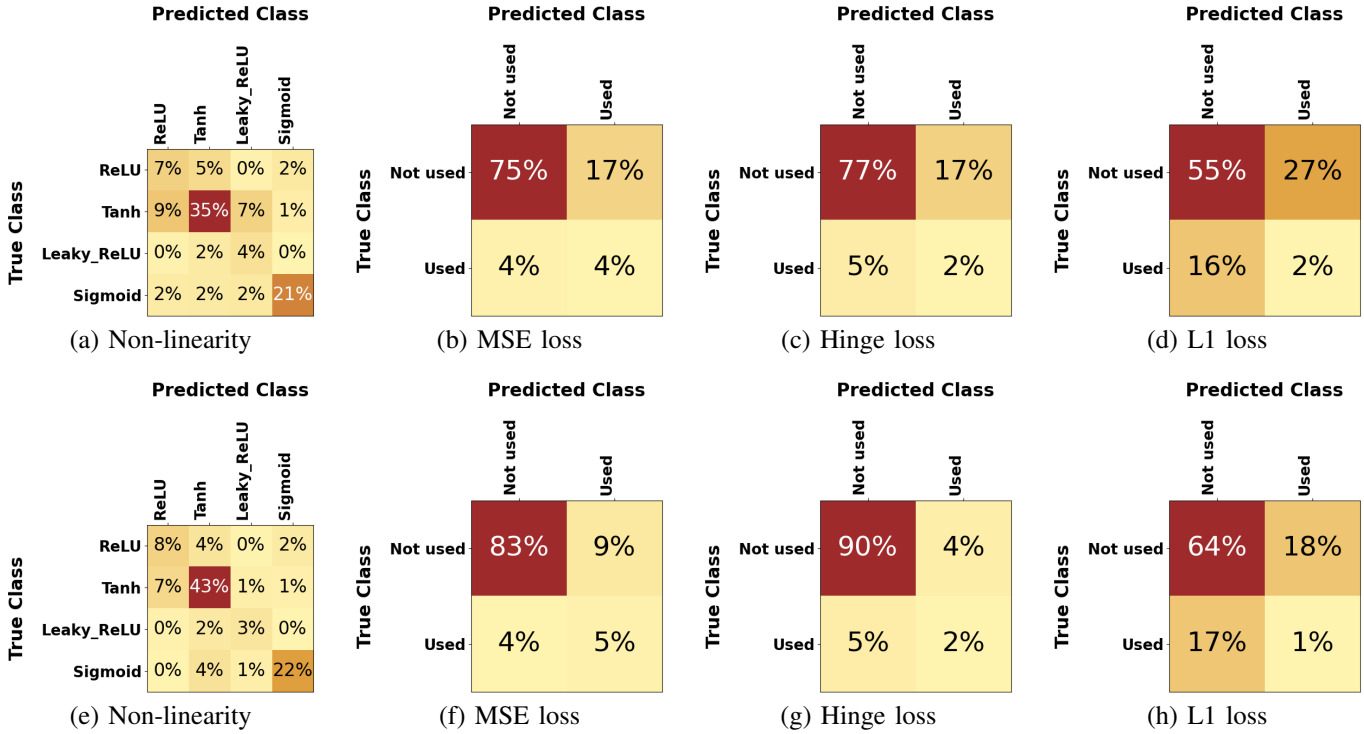
C. Visualization Results and Analysis

To demonstrate the superiority of our proposed AdaParse over the baseline method [19] more intuitively, we conducted the t-SNE [104] visualization as shown in Fig. 4, where we display how well the compared methods can differentiate different hyperparameter configurations. Our AdaParse (the bottom row) can significantly improve the clustering performance compared to the RE-Net baseline (the top row). Such improvements can be consistently observed in the configuration predictions of both network architecture (e.g., type of non-linearity in the last layer) and usage of loss functions (e.g., MSE loss, MMD loss and WGAN [105] loss). Specifically, in the predictions of type of non-linear layer (subfigures a and e) in the network architectures, our method achieves clear separation between different activation functions (ReLU, Tanh, Leaky ReLU, and Sigmoid), while the baseline shows significant overlap. Similarly, for the determination of employed loss functions (subfigures b-d vs. f-h), our approach produces more distinct clusters between "Used" and "Not used" categories, with particularly notable improvements in the WGAN loss visualization where our method creates well-defined separated clusters compared to the more scattered distribution of the baseline. These visualizations qualitatively demonstrate the enhanced feature representation learning capability for hyperparameter prediction of our hyperparameter-specific fingerprints that are personalized for each input image.

Fig. 5 presents the confusion matrices for key discrete hyperparameter predictions, where we compare the RE-Net baseline (top row) with our method (bottom row). Each matrix displays the predicted classes (columns) against ground-truth classes (rows), with diagonal elements representing correct classifications. For the predictions of types of non-linear layers (a,e), our approach achieves higher accuracy in identifying activation functions, particularly for Tanh (43% vs. 35%). In the predictions of employed loss functions, our method consistently outperforms the baseline with higher true classification rates across the shown loss functions (MSE, Hinge, and L1). Notably, our approach achieves substantial improvements in true negative rates, with increases of 8-13 %, which suggests the enhanced ability to correctly identify when specific loss functions were not employed in the generative process. This improved discrimination capability is particularly valuable for forensic analysis where ruling out certain architectural choices is often as important as confirming others.

D. Ablation Studies and Analysis

Key Components in AdaParse: We firstly conducted ablation studies to validate the effectiveness of key components in AdaParse, including the instance-aware encoder, model-agnostic encoder and broadcasted fusion module. This is achieved by ablating the combination of instance-aware code z_{it}, model-agnostic code z_b and hyperparameter-specific codes $\{z_{hp}^1, z_{hp}^2, \cdots, z_{hp}^{N_{hp}}\}$ each associated with one type of tatget hyperparameter. TABLE VIII presents the impact of systematically removing different encoding components from our framework. The instance-aware encoding (z_{it}) provides a baseline performance. When combined with hyperparameter-



(a) Non-linearity (b) MSE loss (c) Hinge loss (d) L1 loss

(e) Non-linearity (f) MSE loss (g) Hinge loss (h) L1 loss

Fig. 5: Confusion matrices of discrete hyper-parameters prediction produced by RE-Net (upper row, a-d) and our method (bottom row, e-h). The corresponding hyper-parameters: (a,e) Type of non-linearity in the last layer; (b,f) Usage of MSE loss; (c,g) Usage of Hinge loss; (d,h) Usage of L1 loss. The first three columns show results on the RE-116 benchmark, and the last column shows results on the RE-123 benchmark.

TABLE VIII: Results of ablation studies on the effectiveness of key components (instance-aware code, model-agnostic code and the hyperparameter-specific codes in Broadcasted Fusion module) in AdaParse. The results were evaluated by F1 scores.

Components			Arch.	Loss
Instance-aware	Hyperparam.-specific	Model-agnostic		
✓			0.614	0.629
✓	✓		0.630	0.593
✓	✓	✓	**0.648**	**0.662**
✓		✓	0.623	0.613
	✓	✓	0.629	0.583

specific codes (z_{hp}), the architecture prediction improves to 0.630 (+1.6%) while the loss function prediction decreases slightly to 0.593. Pairing instance-aware with model-agnostic encoding (z_b) shows modest gains in architecture prediction and maintains similar loss function performance. The integration of all three components achieves the highest scores and outperforms all partial configurations. These results collectively confirm that each component contributes to our reverse engineering task, with their combination capturing both sample-specific features and broader contextual information necessary for accurate hyperparameter prediction.

How FENs are Personalized: We then investigate the optimal position of our personalization strategy within the DnCNN [92] architecture. By default, our approach takes the output from block 5 as input to our Weight Generation HyperNetwork, which then generates parameters for a personalized linear transformation layer that processes features from block 10 before entering block 11. This creates a feature pathway that focuses on hyperparameter-specific information. TABLE IX(a) shows the impact of adjusting the feature extraction position by varying which block's output is fed to the HyperNetwork while maintaining the same transformation position. When shifting the starting position earlier (blocks 4-10), we observe a decrease in F1 scores for both discrete architecture prediction and loss function. Similarly, delaying the starting point to later layers (blocks 8-10 or 10-10) degrades performance, with the most significant drop occurring at the extreme case (10-10). In TABLE IX(b), we further investigate the effect of extending or shortening the customization range by adjusting the endpoint while keeping the starting position fixed at block 6. Similarly, we observe that extending personalization to later layers (blocks 6-12) leads to moderate degradation, while reducing the range (blocks 6-8 or 6-6) causes more significant performance drops, particularly when customization is limited to a single block (6-6). These results collectively indicate that the intermediate blocks (6-10) of FEN personalization by our Weight Generation HyperNetwork represent an optimal position that captures features at an appropriate level of abstraction. This strategic positioning enables effective parameter adaptation while maintaining the overall representational capacity for hyperparameter personalization.

Strategy of Broadcasted Fusion: Apart from the concatenation operation in Eqn. 5, we have further experimented on alternative strategies including addition and attentional weighting. The corresponding quantitative results are summarized in TABLE X(a). It can be seen that the default strategy achieving the highest F1 scores for both network architecture and loss function predictions. Simple addition operation leads

TABLE IX: Results of ablation studies on the position of personalization pathway in FEN networks.

Blocks Range	Network Architecture	Loss Function
(a) Varying feature extraction position		
4-10	0.619	0.627
6-10 (Ours)	**0.648**	**0.662**
8-10	0.642	0.617
10-10	0.597	0.624
(b) Varying ending connection position		
6-12	0.635	0.609
6-10 (Ours)	**0.648**	**0.662**
6-8	0.640	0.596
6-6	0.597	0.608

TABLE X: Results of ablation studies on different design choices in AdaParse.

Design Choice	Network Architecture	Loss Function
(a) Fusion Strategy		
Addition	0.606	0.596
Attention	0.634	0.621
Concatenation (Ours)	**0.648**	**0.662**
(b) Code Dimensionality		
64	0.621	0.588
128	0.623	0.580
256 (Ours)	0.648	**0.662**
512	**0.650**	0.607

to the most performance degradation as it compresses the representation space and blends the distinct characteristics of different types of learned codes. While attention mechanisms partially mitigate this issue through learned weighting, they still underperform our concatenation-based strategy by 1.4% and 4.1% respectively. These results empirically suggest that preserving the independence of instance-aware, model-agnostic, and hyperparameter-specific encodings is essential for effective fingerprint generation.

Dimensionality of Codes: By default we set the dimension of the learned codes (z_{it}, z_b, z_{hp}^i) as 256. As displayed in TABLE X(b), we have also experimented on other choices including 64, 128 and 512. The dimensionality of 256 achieves the best overall results across two predicted hyperparameters, particularly for loss function prediction. While increasing to 512 dimensions marginally improves network architecture prediction (+0.002), it degrades loss function performance and substantially increases computational overhead.

E. Robustness and Analysis

To compare the robustness of AdaParse and RE-Net [19], we conducted experiments without retraining the models under traditional image post-processing operations and advanced attacks, including geometric transformations (rotation and flip), JPEG compression, adversarial attack (FGSM [106]-based perturbations), and reconstruction attack (DnCNN [92]-based reconstruction). These manipulations simulate real-world scenarios where AI-generated images may be manipulated after creation. As shown in TABLE XI, we report the performance of AdaParse on the RE-116 benchmark. The results demonstrate that AdaParse maintains stronger robustness across most of them. In average, AdaParse shows smaller performance degradations than RE-Net [19]. This superiority is particularly evident in discrete architecture prediction under the former three manipulations, where the drop in F1 score for AdaParse is significantly lower, indicating its robustness. The overall stronger resilience of AdaParse suggests it learns more robust representations that are less susceptible to common image manipulations. For the reconstruction attack, both methods experience substantial performance degradation, with AdaParse showing slightly larger drops in discrete architecture hyperparameters and loss function types. This is because the DnCNN-based reconstruction process completely overwriting the original fingerprints in images, which makes both methods practically ineffective. The marginal differences in performance drop under such severe fingerprint destruction thus hold limited practical significance.

TABLE XI: Quantitative results of the robustness of the reverse engineering on visual generative models, experimented on the RE-116 benchmark. The numbers are change values of L1 error or F1 score after the manipulation of the row, while the left and right sides of '/' correspond to AdaParse and RE-Net [19], respectively. GeoT: Geometric Transformations. JPEGC: JPEG Compression. AA: Adversarial Attack. RA: Reconstruction Attack.

Manipulations	Network Architecture Parameters		Loss Function
	Continuous (L1↓)	**Discrete** (F1↑)	
GeoT	**+0.006**/+0.010	**-0.018**/-0.044	-0.010/-0.010
JPEGC	+0.005/**+0.004**	**-0.010**/-0.031	**-0.010**/-0.018
AA	**+0.009**/+0.013	**-0.032**/-0.075	**-0.018**/-0.041
RA	**+0.038**/+0.043	-0.344/**-0.313**	-0.178/**-0.153**
Average	**+0.015**/+0.018	**-0.101**/-0.116	**-0.054**/-0.056

TABLE XII: Results of deepfake detection and model attribution. The results were evaluated by accuracy.

Method	Detection	Attribution
RE-Net [19]	0.707	0.484
Ours	**0.788**	**0.490**

F. Applications to Other Forensic Tasks

We conducted experiments that bridge the technical contributions of AdaParse to other real-world forensic applications, specifically focusing on deepfake detection and model attribution. For deepfake detection, we evaluated our method on the Celeb-DF-v2 [107]. For model attribution, following the fake classes of [20] and incorporating two advanced models, the benchmark comprises six generative models from RE-124, namely CRAMERGAN [108], MMDGAN [109], ProGAN [110], SNGAN [111], GLIDE [102] and SDXL [103]. AdaParse and RE-Net [19] can be adopted by replacing the parsing network with a shallow network for binary or multi-class classification. As shown in TABLE XII, AdaParse consistently outperforms RE-Net in both tasks, with improved accuracy in detection and attribution, thereby demonstrating its efficacy in tangible use cases in other forensic scenarios.

G. Hyperparameter Correlations

To investigate the potential influence of design-level correlations between hyperparameters, we conducted an experiment focusing on a pair of architectural hyperparameters, namely

TABLE XIII: F1 scores of Norm. and Upsamp. before and after filtering test sets. The left and right sides of '/' correspond to AdaParse and RE-Net [19], respectively.

Filtered	Norm.	Upsamp.
Before	**0.620**/0.614	**0.693**/0.671
After	**0.608**/0.605	**0.684**/0.660

normalization type (Norm.) and upsampling type (Upsamp.). In RE-116 [19], there are a number of models where both Norm. and Upsamp. take on the value '0'. We have created a filtered version of the test sets by removing all models where both Norm. and Upsamp. are '0'. The results are summarized in TABLE XIII. The performance for both hyperparameters shows a slight decrease after filtering. This marginal decline indicates that, while the double-'0' configuration in the original dataset provides a correlated signal that slightly inflates the performance, our AdaParse does not heavily rely on this specific correlation for prediction. Meanwhile, AdaParse still maintains an advantage over RE-Net [19].

V. Conclusion

In this paper, we propose AdaParse, a hyperparameter-specialized adaptive fingerprinting framework for model reverse engineering from AI-generated images. Unlike previous approaches that rely on a single coarse model fingerprint, our method dynamically personalizes the hyperparameter estimation process by generating tailored fingerprint networks for each input image. Through our complementary two-branch encoder architecture that balances instance-aware and model-agnostic information, along with our Broadcasted Fusion mechanism, AdaParse effectively captures fine-grained hyperparameter-specific traces that reveal architectural choices and training objectives of generative models. Extensive experiments across 123 generative models demonstrate that our approach significantly outperforms previous state-of-the-art methods in reverse engineering critical hyperparameters. In future work, we plan to improve our framework along several directions to address its current limitations. Beyond incorporating data attribution predictions to detect the influence of training data and realizing the visualization of the latent fingerprint to enhance interpretability, we will focus on strengthening the practical applicability by expanding the dataset to encompass a much broader and more diverse library of advanced generative models, coupled with integrating more effective representation learning techniques.

VI. Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant 62125603, Grant 62336004, Grant 62321005, and Grant 62441616, and in part by the China Postdoctoral Science Foundation under Grant 2024M761674.

References

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[2] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *NeurIPS*, vol. 28, 2015.

[3] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *NeurIPS*, vol. 30, 2017.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NeurIPS*, vol. 27, 2014.

[5] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019, pp. 4401–4410.

[6] Z. Huang, S. Chen, J. Zhang, and H. Shan, "Pfa-gan: Progressive face aging with generative adversarial network," *TIFS*, vol. 16, pp. 2031–2045, 2020.

[7] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021.

[8] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *NeurIPS*, vol. 34, pp. 8780–8794, 2021.

[9] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022, pp. 10 684–10 695.

[10] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *WIFS*, 2018, pp. 1–7.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[12] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, "Leveraging frequency analysis for deep fake image recognition," in *ICML*, 2020, pp. 3247–3258.

[13] Z. Guo, Z. Jia, L. Wang, D. Wang, G. Yang, and N. Kasabov, "Constructing new backbone networks via space-frequency interactive convolution for deepfake detection," *TIFS*, vol. 19, pp. 401–413, 2023.

[14] R. Xia, D. Liu, J. Li, L. Yuan, N. Wang, and X. Gao, "Mmnet: multi-collaboration and multi-supervision network for sequential deepfake detection," *TIFS*, vol. 19, pp. 3409–3422, 2024.

[15] X. Zhou, H. Han, S. Shan, and X. Chen, "Fine-grained open-set deepfake detection via unsupervised domain adaptation," *TIFS*, vol. 19, pp. 7536–7547, 2024.

[16] Z. Sun, N. Ruan, and J. Li, "Ddl: Effective and comprehensible interpretation framework for diverse deepfake detectors," *TIFS*, vol. 20, pp. 3601–3615, 2025.

[17] G. Somepalli, A. Gupta, K. Gupta, S. Palta, M. Goldblum, J. Geiping, A. Shrivastava, and T. Goldstein, "Investigating style similarity in diffusion models," in *ECCV*, 2024, pp. 143–160.

[18] C. Kim, K. Min, M. Patel, S. Cheng, and Y. Yang, "Wouaf: Weight modulation for user attribution and fingerprinting in text-to-image diffusion models," in *CVPR*, 2024, pp. 8974–8983.

[19] V. Asnani, X. Yin, T. Hassner, and X. Liu, "Reverse engineering of generative models: Inferring model hyperparameters from generated images," *TPAMI*, 2023.

[20] N. Yu, L. S. Davis, and M. Fritz, "Attributing fake images to gans: Learning and analyzing gan fingerprints," in *ICCV*, 2019, pp. 7556–7566.

[21] N. Yu, V. Skripniuk, D. Chen, L. Davis, and M. Fritz, "Responsible disclosure of generative models using scalable fingerprinting," in *ICLR*, 2021.

[22] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz, "Artificial fingerprinting for generative models: Rooting deepfake attribution in training data," in *ICCV*, 2021, pp. 14 448–14 457.

[23] C. Kim, Y. Ren, and Y. Yang, "Decentralized attribution of generative models," in *ICLR*, 2021.

[24] J. Fei, Y. Dai, Z. Xia, F. Huang, and J. Zhou, "Omnimark: Efficient and scalable latent diffusion model fingerprinting," in *AAAI*, vol. 39, no. 16, 2025, pp. 16 550–16 558.

[25] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "Do gans leave artificial fingerprints?" in *MIPR*, 2019, pp. 506–511.

[26] L. Guarnera, O. Giudice, M. Nießner, and S. Battiato, "On the exploitation of deepfake model recognition," in *CVPR*, 2022, pp. 61–70.

[27] T. Yang, Z. Huang, J. Cao, L. Li, and X. Li, "Deepfake network architecture attribution," in *AAAI*, vol. 36, no. 4, 2022, pp. 4662–4670.

[28] T. Bui, N. Yu, and J. Collomosse, "Repmix: Representation mixing for robust attribution of synthesized images," in *ECCV*, 2022, pp. 146–163.

[29] Z. Wang, C. Chen, Y. Zeng, L. Lyu, and S. Ma, "Where did i come from? origin attribution of ai-generated images," *NeurIPS*, vol. 36, pp. 74 478–74 500, 2023.

[30] Z. Sun, S. Chen, T. Yao, B. Yin, R. Yi, S. Ding, and L. Ma, "Contrastive pseudo learning for open-world deepfake attribution," in *ICCV*, 2023, pp. 20 882–20 892.

[31] T. Yang, D. Wang, F. Tang, X. Zhao, J. Cao, and S. Tang, "Progressive open space expansion for open-set model attribution," in *CVPR*, 2023, pp. 15 856–15 865.

[32] F. Liu, H. Luo, Y. Li, P. Torr, and J. Gu, "Which model generated this image? a model-agnostic approach for origin attribution," in *ECCV*, 2024, pp. 282–301.

[33] J. Li, H. Wang, S. Li, Z. Qian, X. Zhang, and A. V. Vasilakos, "Are handcrafted filters helpful for attributing ai-generated images?" in *ACM MM*, 2024, pp. 10 698–10 706.

[34] Z. Sun, S. Chen, T. Yao, R. Yi, S. Ding, and L. Ma, "Rethinking open-world deepfake attribution with multi-perspective sensory learning," *IJCV*, pp. 1–24, 2024.

[35] S. Baxevanakis, M. Schinas, and S. Papadopoulos, "Do deepfake attribution models generalize?" in *MAD*, 2025, pp. 45–54.

[36] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *IEEE S&P*, 2018, pp. 36–52.

[37] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *ICLR*, 2018.

[38] Y. Tay, D. Bahri, C. Zheng, C. Brunk, D. Metzler, and A. Tomkins, "Reverse engineering configurations of neural text generation models," in *ACL*, 2020, pp. 275–279.

[39] Y. Zhang, R. Yasaei, H. Chen, Z. Li, and M. A. Al Faruque, "Stealing neural network structure through remote fpga side-channel analysis," *TIFS*, vol. 16, pp. 4377–4388, 2021.

[40] D. Ippolito, N. Carlini, K. Lee, M. Nasr, and Y. W. Yu, "Reverse-engineering decoding strategies given blackbox access to a language generation system," in *ACM INLG*, 2023, pp. 396–406.

[41] A. Naseh, K. Krishna, M. Iyyer, and A. Houmansadr, "Stealing the decoding algorithms of language models," in *ACM CCS*, 2023, pp. 1835–1849.

[42] B. Zhang, X. He, Y. Shen, T. Wang, and Y. Zhang, "A plot is worth a thousand words: model information stealing attacks via scientific plots," in *USENIX Security*, 2023, pp. 5289–5306.

[43] R. Li, J. Yu, C. Li, W. Luo, Y. Yuan, and G. Wang, "Dream: Domain-agnostic reverse engineering attributes of black-box model," *TKDE*, 2024.

[44] X. Guo, V. Asnani, S. Liu, and X. Liu, "Tracing hyperparameter dependencies for model parsing via learnable graph pooling network," *NeurIPS*, vol. 37, pp. 116 899–116 932, 2024.

[45] Y. Yao, X. Guo, V. Asnani, Y. Gong, J. Liu, X. Lin, X. Liu, S. Liu *et al.*, "Reverse engineering of deceptions on machine-and human-centric attacks," *Foundations and Trends® in Privacy and Security*, vol. 6, no. 2, pp. 53–152, 2024.

[46] L. Gao, W. Liu, K. Liu, and J. Wu, "Augsteal: Advancing model steal with data augmentation in active learning frameworks," *TIFS*, vol. 19, pp. 4728–4740, 2024.

[47] N. Carlini, D. Paleka, K. Dvijotham, T. Steinke, J. Hayase, A. F. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy *et al.*, "Stealing part of a production language model," in *ICML*, 2024, pp. 5680–5705.

[48] H. J. Song, M. Khayatkhoei, and W. AbdAlmageed, "Manifpt: Defining and analyzing fingerprints of generative models," in *CVPR*, 2024, pp. 10 791–10 801.

[49] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *CVPR*, 2018, pp. 8789–8797.

[50] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *CVPR*, 2020, pp. 8188–8197.

[51] C. Tan, Y. Zhao, S. Wei, G. Gu, P. Liu, and Y. Wei, "Rethinking the up-sampling operations in cnn-based generative network for generalizable deepfake detection," in *CVPR*, 2024, pp. 28 130–28 139.

[52] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *ICML*, 2020, pp. 1691–1703.

[53] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, "Autoregressive image generation using residual quantization," in *CVPR*, 2022, pp. 11 523–11 532.

[54] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *JMLR*, vol. 22, no. 57, pp. 1–64, 2021.

[55] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *TPAMI*, vol. 43, no. 11, pp. 3964–3979, 2020.

[56] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," in *ICASSP*, 2019, pp. 8261–8265.

[57] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1251–1258.

[58] M. Du, S. Pentyala, Y. Li, and X. Hu, "Towards generalizable deepfake detection with locality-aware autoencoder," in *CIKM*, 2020, pp. 325–334.

[59] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," in *ICASSP*, 2019, pp. 2307–2311.

[60] T. M. Wani, R. Gulzar, and I. Amerini, "Abc-capsnet: Attention based cascaded capsule network for audio deepfake detection," in *CVPRW*, 2024, pp. 2464–2472.

[61] H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, and N. Yu, "Multi-attentional deepfake detection," in *CVPR*, 2021, pp. 2185–2194.

[62] K. Sun, H. Liu, T. Yao, X. Sun, S. Chen, S. Ding, and R. Ji, "An information theoretic approach for attention-driven face forgery detection," in *ECCV*, 2022, pp. 111–127.

[63] A. Luo, R. Cai, C. Kong, Y. Ju, X. Kang, J. Huang, and A. C. K. Life, "Forgery-aware adaptive learning with vision transformer for generalized face forgery detection," *TCSVT*, 2024.

[64] I. Masi, A. Killekar, R. M. Mascarenhas, S. P. Gurudatt, and W. AbdAlmageed, "Two-branch recurrent network for isolating deepfakes in videos," in *ECCV*, 2020, pp. 667–684.

[65] Y. Qian, G. Yin, L. Sheng, Z. Chen, and J. Shao, "Thinking in frequency: Face forgery detection by mining frequency-aware clues," in *ECCV*, 2020, pp. 86–103.

[66] H. Liu, Z. Tan, Q. Chen, Y. Wei, Y. Zhao, and J. Wang, "Unified frequency-assisted transformer framework for detecting and grounding multi-modal manipulation," *IJCV*, vol. 133, no. 3, pp. 1392–1409, 2025.

[67] Z. Yu, J. Ni, Y. Lin, H. Deng, and B. Li, "Diffforensics: Leveraging diffusion prior to image forgery detection and localization," in *CVPR*, 2024, pp. 12 765–12 774.

[68] X. Guo, X. Liu, I. Masi, and X. Liu, "Language-guided hierarchical fine-grained image forgery detection and localization," *IJCV*, vol. 133, no. 5, pp. 2670–2691, 2025.

[69] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," in *CVPR*, 2020, pp. 8695–8704.

[70] C. Wang and W. Deng, "Representative forgery mining for fake face detection," in *CVPR*, 2021, pp. 14 923–14 932.

[71] L. Chen, Y. Zhang, Y. Song, L. Liu, and J. Wang, "Self-supervised learning of adversarial example: Towards good generalizations for deepfake detection," in *CVPR*, 2022, pp. 18 710–18 719.

[72] C. Li, Z. Huang, D. P. Paudel, Y. Wang, M. Shahbazi, X. Hong, and L. Van Gool, "A continual deepfake detection benchmark: Dataset, methods, and essentials," in *WACV*, 2023, pp. 1339–1349.

[73] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy–move forgery detection," *TIFS*, vol. 10, no. 11, pp. 2284–2297, 2015.

[74] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Busternet: Detecting copy-move image forgery with source/target localization," in *ECCV*, 2018, pp. 168–184.

[75] Y. Li, Y. He, C. Chen, L. Dong, B. Li, J. Zhou, and X. Li, "Image copy-move forgery detection via deep patchmatch and pairwise ranking learning," *TIP*, 2024.

[76] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection," in *ACM MM*, 2017, pp. 1480–1502.

[77] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency," in *ECCV*, 2018, pp. 101–117.

[78] A. Xiang, J. Zhang, Q. Yang, L. Wang, and Y. Cheng, "Research on splicing image detection algorithms based on natural image statistical characteristics," *arXiv preprint arXiv:2404.16296*, 2024.

[79] B. Yu, W. Li, X. Li, J. Lu, and J. Zhou, "Frequency-aware spatiotemporal transformers for video inpainting detection," in *ICCV*, 2021, pp. 8188–8197.

[80] M. Liu, X. Di, and M. Liao, "Image inpainting detection via dual guidance of uncertainty and precise boundary information," *TCSVT*, 2025.

[81] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," in *WIFS*, 2018, pp. 1–7.

[82] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *ICCV*, 2019, pp. 1–11.

[83] Z. Lu, D. Huang, L. Bai, J. Qu, C. Wu, X. Liu, and W. Ouyang, "Seeing is not always believing: Benchmarking human and model perception of ai-generated images," *NeurIPS*, vol. 36, pp. 25 435–25 447, 2023.

[84] Y. Li, X. Liu, X. Wang, B. S. Lee, S. Wang, A. Rocha, and W. Lin, "Fakebench: Probing explainable fake image detection via large multimodal models," *TIFS*, 2025.

[85] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal {DNN} models with lossless inference accuracy," in *USENIX Security*, 2021.

[86] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *USENIX Security*, 2016, pp. 601–618.

[87] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "Calpa-net: Channel-pruning-assisted deep residual network for steganalysis of digital images," *TIFS*, vol. 16, pp. 131–146, 2020.

[88] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *TPAMI*, vol. 44, no. 11, pp. 7436–7456, 2021.

[89] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *ICLR*, 2017.

[90] Y. Nirkin, L. Wolf, and T. Hassner, "Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation," in *CVPR*, 2021, pp. 4061–4070.

[91] B. Sen, G. Singh, A. Agarwal, R. Agaram, M. Krishna, and S. Sridhar, "Hyp-nerf: Learning improved nerf priors using a hypernetwork," *NeurIPS*, vol. 36, pp. 51 050–51 064, 2023.

[92] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *TIP*, vol. 26, no. 7, pp. 3142–3155, 2017.

[93] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTAS*, 2010, pp. 249–256.

[94] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[95] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[96] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016, pp. 2818–2826.

[97] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, vol. 31, no. 1, 2017.

[98] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain, "On the detection of digital face manipulation," in *CVPR*, 2020, pp. 5781–5790.

[99] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, vol. 33, pp. 6840–6851, 2020.

[100] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021.

[101] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "Stable diffusion: A latent text-to-image diffusion model," https://github.com/CompVis/stable-diffusion, 2022.

[102] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *ICML*, 2022, pp. 16 784–16 804.

[103] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," in *ICLR*, 2024.

[104] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *JMLR*, vol. 9, no. 11, 2008.

[105] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017, pp. 214–223.

[106] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[107] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-df: A large-scale challenging dataset for deepfake forensics," in *CVPR*, 2020, pp. 3207–3216.

[108] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[109] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *NeurIPS*, vol. 30, 2017.

[110] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *ICLR*, 2018.

[111] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

Yu Zheng received the BS and PhD degrees from the Department of Automation, Tsinghua University, in 2019 and 2024, respectively. He is currently a postdoctoral researcher with the Department of Automation, Tsinghua University. His research interests include AI safety and 3D understanding. He has published 10 scientific papers in TPAMI, TIP, CVPR, ECCV and ICLR. He serves as a regular reviewer member for TIP, TMM, TCSVT, CVPR, ICCV, ECCV, AAAI and ICME.

Zhuoxun Li received the B.Eng. degree from Xinya College, Tsinghua University in 2022. His research interests include computer vision, with an emphasis on detection and attribution of AI-generated visual content.

Bingyao Yu received the B.S. and Ph.D. degrees both in the Department of Automation, Tsinghua University, China, in 2018 and 2023. His current research interests include computer vision, deep learning, face forgery detection and face anti-spoofing. He serves as a regular reviewer member for a number of journals and conferences, e.g. TIP, ICML, ICLR, NeurIPS, CVPR, ICCV, and ECCV.

Jiwen Lu (Fellow, IEEE) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in electrical engineering from the Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2012. From 2011 to 2015, He was with the Advanced Digital Sciences Center, Singapore. In November 2015, he joined the Department of Automation, Tsinghua University, where he is currently a full professor and the deputy chair of the department. His current research interests include computer vision, pattern recognition, embodied intelligent and artificial intelligence safety. He serves/has served as the Co-Editor-of-Chief for Pattern Recognition Letters, an Associate Editor for the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, and the IEEE Transactions on Biometrics, Behavior, and Identity Sciences, and Pattern Recognition. He was a recipient of the National Natural Science Funds for Distinguished Young Scholar. He is an IEEE/IAPR Fellow.

Jie Zhou (Fellow, IEEE) received the BS and MS degrees both from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the PhD degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995. From then to 1997, he served as a postdoctoral fellow in the Department of Automation, Tsinghua University, Beijing, China. Since 2003, he has been a full professor in the Department of Automation, Tsinghua University. His research interests include computer vision, pattern recognition, and image processing. In recent years, he has authored more than 300 papers in peer-reviewed journals and conferences. Among them, more than 100 papers have been published in top journals and conferences such as the IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Image Processing, and CVPR. He is an associate editor for the IEEE Transactions on Pattern Analysis and Machine Intelligence and two other journals. He received the National Outstanding Youth Foundation of China Award. He is an IEEE/IAPR Fellow.