TRAINING RECIPES LEAK THROUGH BEHAVIORAL FINGERPRINTS IN TEXT-TO-IMAGE DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A deployed text-to-image diffusion model exposes generated images while keeping its fine-tuning procedure hidden. Training choices such as LoRA rank and scaling, learning rate, batch size, and training steps therefore appear inaccessible to a black-box user. We show that generated images can nevertheless reveal these choices. We present the first systematic study of whether fine-tuning hyperparameters of text-to-image diffusion models can be inferred from generated images alone. Given only black-box query access, we probe a target model with a fixed set of prompts and summarize its generations into a behavioral fingerprint using semantic, perceptual, and low-level image statistics. We then train predictors on fingerprints from shadow models with known recipes to infer the target model's hidden hyperparameters. Across 297 LoRA models, we recover five of six varied hyperparameters at rates significantly above chance. LoRA rank leaves the strongest fingerprint: we recover it with 85.4% exact accuracy and 100% accuracy within one grid step. When we consider all six predictions jointly, we recover the complete recipe exactly in 9.8% of cases, compared with a 0.21% uniform-chance baseline, a 48$\times$ increase over chance. We continue to recover rank significantly above chance across changes in model architecture, base-model weights, and fine-tuning data, and we find that controlling for adapter drift does not remove the signal. We also recover rank using only 96 generated images and continue to detect its fingerprint under substantial output perturbations. In contrast, LoRA dropout exhibits weak and classifier-dependent evidence of recovery. These results show that fine-tuning leaves persistent but uneven behavioral fingerprints in generated images, allowing black-box outputs to reveal parts of an otherwise hidden training recipe.

1 INTRODUCTION

Text-to-image diffusion models provide a powerful foundation for image generation (Ho et al., 2020; Rombach et al., 2022; Saharia et al., 2022), and practitioners commonly adapt pretrained models to new subjects, styles, and domains. Parameter-efficient methods such as Low-Rank Adaptation (LoRA) make this process practical by updating a small number of additional parameters while keeping the base model fixed (Hu et al., 2022). The resulting model, however, depends not only on its training data and base checkpoint but also on choices such as LoRA rank and scaling, learning rate, batch size, and fine-tuning steps (Hu et al., 2022; Wang et al., 2025; Yen et al., 2025). These choices can encode valuable training know-how (Wang & Gong, 2018), yet a generation API does not explicitly reveal them.

This paper asks whether generated images reveal these choices anyway: *does a fine-tuning recipe leave enough information in a model's black-box behavior to be recovered?* Prior work shows that deployed models can leak information beyond what their interfaces explicitly expose. Model extraction can recover aspects of a model through prediction interfaces (Tramèr et al., 2016), and hyperparameter-stealing attacks can recover training choices under stronger access assumptions (Wang & Gong, 2018). For diffusion models, privacy attacks have instead shown that model behavior can reveal information about the training data, such as whether a particular example was used for training (Duan et al., 2023; Tang et al., 2024; Li et al., 2025; Pang & Wang, 2025). Whether generated images also reveal the *fine-tuning procedure itself* remains largely unexplored.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We show that they do. Our key observation is that different fine-tuning recipes leave subtle but measurable differences in generated behavior. We capture these differences with a *behavioral fingerprint*: a compact representation that aggregates semantic, perceptual, and low-level statistics over generations from a controlled set of prompts. We construct shadow diffusion models with known recipes, extract their fingerprints, and learn to map these fingerprints to individual training choices. We then apply the learned mapping to unseen models using only their generated images.

Across 297 LoRA models, we find evidence of above-chance recovery for five of six varied hyperparameters, although statistical significance depends on the predictor. LoRA rank leaves the strongest fingerprint, reaching 85.4% classification accuracy. For all 48 models with unseen rank values, the two highest-probability classes bracket the true rank, while continuous predictions preserve rank ordering with Spearman $\rho = 0.813$. The leakage persists under changes in model architecture, base-model weights, and fine-tuning data, while LoRA dropout exhibits only weak and classifier-dependent evidence of recovery. When we consider all six hyperparameters jointly, we recover the complete recipe exactly in 9.8% of cases, compared with a 0.21% uniform-chance baseline—a 48$\times$ increase over chance. These results show that fine-tuning leaves persistent but uneven behavioral fingerprints in generated images.

Our contributions are:

- To the best of our knowledge, we present the first systematic study of *training-recipe inference* for text-to-image diffusion models, investigating whether black-box generations reveal the fine-tuning choices that produced a model.
- We introduce a behavioral-fingerprinting approach that uses shadow models with known recipes to infer hidden training choices from semantic, perceptual, and low-level statistics of generated images. Across 297 LoRA models, we find evidence of statistically significant recovery for five of six varied hyperparameters across the evaluated predictors.
- We show that recipe leakage is structured but heterogeneous. LoRA rank reaches 85.4% classification accuracy, exhibits systematic behavior on unseen rank values, and remains detectable across model and data shifts. In contrast, evidence of LoRA dropout recovery is weak and classifier-dependent.

2 Related Work

Model Extraction and Hyperparameter Stealing. Black-box prediction interfaces can reveal information beyond their intended outputs. Model extraction attacks recover model functionality or parameters through query access (Tramèr et al., 2016). More closely related, Wang and Gong (Wang & Gong, 2018) showed that training hyperparameters can themselves be inferred from trained models, developing attacks for classical machine-learning models and neural networks. Our setting differs in both interface and target: we infer multiple fine-tuning choices of text-to-image diffusion models from generated images alone, without access to model parameters, gradients, adapters, or prediction logits.

Confidentiality and Privacy in Generative Models. Generative models can expose information about their training data. Diffusion models can memorize and reproduce training examples (Carlini et al., 2023), while membership-inference attacks determine whether individual samples contributed to training (Duan et al., 2023; Tang et al., 2024; Li et al., 2025; Pang & Wang, 2025). Pang and Wang (Pang & Wang, 2025), in particular, study black-box membership inference against fine-tuned diffusion models using similarities between generated outputs and candidate training samples. Beyond individual records, property-inference attacks recover aggregate properties of training data (Ganju et al., 2018; Zhang et al., 2021; Wang et al., 2024). These works ask *what data* contributed to training. We instead ask *how the model was fine-tuned*: our target is the training recipe itself. The unit of inference therefore shifts from individual records or dataset properties to the fine-tuned model and its training run.

Model Fingerprinting and Watermarking. Model fingerprinting and watermarking use characteristic or deliberately embedded signals to identify models, verify ownership, or trace generated content. Tree-Ring Watermarks embed signals into the diffusion sampling process (Wen et al., 2023), while adversarial fingerprinting methods identify neural networks from distinctive responses

to crafted queries (Xu et al., 2024). We use *behavioral fingerprints* for a different purpose. Our method introduces no watermark or trigger and does not attempt to identify individual models. Instead, it aggregates naturally occurring differences in generated images to infer the fine-tuning choices that produced a model.

3 Problem Formulation and Threat Model

3.1 Problem Formulation

Let M_{θ_0} denote a pretrained text-to-image diffusion model, $\mathcal{D}$ a fine-tuning dataset, and $\mathcal{A}$ a fine-tuning procedure. A training recipe $\lambda = (\lambda_1, \ldots, \lambda_K)$ specifies K fine-tuning hyperparameters. The resulting model is

$$M_\lambda = \mathcal{A}(M_{\theta_0}, \mathcal{D}; \lambda). \tag{1}$$

In our primary setting, $K = 6$, comprising LoRA rank, LoRA scaling, learning rate, batch size, fine-tuning steps, and dropout. Our primary evaluation fixes the base model and fine-tuning dataset to isolate recipe-dependent differences.

The deployed model provides black-box generation access. Let $\mathcal{P}$ be a set of text prompts and $\mathcal{S}$ a set of inference seeds. For a prompt $p \in \mathcal{P}$ and seed $s \in \mathcal{S}$, the attacker observes the generated image $M_\lambda(p, s)$. We represent the collected observations as

$$\mathcal{O}(M_\lambda) = \big(M_\lambda(p, s)\big)_{(p,s)\in\mathcal{P}\times\mathcal{S}}. \tag{2}$$

All other inference settings are fixed unless explicitly varied in a robustness experiment.

3.2 Threat Model

Adversary Goal. The adversary aims to recover the victim's hidden fine-tuning recipe from its generated images. For each recipe component λ_k, where $k \in \{1, \ldots, K\}$, the attacker seeks to estimate its value using behavioral information extracted from black-box observations. The attacker targets properties of the fine-tuning procedure rather than model parameters or individual training records. We consider both component-wise recovery and joint recovery of the complete recipe.

Adversary Capabilities and Knowledge. Following prior work on black-box inference against generative and diffusion models (Duan et al., 2023; Li et al., 2025; Pang & Wang, 2025), we consider a black-box interface that accepts attacker-selected text prompts and inference seeds and returns the resulting images. The attacker cannot inspect the victim's internal state or access its fine-tuning weights, LoRA adapters, gradients, optimizer state, training trajectory, configuration files, or training logs. The attacker also receives no labels revealing the victim's recipe.

We assume that the attacker knows and can execute the public base-model checkpoint used to fine-tune the victim. The attacker also has access to auxiliary images from the same data domain as the victim's training corpus and can independently fine-tune shadow models under known recipes without requiring the victim's exact training examples.

Our primary evaluation uses the same fine-tuning corpus for shadow and victim models to isolate recipe-dependent behavioral differences. We then relax this assumption by evaluating victims trained on disjoint images from the same domain. This evaluation uses matched training recipes to isolate the effect of training-data overlap. We further investigate transferability across different base-model weights, model architectures, and fine-tuning domains.

4 Behavioral Fingerprinting Attack

4.1 Attack Overview

Our attack exploits differences in generation behavior induced by fine-tuning choices. Rather than inferring a recipe from individual images, we aggregate observations across controlled prompts and inference seeds into a model-level behavioral fingerprint.

As illustrated in Figure 1, the attack proceeds in three stages. First, the attacker constructs N shadow diffusion models with known training recipes $\{\lambda_i\}_{i=1}^N$, where N denotes the number of shadow models and λ_i is the recipe used to train shadow model i.

Second, the attacker queries each shadow model M_{λ_i} using the same probing protocol and extracts a fingerprint

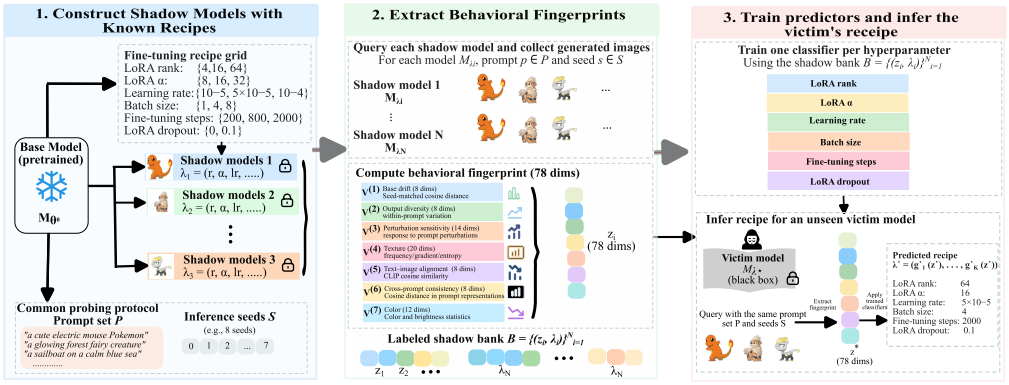

Figure 1: Overview of the proposed black-box training-recipe inference framework. The attacker fine-tunes shadow diffusion models under known recipes and queries them using a common probing protocol. Behavioral fingerprints extracted from the generated images are used to train hyperparameter predictors, which are subsequently applied to an unseen victim using only its generated images.

$$\mathbf{z}_i = F_{\theta_0}\left(\mathcal{O}(M_{\lambda_i})\right) \in \mathbb{R}^d, \tag{3}$$

where F_{θ_0} denotes the fingerprint extractor conditioned on the known base checkpoint M_{θ_0}, and d is the fingerprint dimension. The extractor uses paired base-model generations when computing base drift. Our implementation uses $d = 78$. The resulting labeled shadow bank is

$$\mathcal{B} = \{(\mathbf{z}_i, \lambda_i)\}_{i=1}^{N}. \tag{4}$$

Third, the attacker trains predictors that associate behavioral fingerprints with individual recipe components and applies them to an unseen victim using only its generated images. The victim does not contribute to predictor training.

4.2 Shadow-Model Construction

The attacker independently fine-tunes shadow diffusion models from a known base checkpoint under different recipes, while controlling other training conditions.

Our primary evaluation uses 297 LoRA models partitioned at the model level into training shadows and held-out evaluation models. Thus, N denotes the number of training shadows in a given split, rather than the entire model collection. The recipes vary the targeted hyperparameters while approximately balancing their marginal distributions.

Each shadow model is queried using the common prompt and seed sets $\mathcal{P}$ and $\mathcal{S}$. Its fingerprint and known recipe form one labeled observation in $\mathcal{B}$. Appendix A details the recipe grid, bank composition, and training configuration.

4.3 Behavioral Fingerprint

We construct a behavioral fingerprint that captures recipe-dependent differences in generation behavior. It combines semantic, perceptual, and low-level image statistics measured across controlled prompts and inference seeds.

For shadow model i, we concatenate seven feature blocks into

$$\mathbf{z}_i = \left[\mathbf{v}_i^{(1)} \| \cdots \| \mathbf{v}_i^{(7)}\right] \in \mathbb{R}^{78}, \tag{5}$$

where $\|$ denotes vector concatenation and $\mathbf{v}_i^{(j)}$ denotes the jth feature block. Their respective dimensions are $(8, 8, 14, 20, 8, 8, 12)$.

The fingerprint comprises seven feature blocks. $\mathbf{v}^{(1)}$ measures drift from the base model by comparing generations produced under matching prompts and seeds. $\mathbf{v}^{(2)}$ captures output diversity through differences among generations of the same prompt. $\mathbf{v}^{(3)}$ measures sensitivity to controlled prompt perturbations. $\mathbf{v}^{(4)}$ summarizes texture through frequency, gradient, and entropy statistics. $\mathbf{v}^{(5)}$ measures text–image alignment, while $\mathbf{v}^{(6)}$ captures consistency across different prompts. Finally,

$\mathbf{v}^{(7)}$ summarizes color and luminance. Together, these blocks characterize how a model responds to controlled changes in its inputs and sampling conditions. Appendix B provides their exact definitions, aggregation rules, and dimensionalities.

For embedding-based measurements, let $E(x)$ denote the ℓ_2-normalized representation of image x produced by a frozen visual encoder. As an example, we measure base-model drift using the cosine distance between generations from the fine-tuned model and its base checkpoint:

$$d_{\text{base}}(p, s) = 1 - \langle E(M_{\lambda_i}(p, s)), E(M_{\theta_0}(p, s)) \rangle , \tag{6}$$

We aggregate these distances across matched prompts and seeds to obtain $\mathbf{v}_i^{(1)}$.

We instantiate the embedding-based feature blocks separately using frozen DINOv2, CLIP, and DeiT encoders, producing a 78-dimensional fingerprint for each encoder. The pixel-level blocks remain unchanged across encoder choices. For simplicity, we omit the visual encoder index from the fingerprint extractor F_{θ_0}.

4.4 Recipe Inference

We formulate recipe inference as supervised learning over model-level behavioral fingerprints. For each recipe component $k \in \{1, \ldots, K\}$, let $\mathcal{Y}_k$ denote its set of candidate values. Using the labeled shadow bank $\mathcal{B}$, the attacker learns a predictor $g_k^\star$ for component k:

$$g_k^\star = \arg\min_{g \in \mathcal{G}_k} \frac{1}{N} \sum_{i=1}^{N} \ell_k \left(g(\mathbf{z}_i), \lambda_{i,k} \right) , \tag{7}$$

where $\mathcal{G}_k$ denotes the candidate predictor class, ℓ_k is its training loss, and $\lambda_{i,k} \in \mathcal{Y}_k$ is the value of component k in shadow model i.

Let $M_{\lambda^\star}$ denote an unseen victim model with hidden recipe $\lambda^\star$. The attacker collects its black-box observations and extracts the fingerprint

$$\mathbf{z}^\star = F_{\theta_0} \left(\mathcal{O}(M_{\lambda^\star}) \right) . \tag{8}$$

The estimated recipe is

$$\hat{\lambda}^\star = (g_1^\star(\mathbf{z}^\star), \ldots, g_K^\star(\mathbf{z}^\star)) . \tag{9}$$

We evaluate component-wise accuracy and joint recovery, which requires all recipe components to be predicted correctly. Training and evaluation use disjoint model-level splits. In the primary setting, predictors classify each hyperparameter among candidate values represented in the shadow bank. We separately evaluate whether rank fingerprints generalize to values outside the training grid.

5 Experimental Evaluation

5.1 Experimental Setup

Models and Probing. We evaluate 297 Stable Diffusion v1.5 models fine-tuned with LoRA across six recipe dimensions: rank, scaling, learning rate, batch size, training steps, and dropout. The collection comprises one 27-model factorial bank and two 135-model banks. All models are trained on the same 833-image Pokémon dataset and queried using 96 prompts and eight inference seeds, yielding 768 images per model. We extract 78-dimensional behavioral fingerprints using DINOv2, CLIP, and DeiT. Appendices A and B provide the training and fingerprint details.

Predictors and Evaluation. We evaluate RF, XGBoost, logistic regression, and two multi-head neural networks (MLP-Shared and MLP-Split). We use three-fold, ten-repeat model-level cross-validation stratified by LoRA rank, fitting preprocessing exclusively on training folds. We report mean accuracy and standard deviation across repeats and compare against uniform, majority-class, and stratified-random baselines. A fixed 300-tree RF provides a selection-independent reference for component-wise and joint recovery.

Statistical Testing. We assess recovery using 1,000 label permutations per predictor and hyperparameter, with each permutation evaluated using a single three-fold pass. We apply Benjamini–Hochberg correction across the six hyperparameters separately for each predictor and visual encoder. Since this procedure does not account for selection among the five predictors, we treat best-of-five results as exploratory. Further details appear in Appendix C.4.

Table 1: Individual hyperparameter recovery using DINOv2 fingerprints from 297 LoRA models. Accuracy (%) is reported as mean ± standard deviation over ten repetitions of three-fold model-level cross-validation. Bold indicates the highest accuracy per row. † denotes results that are not significant after multiple-testing correction ($q \geq 0.05$). Significance tests are predictor-specific and do not correct for best-of-five selection.

Hyperparameter	Chance	Majority	Stratified	RF	XGB	LogReg	MLP-Shared	MLP-Split
LoRA rank	33.3	34.0 ± 0.0	34.4 ± 1.6	$\mathbf{85.4 \pm 1.5}$	84.6 ± 1.6	81.3 ± 1.7	77.3 ± 0.9	84.5 ± 1.2
LoRA scaling	33.3	38.4 ± 0.0	35.5 ± 1.6	$\mathbf{72.8 \pm 2.1}$	71.4 ± 1.2	62.7 ± 2.0	57.1 ± 1.6	71.3 ± 2.4
Learning rate	33.3	29.8 ± 2.0	33.1 ± 2.7	54.4 ± 3.7	$\mathbf{55.7 \pm 2.9}$	49.4 ± 3.2	48.4 ± 2.4	53.1 ± 2.6
Batch size	33.3	36.1 ± 3.3	33.6 ± 2.6	$\mathbf{49.4 \pm 2.0}$	49.0 ± 1.4	44.4 ± 2.5	39.4 ± 2.5	47.4 ± 2.8
Training steps	33.3	31.6 ± 1.8	33.8 ± 2.2	$\mathbf{55.9 \pm 2.1}$	55.6 ± 2.1	53.6 ± 2.9	48.5 ± 1.8	55.0 ± 1.8
LoRA dropout	50.0	53.8 ± 3.1	50.1 ± 2.7	$55.0 \pm 2.1^{\dagger}$	$\mathbf{56.4 \pm 2.0}$	55.2 ± 2.1	$52.8 \pm 2.0^{\dagger}$	$56.1 \pm 2.6^{\dagger}$

5.2 Training-Recipe Recovery

Table 1 reports individual hyperparameter recovery using DINOv2 fingerprints from 297 LoRA models. All five predictors are evaluated on held-out models using repeated model-level cross-validation.

Individual Hyperparameter Recovery. LoRA rank exhibits the strongest signal, reaching 85.4% accuracy against a 34.4% stratified baseline (Table 1). LoRA scaling reaches 72.8%, compared with a 38.4% majority-class baseline. Learning rate, training steps, and batch size also show statistically significant recovery, with the fixed RF achieving 54.4%, 55.9%, and 49.4% accuracy, respectively. Tree-based predictors achieve the highest accuracy across all six hyperparameters. The fixed RF remains within 1.5 percentage points of the best predictor for every component.

LoRA Dropout. Dropout exhibits weaker and less consistent recovery. Although XGBoost reaches 56.4%, it exceeds the majority-class baseline by only 2.6 percentage points. XGBoost and logistic regression yield significant permutation-test results, whereas the fixed RF (55.0%, $q = 0.101$) and both neural predictors do not. These tests assess dependence between fingerprints and recipe labels, rather than improvement over the majority-class baseline, and do not correct for best-of-five selection. We therefore interpret dropout recovery as limited and predictor-dependent.

5.3 Joint Training-Recipe Recovery

Recovering individual hyperparameters does not necessarily imply recovery of the complete training recipe. We therefore evaluate exact joint recovery, requiring all six hyperparameters to be correctly predicted for the same held-out model. We use a fixed 300-tree RF for each component and align predictions across identical model-level cross-validation splits.

Exact Recipe Recovery. Table 2 reports exact recovery rates of 9.80%, 10.24%, and 9.60% using DINOv2, CLIP, and DeiT fingerprints, respectively. These correspond to approximately 47–50 times the theoretical uniform-guessing probability of 0.21% across the complete recipe grid.

Interpretation. Joint recovery is substantially more restrictive than component-wise prediction, requiring six simultaneously correct predictions. The comparable results across encoders suggest that joint recipe information is not confined to one visual representation. Nevertheless, complete recovery remains unsuccessful for most held-out models. Additional evaluation details appear in Appendix C.3.

5.4 Off-Grid Rank Generalization

We examine whether rank fingerprints generalize beyond the training grid $\{4, 16, 64\}$. Using the 297-model shadow bank, we evaluate 48 independently fine-tuned models with six unseen ranks: $\{6, 8, 12, 24, 32, 48\}$, with eight models per rank.

Experimental Design. We evaluate two complementary tasks. First, we test whether the two highest-probability RF classes bracket the true rank, against a 66.7% uniform-chance reference. Second, we train a regressor on $\log_2$ rank and measure Spearman correlation and mean absolute error (MAE) on unseen ranks.

Recovery of Unseen Ranks. Figure 2 shows how predicted class probabilities vary across unseen ranks. All three encoders achieve 100% bracketing accuracy, compared with a 66.7% uniform-

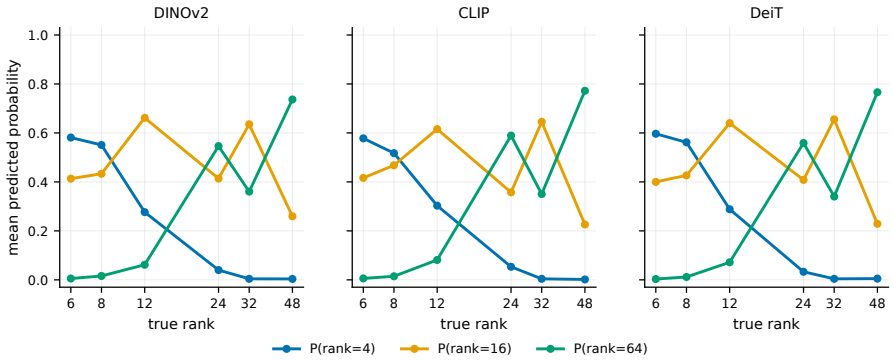

Figure 2: Rank-class probabilities for 48 models with six unseen LoRA ranks, using DINOv2, CLIP, and DeiT. Predictions are over training ranks $\{4, 16, 64\}$, not continuous ranks.

Table 2: Exact six-component recipe recovery using fixed random forests. Results are mean accuracy (%) $\pm$ standard deviation over ten repetitions of three-fold model-level cross-validation. Chance lift is relative to independent, uniform guessing.

Encoder	**Exact Recovery (%)**	**Chance Lift**
DINOv2	9.80 ± 1.37	$48\times$
CLIP	10.24 ± 1.66	$50\times$
DeiT	9.60 ± 1.20	$47\times$
Uniform chance	0.21	$1\times$

Table 3: LoRA rank recovery accuracy (%) under model and data shifts. All predictors are trained on the original shadow bank. Aware and blind denote the base checkpoint used to compute victim fingerprints. All transfer results shown are significant after Benjamini–Hochberg correction.

Victim setting	**DINOv2**	**CLIP**	**DeiT**
SD 2.1, aware	78.3	75.9	75.4
SD 2.1, blind	60.1	74.6	71.6
Different base weights	70.8	69.4	69.4
Different training domain	78.3	75.8	77.2
Uniform chance	33.3	33.3	33.3

chance reference. The corresponding Spearman correlations are 0.813, 0.782, and 0.807 for DINOv2, CLIP, and DeiT, with $\log_2$-rank MAEs of 0.682, 0.691, and 0.708, respectively. The probability transitions are not strictly monotonic, including an inversion between ranks 24 and 32. These results support rank ordering and bracketing, rather than exact recovery of continuous rank.

5.5 Generalization Across Model and Data Shifts

We examine whether fingerprints transfer across model architectures, base-model weights, and fine-tuning datasets without retraining the attacker's predictors.

Experimental Design. We evaluate three distribution shifts: 138 Stable Diffusion v2.1 victims, 72 victims using Realistic Vision v5.1, and 72 victims trained on Naruto images instead of Pokémon. For SD v2.1, we consider architecture-aware and architecture-blind settings, which use the victim's actual base checkpoint and the original shadow checkpoint, respectively, to compute base drift. All predictors are trained exclusively on the original 297-model shadow bank.

Transfer Results. Table 3 shows that rank recovery remains statistically significant across all evaluated shifts and visual encoders. For SD v2.1, architecture-aware recovery reaches 75.4–78.3%. Without the correct base checkpoint, DINOv2 accuracy decreases from 78.3% to 60.1%, while CLIP and DeiT retain 74.6% and 71.6%, respectively. Changing the base-model weights yields 69.4–70.8% accuracy, while changing the training domain yields 75.8–78.3%.

Other hyperparameters exhibit less consistent transfer. LoRA scaling remains recoverable across the architecture shift, whereas learning rate and batch size are less robust. These findings indicate that rank fingerprints transfer across the evaluated shifts, but do not establish comparable generalization for the complete training recipe. Detailed results appear in Appendix C.5.

5.6 Generalization to Disjoint Fine-Tuning Data

We examine whether recipe inference requires access to the victim's exact training images.

Experimental Design. We train additional victims on a disjoint Pokémon corpus, screened against the shadow-model corpus using perceptual hashes and DINOv2 embeddings. To isolate the effect of training-data overlap, we compare nine matched recipes across the original and disjoint corpora, fixing LoRA scaling at $\alpha = 16$.

Results. Rank recovery remains strong on disjoint training data: DINOv2 and DeiT retain 100% accuracy, while CLIP decreases from 100% to 91.1%. All three results are significant under permutation testing ($p \leq 0.0023$). In contrast, learning-rate accuracy decreases from 70.4–74.1% to 44.4–53.3%, with no significant disjoint-corpus recovery ($p = 0.341$). These results indicate that exact training-image overlap is unnecessary for rank recovery in this controlled same-domain setting, but do not establish robust learning-rate transfer. Detailed results appear in Appendix C.6.

5.7 Behavioral Fingerprint Ablation

We investigate the contribution of the seven fingerprint blocks using the 297-model collection, DINOv2 fingerprints, and the fixed RF predictor.

Individual Feature Blocks. Each block independently supports rank recovery, with accuracies ranging from 78.9% to 86.2%, compared with 85.4% for the complete fingerprint. Notably, the 12-dimensional color block achieves 86.2%, while texture and output diversity reach 82.8% and 82.6%, respectively. These results show that rank information is accessible even through pixel-level statistics that require no learned visual encoder.

Feature Redundancy. Removing any single block yields 84.0–86.3% rank accuracy, indicating that no individual block is essential to the observed recovery performance. Together, the individual-block and leave-one-out experiments suggest substantial predictive redundancy. However, the strong performance of color statistics does not establish a causal relationship between LoRA rank and image color. Detailed ablation results appear in Appendix C.7.

5.8 Controlling for Adapter Drift

We examine whether recipe recovery primarily reflects adapter-update magnitude rather than broader generation behavior.

Experimental Design. Using DINOv2 fingerprints, we evaluate three controls: prediction from adapter norms alone, recovery within three groups stratified by $\|B\|$, and recovery after regressing $\log \|B\|$ out of the behavioral features. Adapter statistics are used only for these controls and remain inaccessible to the black-box attacker.

Results. Adapter norms alone identify LoRA rank with 99.9% accuracy. Within drift-stratified groups, behavioral rank accuracy remains between 78.8% and 88.7%. Residualization reduces rank accuracy from 85.4% to 82.7% and scaling accuracy from 71.9% to 70.1% within the drift-control evaluation.

These findings indicate that the fingerprints retain predictive information beyond their linear association with the measured adapter norm. They do not establish complete independence from update magnitude or identify the underlying causal mechanism. Additional results appear in Appendix C.8.

5.9 Query Efficiency and Inference-Condition Sensitivity

Query Efficiency. We evaluate reduced probing budgets by varying the number of prompts and inference seeds independently. With 96 prompts, reducing sampling from eight to one image per prompt decreases DINOv2 rank accuracy from 85.4% to 84.0%. Learning rate, batch size, and training-step recovery remain comparable, whereas scaling accuracy decreases from 72.8% to 66.6%.

With one image per prompt, reducing the budget to eight prompts yields $82.9\% \pm 1.1\%$ rank accuracy across 20 prompt-subset replicates. This result uses the highest-performing evaluated predictor under pooled model-level cross-validation. Although it is numerically close to the full-budget result, the comparison does not control for predictor selection or establish equivalent performance on independently held-out victims. Detailed results appear in Appendices D.1 and D.2.

Inference-Condition Sensitivity. We evaluate rank recovery on 135 independently held-out victims under changes to the sampler, inference steps, and classifier-free guidance. Table 13 shows

that changing the sampler or inference-step count preserves 67.4–73.3% rank accuracy in the blind setting, compared with 77.9% without an inference shift.

Reducing guidance from 7.5 to 5.0 decreases accuracy to 51.0% in the blind setting. Matching the victim's inference configuration partially restores accuracy to 63.0%. Increasing guidance to 12.0 instead preserves approximately the original recovery accuracy.

These results indicate that rank leakage persists under the evaluated inference changes but is particularly sensitive to reduced guidance. Additional cross-encoder and reference-matching analyses appear in Appendix D.3.

5.10 Deployment-Level Defenses

We examine two deployment-level defenses: query restrictions and output perturbations.

Query Restrictions. Eight generated images achieve 82.9% rank recovery under pooled model-level cross-validation, compared with 84.0% using 96 images. These results suggest that restricting the query budget alone may provide limited protection in the evaluated setting, although the reported accuracies do not necessarily use identical predictor-selection protocols.

Output Perturbations. We evaluate 19 perturbation settings across Gaussian noise, JPEG compression, color quantization, and spatial downscaling. Using 135 held-out victims, we perturb only the victim outputs without retraining the attacker.

Gaussian noise ($\sigma = 32$) retains 94% of the original above-chance rank performance at 18.5 dB PSNR, whereas spatial downscaling retains 29% at 22.3 dB. Although these settings are not quality-matched, downscaling achieves greater leakage suppression despite its higher PSNR.

Rank remains statistically detectable under all evaluated perturbations. Learning rate loses significance only at JPEG quality 10. These findings demonstrate partial suppression under non-adaptive output perturbations. The complete results and experimental protocol appear in Appendix E.

6 Discussion

Our findings show that fine-tuning recipes leave recoverable behavioral fingerprints, although leakage varies across hyperparameters. LoRA rank exhibits the most consistent signal, including under unseen rank values and changes in model architecture, base-model weights, and training data. Other hyperparameters are less consistently recovered, while complete-recipe recovery remains challenging. Feature ablations and adapter-drift controls indicate that rank recovery draws on multiple behavioral measurements and is not fully explained by the measured linear association with adapter norms. These findings establish predictive associations rather than a causal mechanism. Our evaluation relies on a controlled LoRA training grid, a known base checkpoint, auxiliary training data, and independently trained shadow models. Generalization to arbitrary proprietary models remains unestablished, and our deployment-defense evaluation considers only non-adaptive attacks. Developing defenses that suppress recipe leakage while preserving generation quality remains an open problem.

7 Conclusion

We investigated training-recipe leakage in fine-tuned text-to-image diffusion models using behavioral fingerprints extracted solely from generated images. Across 297 LoRA models, we found that hidden hyperparameters are partially recoverable, with LoRA rank exhibiting the strongest and most consistent signal. Our evaluations of joint recovery, unseen rank values, distribution shifts, query efficiency, and deployment-level defenses further characterize the extent and limitations of this leakage. These findings demonstrate that black-box generation interfaces can expose information about otherwise hidden fine-tuning configurations and motivate further research into protecting training-recipe confidentiality.

AI Use Statement

We used large language models (e.g., ChatGPT and Writefull) solely for language polishing, improving tone, and checking grammar. Grammarly was also used for grammar correction. All research

ideas, experiments, analyses, and technical content were developed by the authors, who take full responsibility for the manuscript.

Ethics Statement

This work investigates the leakage of fine-tuning hyperparameters from text-to-image diffusion models through black-box access. Our experiments use independently fine-tuned models, publicly available pretrained checkpoints, and controlled evaluation settings. No human subjects or private user data were involved. Although the proposed attack could potentially expose proprietary training configurations, our objective is to characterize this security risk and motivate stronger defenses for deployed generative models. We evaluate deployment-level defenses and acknowledge the limitations of the proposed approach. We adhere to the ICLR Code of Ethics and present our findings to advance research on training-recipe confidentiality.

Reproducibility Statement

We provide the experimental methodology and implementation details necessary to reproduce our findings. The main paper describes the threat model, shadow-model construction, behavioral fingerprint extraction, and hyperparameter inference framework. The appendix provides additional training configurations, feature definitions, evaluation protocols, statistical analyses, and experimental results. We will release our implementation code, configuration files, and experiment scripts to facilitate independent reproduction of our results.

References

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270. USENIX Association, 2023.

Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *International Conference on Machine Learning*, pp. 8717–8730. PMLR, 2023.

Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 619–633, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jingwei Li, Jing Dong, Tianxing He, and Jingzhao Zhang. Towards black-box membership inference attack for diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34271–34287. PMLR, 2025.

Yan Pang and Tianhao Wang. Black-box membership inference attacks against fine-tuned diffusion models. In *Network and Distributed System Security Symposium*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pp. 36479–36494, 2022.

Table 4: Training-recipe grid used in the primary LoRA evaluation. Counts indicate the number of models at each value among the 297 successfully trained models.

Hyperparameter	Values (class order)	Models per class
LoRA rank	$\{4, 16, 64\}$	$99/97/101$
LoRA α	$\{8, 16, 32\}$	$94/114/89$
Learning rate	$\{10^{-5}, 5\times10^{-5}, 10^{-4}\}$	$100/101/96$
Batch size	$\{1, 4, 8\}$	$94/112/91$
Fine-tuning steps	$\{200, 800, 2000\}$	$96/94/107$
LoRA dropout	$\{0, 0.1\}$	$163/134$

Shuai Tang, Steven Wu, Sergul Aydore, Michael Kearns, and Aaron Roth. Membership inference attacks on diffusion models via quantile regression. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47819–47829. PMLR, 2024.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 601–618. USENIX Association, 2016.

Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 36–52. IEEE, 2018.

Lijin Wang, Jingjing Wang, Jie Wan, Lin Long, Ziqi Yang, and Zhan Qin. Property existence inference against generative models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 2423–2440. USENIX Association, 2024.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, 2025.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-rings watermarks: Invisible fingerprints for diffusion images. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tianlong Xu, Chen Wang, Gaoyang Liu, Yang Yang, Kai Peng, and Wei Liu. United we stand, divided we fall: Fingerprinting deep neural networks via adversarial trajectories. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Venkata Sai Surya Subramanyam Duvvuri, Inderjit Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. LoRA done RITE: Robust invariant transformation equilibration for LoRA optimization. In *International Conference on Learning Representations*, 2025.

Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in {Multi-Party} machine learning. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2687–2704, 2021.

A Shadow-Model Bank and Training Details

A.1 Main Recipe Grid and Bank Composition

Our primary evaluation comprises 297 successfully trained LoRA models spanning six recipe dimensions: LoRA rank, LoRA scaling factor α, learning rate, batch size, fine-tuning steps, and LoRA dropout. Table 4 reports the candidate values and their empirical frequencies. Class indices follow the numerical ordering shown in the table.

The 297 models are drawn from three independently constructed banks. The first contains 27 models forming a full factorial over learning rate, fine-tuning steps, and LoRA rank, with $\alpha = 16$, batch size 4, and dropout 0. The other two banks contribute 135 successfully trained AdamW models each. Their configurations originate from stratified draws over the larger Cartesian recipe space, with the second draw constructed to be disjoint from the first.

Stratified sampling approximately balances the marginal distribution of each recipe dimension, limiting gains from predicting a dominant class. Dropout is the least balanced dimension, with 163 models at 0 and 134 at 0.1. Its empirical majority-class baseline is therefore 54.9%, which we report alongside the 50% uniform-chance baseline.

We preserve the numerical ordering of each recipe dimension when computing ordinal metrics, including adjacent-class accuracy and mean class-index distance. LoRA ranks $\{4, 16, 64\}$ correspond to class indices $\{0, 1, 2\}$ and are equally spaced in $\log_2$ space. This ordering also motivates continuous prediction in log-rank space during our off-grid evaluation.

A.2 Fine-Tuning and Generation Configuration

We construct the primary model collection using Stable Diffusion v1.5. LoRA adapters are applied to the attention projections `to_q`, `to_k`, `to_v`, and `to_out.0`. Models are fine-tuned on 833 Pokémon images at 512×512 resolution using half-precision computation. Unless varied as part of the recipe grid, the remaining training configuration is held fixed.

We evaluate each model using a fixed probe suite of 96 prompts and eight inference seeds,

$$\mathcal{S} = \{0, \ldots, 7\},$$

yielding 768 generated images per model under the full probing configuration. Generation uses DPM-Solver++ with 30 inference steps and classifier-free guidance scale 7.5.

The same probing protocol is applied to training shadows and held-out evaluation models. Separate robustness experiments vary inference conditions to examine sensitivity to the generation configuration.

B Behavioral Fingerprint Details

B.1 Probe Organization and Notation

The behavioral fingerprint summarizes the generations of one fine-tuned model into a 78-dimensional vector.

Let $P = 96$ and $S = 8$ denote the numbers of probing prompts and inference seeds, respectively. Following the main text, let E denote a frozen visual encoder whose outputs are ℓ_2-normalized. For a given fine-tuned model M_{λ_i}, we write

$$E_{p,s} = E\left(M_{\lambda_i}(p, s)\right) \in \mathbb{R}^D, \tag{10}$$

where D is the embedding dimension. The corresponding base-model embedding is

$$E^0_{p,s} = E\left(M_{\theta_0}(p, s)\right). \tag{11}$$

The probe suite contains in-domain, near-domain, and out-of-domain prompts, together with controlled prompt perturbations. Depending on the feature block, statistics are aggregated over the three individual domains and their union:

$$\mathcal{D}_{\text{agg}} = \{\text{in}, \text{near}, \text{out}, \text{all}\}.$$

Prompt-perturbation features additionally distinguish synonym, shuffle, and typo perturbations.

The complete fingerprint is

$$\mathbf{z} = [\mathbf{v}^{(1)} \| \mathbf{v}^{(2)} \| \cdots \| \mathbf{v}^{(7)}] \in \mathbb{R}^{78}, \tag{12}$$

with block dimensions

$$8 + 8 + 14 + 20 + 8 + 8 + 12 = 78.$$

Table 5 summarizes the seven feature blocks.

B.2 Embedding-Based Features

Base drift ($\mathbf{v}^{(1)}$). For every prompt and seed, we compute the cosine distance between the fine-tuned model and the base checkpoint:

$$d_{\text{base}}(p, s) = 1 - \langle E_{p,s}, E^0_{p,s} \rangle. \tag{13}$$

We calculate the mean and standard deviation separately over the in-domain, near-domain, out-of-domain, and combined prompt sets, producing eight features.

Table 5: Composition of the 78-dimensional behavioral fingerprint. Embedding-space distances use cosine distance. Statistics are aggregated over the controlled probe suite and inference seeds.

Block	Signal	Construction	Dim.
$\mathbf{v}^{(1)}$	Base drift	Seed-matched cosine distance between fine-tuned and base-model generations; mean and standard deviation over four aggregation domains.	8
$\mathbf{v}^{(2)}$	Output diversity	Mean pairwise cosine distance among generations of the same prompt; mean and standard deviation over four aggregation domains.	8
$\mathbf{v}^{(3)}$	Perturbation sensitivity	Seed-matched cosine distance between perturbed prompts and their parents; mean and standard deviation over four perturbation groups and, separately, three prompt domains.	14
$\mathbf{v}^{(4)}$	Texture	Laplacian variance, high-frequency FFT energy ratio, mean gradient magnitude, and histogram entropy; global mean and standard deviation, plus domain-specific means.	20
$\mathbf{v}^{(5)}$	Text–image alignment	CLIP cosine similarity between each generated image and its prompt; mean and standard deviation over four aggregation domains.	8
$\mathbf{v}^{(6)}$	Cross-prompt consistency	Cosine distance between prompt-level representations and their corresponding domain centroids; mean and standard deviation over four aggregation domains.	8
$\mathbf{v}^{(7)}$	Color	Mean RGB channels, saturation, luminance mean, and luminance standard deviation; global mean and standard deviation of each statistic.	12
		Total	**78**

Output diversity ($\mathbf{v}^{(2)}$). For each prompt p, we measure output diversity using the average pairwise cosine distance among its S generations:

$$d_{\mathrm{div}}(p) = \frac{2}{S(S-1)} \sum_{s<s'} \left(1 - \langle E_{p,s}, E_{p,s'} \rangle\right). \tag{14}$$

The mean and standard deviation of d_{div} over each of the four aggregation domains yield eight features.

Prompt-perturbation sensitivity ($\mathbf{v}^{(3)}$). Each perturbed prompt is associated with an unmodified parent prompt. For a child–parent pair (p_c, p_r), we calculate the seed-matched cosine distance

$$d_{\mathrm{pert}}(p_c, p_r, s) = 1 - \langle E_{p_c,s}, E_{p_r,s} \rangle. \tag{15}$$

We aggregate these distances using their mean and standard deviation over four perturbation groups: synonym, shuffle, typo, and all perturbations. This produces eight features. We additionally calculate the mean and standard deviation within the in-domain, near-domain, and out-of-domain prompt groups, contributing six more features. The resulting block contains 14 dimensions.

Text–image alignment ($\mathbf{v}^{(5)}$). For each generated image $x_{p,s}$ and its corresponding prompt p, we compute

$$a(p,s) = \cos\Big(E^{\mathrm{img}}_{\mathrm{CLIP}}(x_{p,s}), E^{\mathrm{text}}_{\mathrm{CLIP}}(p)\Big). \tag{16}$$

We calculate the mean and standard deviation over each of the four aggregation domains, producing eight features.

This block uses CLIP image and text encoders regardless of the visual encoder selected for the other embedding-based blocks.

Cross-prompt consistency ($\mathbf{v}^{(6)}$). For each prompt, we average its ℓ_2-normalized image embeddings over inference seeds to obtain a prompt-level representation. Within each aggregation domain,

we compute the centroid of these representations. We then calculate the cosine distance between each prompt-level representation and its corresponding domain centroid.

The mean and standard deviation of these distances over each of the four aggregation domains produce eight features.

B.3 Pixel-Space Features

Two fingerprint blocks are computed directly from generated pixels and require no learned visual encoder. Texture features $\mathbf{v}^{(4)}$ contribute 20 dimensions, while color features $\mathbf{v}^{(7)}$ contribute 12 dimensions.

Texture ($\mathbf{v}^{(4)}$). Let H and W denote image height and width, respectively, and let R, G, and B denote the red, green, and blue pixel intensities on the 0–255 scale. We compute grayscale luminance as

$$Y = 0.299R + 0.587G + 0.114B. \tag{17}$$

For every generated image, we compute four statistics:

- **Laplacian variance:** The variance of the four-neighbor discrete Laplacian of Y.
- **High-frequency energy ratio:** The fraction of two-dimensional Fourier power lying outside a centered disk of radius $\min(H, W)/8$.
- **Gradient magnitude:** The image-wide mean of $\sqrt{g_x^2 + g_y^2}$, where g_x and g_y are spatial luminance gradients.
- **Histogram entropy:** The Shannon entropy, in bits, of the 256-bin grayscale histogram.

For each statistic, we retain its global mean and standard deviation, yielding eight dimensions. We additionally retain its mean within each of the three prompt domains, contributing another 12 dimensions. Thus,

$$\mathbf{v}^{(4)} \in \mathbb{R}^{20}.$$

Color ($\mathbf{v}^{(7)}$). Each generated image contributes six statistics: mean red, mean green, mean blue, saturation, luminance mean, and luminance standard deviation.

We calculate saturation per pixel as

$$\text{sat} = \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B)}. \tag{18}$$

For pixels with a zero-valued denominator, we define saturation as zero. We then average saturation over the image.

Luminance uses the same definition of Y introduced above. The mean and standard deviation of each of the six image-level statistics across the probe suite produce

$$\mathbf{v}^{(7)} \in \mathbb{R}^{12}.$$

B.4 Visual Encoders

We construct the embedding-based fingerprint components using frozen visual encoders. No encoder is fine-tuned on recipe labels or victim outputs.

We independently instantiate the fingerprinting pipeline using DINOv2, CLIP, and DeiT representations. The same encoder is used for training-shadow and held-out victim fingerprints within each experiment.

Text–image alignment always uses CLIP, regardless of the visual encoder used for the remaining embedding-based blocks.

The pixel-space blocks $\mathbf{v}^{(4)}$ and $\mathbf{v}^{(7)}$ remain identical across encoder choices. Consequently, 32 of the 78 fingerprint dimensions are independent of the selected learned visual representation.

C Additional Evaluation Details

C.1 Predictor Configurations

We evaluate five predictors on the same 78-dimensional behavioral fingerprints.

The random forest uses 300 trees with a fixed random seed of 42. XGBoost uses 200 estimators, a maximum tree depth of 3, a learning rate of 0.1, and histogram-based tree construction. Logistic regression uses a regularization parameter of $C = 1.0$ and a maximum of 2,000 optimization iterations.

We additionally evaluate two multi-head neural networks. MLP-Shared uses a shared representation with separate prediction heads for the six hyperparameters. MLP-Split uses decoupled per-head branches and PCGrad to address conflicting gradients during multi-task optimization.

The three conventional classifiers are trained separately for each hyperparameter. The neural predictors are trained jointly across hyperparameter heads.

C.2 Cross-Validation and Baselines

All predictors use the same model-level cross-validation protocol: three stratified folds repeated ten times, with stratification by LoRA rank. Feature standardization is estimated using the training portion of each fold and then applied to the corresponding held-out models.

The uniform baseline samples each candidate hyperparameter value with equal probability. The majority-class baseline predicts the most frequent class in each training fold. The stratified baseline samples predictions according to the training-fold class distribution. Both empirical baselines are evaluated using the same repeated cross-validation splits as the predictors.

For each repeat, we aggregate predictions across its three held-out folds to calculate accuracy. We report the mean and standard deviation across the ten repetitions.

C.3 Joint Recovery Evaluation

We evaluate joint recovery using six separate fixed 300-tree random forests. Their out-of-fold predictions are aligned by model and repetition across identical cross-validation splits.

For model i, let λ_i denote its true recipe and $\hat{\lambda}_i$ its predicted recipe. Exact joint recovery is defined as

$$A_{\text{joint}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\hat{\lambda}_i = \lambda_i\right], \tag{19}$$

where N is the number of held-out predictions in one evaluation repetition. We report the mean and standard deviation across ten repetitions.

The recipe grid contains five three-class hyperparameters and one binary hyperparameter. Under independent, uniform guessing, the probability of exactly recovering the complete recipe is

$$P_{\text{chance}} = \frac{1}{3^5 \times 2} = \frac{1}{486} \approx 0.21\%. \tag{20}$$

This theoretical probability serves as the reference for the chance-lift values reported in the main paper.

C.4 Permutation Testing

We perform label-permutation tests to examine whether recovery accuracy exceeds the level expected when the relationship between fingerprints and hyperparameter labels is disrupted.

For each predictor and hyperparameter, we randomly permute the corresponding labels across models and repeat the three-fold evaluation. The observed and permuted statistics are pooled out-of-fold accuracies from a single three-fold pass, rather than the ten-repeat estimates reported in the main results table.

The empirical permutation probability is computed as

$$p = \frac{1 + \sum_{b=1}^{B} \mathbb{I}(T_b \geq T_{\text{obs}})}{B + 1}, \tag{21}$$

Table 6: Hyperparameter recovery accuracy (%) on Stable Diffusion v2.1 victims. Aware uses the victim's actual base checkpoint, whereas blind uses the original shadow-model base checkpoint. Bold indicates the highest accuracy per row within each evaluation setting. An asterisk denotes significance after Benjamini–Hochberg correction across the six hyperparameters within each evaluation setting.

Hyperparameter	Chance	Aware			Blind		
		DINOv2	CLIP	DeiT	DINOv2	CLIP	DeiT
LoRA rank	33.3	**78.3***	75.9*	75.4*	60.1*	**74.6***	71.6*
LoRA scaling	33.3	55.8*	54.5*	**58.7***	58.0*	55.8*	**58.8***
Learning rate	33.3	**42.0***	40.6	36.2	**34.8**	31.2	30.4
Batch size	33.3	**36.2**	**36.2**	35.5	34.8	**35.5**	34.8
Training steps	33.3	**43.5***	39.9*	36.4	**42.8***	**42.8***	35.9
LoRA dropout	50.0	**52.6**	51.3	50.7	**52.9**	50.0	49.3

Table 7: Hyperparameter recovery accuracy (%) under changes in base-model weights and fine-tuning data. Bold indicates the highest accuracy per row within each evaluation setting. An asterisk denotes significance after Benjamini–Hochberg correction across the six hyperparameters within each evaluation setting.

Hyperparameter	Chance	Different Base Weights			Different Training Data		
		DINOv2	CLIP	DeiT	DINOv2	CLIP	DeiT
LoRA rank	33.3	**70.8***	69.4*	69.4*	**78.3***	75.8*	77.2*
LoRA scaling	33.3	45.6*	37.2	**45.8**	55.3*	**58.3***	51.4
Learning rate	33.3	**33.3**	30.6	26.4	**50.0***	40.3	40.3
Batch size	33.3	41.7	42.2	**44.4***	**40.0**	38.3	35.8
Training steps	33.3	**44.4***	37.5	40.3	40.0	**43.1**	40.3*
LoRA dropout	50.0	50.0	51.4	**52.2**	48.6	**58.3**	50.3

where $B = 1000$, T_{obs} denotes the observed accuracy, and T_b denotes the accuracy obtained from permutation b.

We apply Benjamini–Hochberg correction across the six hyperparameter heads separately for each predictor and visual encoder. Because predictor selection is not included within the permutation procedure, these adjusted q-values do not constitute a selection-adjusted significance test for the best-of-five results.

For the DINOv2 dropout experiment, the adjusted q-values are 0.101 for RF, 0.025 for XGBoost, 0.043 for logistic regression, 0.395 for MLP-Shared, and 0.102 for MLP-Split. This variation motivates reporting both the individual predictor results and the fixed-RF reference.

C.5 Detailed Results Under Model and Data Shifts

We report the complete hyperparameter recovery results for the transfer experiments described in Section 5.5. All predictors are trained on the original 297-model shadow collection. The additional victim models are excluded from attacker training.

Cross-Architecture Transfer. Table 6 reports recovery on 138 Stable Diffusion v2.1 victims. The architecture-aware setting uses the victim's base checkpoint when computing the base-model reference features. The architecture-blind setting retains the attacker's Stable Diffusion v1.5 reference. Both settings use the same trained predictors.

Base-Checkpoint and Data Transfer. Table 7 reports results from two additional victim collections, each containing 72 models. The first replaces the original base checkpoint with Realistic Vision v5.1 while preserving the model architecture. The second retains the original base checkpoint but replaces the fine-tuning corpus with 833 Naruto images.

The probing protocol and attacker training collection remain unchanged in both experiments.

Table 8: Recovery under disjoint same-domain training data using nine matched recipes ($\alpha = 16$). Accuracy is reported in percent; retention is chance-adjusted. Permutation p-values refer to disjoint-corpus recovery.

Hyperparameter	Encoder	Same	Disjoint	Retention	p
LoRA rank	DINOv2	100.0	100.0	100%	0.0007
	CLIP	100.0	91.1	87%	0.0023
	DeiT	100.0	100.0	100%	0.0007
Learning rate	DINOv2	70.4	44.4	30%	0.341
	CLIP	74.1	53.3	49%	0.341
	DeiT	70.4	53.3	54%	0.341

Table 9: LoRA rank recovery using individual behavioral feature blocks. The complete 78-dimensional fingerprint provides the reference accuracy. All results use DINOv2-based fingerprints.

Feature configuration	**Dim.**	**Accuracy (%)**
Complete fingerprint	78	85.4
Base-model drift	8	79.1
Output diversity	8	82.6
Prompt sensitivity	14	79.6
Texture	20	82.8
Text–image alignment	8	81.7
Cross-prompt consistency	8	78.9
Color	12	**86.2**

Discussion. LoRA rank is the only hyperparameter that remains statistically recoverable across every transfer setting and visual encoder. LoRA scaling also retains measurable information under several shifts, whereas recovery of learning rate, batch size, and training steps is less consistent.

The architecture-blind results further show that the effect of using an incorrect base-model reference depends on the visual encoder. Rank recovery decreases substantially with DINOv2 but remains comparatively stable with CLIP and DeiT.

These results establish that the observed transfer is strongest for LoRA rank. They do not imply that all components of the training recipe generalize equally across model architectures, pretrained weights, and fine-tuning datasets.

C.6 Generalization to Disjoint Training Data

We evaluate generalization to disjoint same-domain training images using nine matched recipes, with LoRA scaling fixed at $\alpha = 16$. The two corpora are screened for overlap using perceptual hashes and DINOv2 embeddings.

C.7 Detailed Fingerprint Ablation

We evaluate the contribution of each behavioral fingerprint block using DINOv2 representations and the fixed 300-tree random forest. All experiments follow the primary model-level evaluation protocol.

Individual Feature Blocks. Table 9 reports LoRA rank recovery using each feature block independently.

Leave-One-Out Ablation. Removing individual blocks from the complete fingerprint yields rank accuracies between 84.0% and 86.3%, compared with 85.4% using all seven blocks. These results suggest substantial predictive redundancy among the fingerprint components.

C.8 Detailed Adapter-Drift Controls

We evaluate whether behavioral recovery can be explained by the magnitude of the learned LoRA adapter. The controls use the adapter statistics $\|A\|$, $\|B\|$, and $\|A\|\|B\|$, which are unavailable to the black-box attacker.

Table 10: Controlling for adapter drift. Accuracy is reported for LoRA rank and scaling using DINOv2-based fingerprints. Drift-only uses adapter norms, which are unavailable to the black-box attacker. Stratified results correspond to three equal-sized groups ($n = 99$ each) formed according to $\|B\|$.

Configuration	Rank (%)	Scaling (%)
Behavioral fingerprint	85.4	71.9
Adapter norms only	99.9	46.8
Stratified: low drift	85.7	64.9
Stratified: medium drift	78.8	72.3
Stratified: high drift	88.7	86.4
Drift-residualized features	82.7	70.1

Table 11: Hyperparameter recovery accuracy (%) as the number of images per prompt varies. All configurations use the same 96 prompts and DINOv2-based fingerprints.

	Images per prompt			
Hyperparameter	**1**	**2**	**4**	**8**
LoRA rank	84.0	85.0	86.1	85.4
LoRA scaling	66.6	70.2	70.9	72.8
Learning rate	54.3	54.7	55.3	55.7
Batch size	50.7	51.0	51.9	49.4
Training steps	56.5	55.9	56.0	55.9

Drift Stratification. We partition the 297 models into three equal-sized groups according to $\|B\|$. Behavioral rank recovery ranges from 78.8% to 88.7%, while scaling recovery ranges from 64.9% to 86.4%.

Drift Residualization. After removing the linear association with $\log \|B\|$, rank and scaling accuracies remain at 82.7% and 70.1%, respectively. These controls show that recovery is not fully explained by this measured linear association, although other relationships with adapter-update magnitude may remain.

D Additional Query-Efficiency and Inference-Condition Experiments

This appendix provides additional results for the query-efficiency and inference-condition experiments discussed in Section 5.9.

D.1 Images-per-Prompt Evaluation

We evaluate the effect of repeated sampling by fixing the 96 probing prompts and varying the number of images per prompt from one to eight. Reduced fingerprints are constructed from subsets of cached generations, keeping the underlying models and prompts unchanged.

Rank recovery remains relatively stable across sampling budgets, whereas LoRA scaling benefits more substantially from repeated sampling.

D.2 Prompt-Budget Evaluation

The primary probing protocol generates eight images for each of 96 prompts, requiring 768 images per model. To measure the effect of the number of prompts independently, we fix one image per prompt and evaluate prompt budgets of 8, 16, 24, 48, and 96.

Prompt subsets are sampled as complete groups consisting of one original prompt and its three perturbations. This preserves the parent–child relationships required by the prompt-sensitivity features. We use 20 independently sampled subsets for each budget below 96 prompts. The complete 96-prompt configuration is deterministic.

Table 12: Recovery accuracy (%) across prompt budgets. One image is generated per prompt. Results for budgets below 96 are reported as mean ± standard deviation over 20 prompt-subset replicates. The evaluation uses pooled model-level cross-validation and DINOv2 fingerprints.

Hyperparameter	**Number of Prompts**				
	8	**16**	**24**	**48**	**96**
LoRA rank	82.9 ± 1.1	82.9 ± 1.9	83.9 ± 1.5	84.3 ± 1.1	84.0
LoRA scaling	65.6 ± 3.3	67.3 ± 2.5	68.0 ± 1.7	68.0 ± 1.2	66.6
Learning rate	53.6 ± 1.8	53.4 ± 2.0	54.2 ± 2.0	54.1 ± 1.5	54.4
Batch size	49.4 ± 2.1	50.4 ± 1.9	51.1 ± 2.3	50.8 ± 1.6	50.7
Training steps	56.0 ± 2.2	55.8 ± 1.9	55.9 ± 1.6	56.3 ± 1.2	56.5
LoRA dropout	56.2 ± 2.3	56.6 ± 2.9	56.5 ± 2.4	56.7 ± 1.9	57.9

Table 13: LoRA rank recovery under inference-condition changes using DINOv2 fingerprints. The unshifted reference accuracy is 77.9%. Blind retains the original base-model inference configuration; matched uses the victim's inference configuration.

Victim configuration	**Blind (%)**	**Matched (%)**
Euler sampler	67.4	68.9
20 inference steps	71.9	71.1
50 inference steps	73.3	74.1
Guidance = 5.0	51.0	63.0
Guidance = 12.0	77.8	81.2

The same prompt subset is used for all models within each replicate. We evaluate the resulting DINOv2 fingerprints using the model-level cross-validation protocol and report the highest accuracy among the evaluated prediction architectures.

Table 12 shows relatively small differences across the evaluated prompt budgets. Rank accuracy increases from $82.9\% \pm 1.1\%$ using eight prompts to 84.0% using all 96 prompts.

Combining this experiment with the images-per-prompt evaluation in Table 11 indicates that a substantially smaller probing budget can preserve much of the observed recovery performance. The prompt-budget results are specific to DINOv2 and pooled cross-validation; they do not establish the same query requirements for independently held-out victims or other visual encoders.

At the eight-prompt operating point, only two complete prompt groups are available. Consequently, the full set of probing domains cannot be represented, and some domain-specific feature entries are absent. The reported accuracy therefore applies to this reduced feature configuration.

D.3 Cross-Encoder Inference-Shift Evaluation

We additionally examine whether the inference-shift results depend on the visual encoder used to construct the behavioral fingerprint.

The experiment uses 135 independently held-out victim models. Each victim is queried under five alternative inference configurations, changing exactly one parameter relative to the reference configuration.

We evaluate DINOv2, CLIP, and DeiT fingerprints separately. In the blind setting, the attacker retains the original base-model inference configuration.

To compare encoders with different unshifted accuracies, we report the fraction of above-chance performance retained under each shift:

$$R = \frac{a_{\text{shift}} - a_{\text{chance}}}{a_{\text{base}} - a_{\text{chance}}}, \tag{22}$$

where a_{shift} is accuracy under the changed inference configuration, a_{base} is the corresponding unshifted accuracy, and a_{chance} is uniform-chance accuracy.

The unshifted rank accuracies are 77.9%, 81.5%, and 80.0% for DINOv2, CLIP, and DeiT, respectively.

Table 14: Fraction of above-chance LoRA rank performance retained under inference shifts. Results use the blind setting and 135 held-out victim models.

Configuration	DINOv2	CLIP	DeiT
Euler sampler	0.76	0.66	0.63
20 inference steps	0.86	0.68	0.73
50 inference steps	0.90	0.75	0.79
Guidance = 5.0	0.40	0.43	0.49
Guidance = 12.0	1.00	0.92	0.94

Table 15: Fraction of above-chance rank performance retained when the base-model reference matches the victim's inference settings.

Configuration	DINOv2	CLIP	DeiT
Euler sampler	0.80	0.66	0.62
20 inference steps	0.85	0.71	0.75
50 inference steps	0.91	0.77	0.81
Guidance = 5.0	0.66	0.43	0.51
Guidance = 12.0	1.07	1.03	1.01

Table 14 shows that the principal inference-shift finding is consistent across all three encoders. Changing the sampler or inference-step count preserves between 63% and 90% of the original above-chance rank performance.

Reducing classifier-free guidance to 5.0 produces the largest degradation for every encoder, retaining only 40–49% of the original signal. Increasing guidance to 12.0 instead retains 92–100%.

Thus, the asymmetric effect of classifier-free guidance is not specific to DINOv2.

D.4 Matched and Blind Reference Settings

We further compare the blind setting with a matched-reference setting in which the base-model generations use the victim's inference configuration.

Matching the base-model reference does not consistently eliminate the effects of inference shifts. In particular, under guidance 5.0, DINOv2 retention increases from 0.40 to 0.66, whereas CLIP remains at 0.43 and DeiT changes only from 0.49 to 0.51.

The larger blind-to-matched improvement observed with DINOv2 therefore does not generalize across encoders. The reduction in rank recovery under lowered guidance cannot be attributed solely to an incorrectly specified base-model reference.

D.5 Rank Prediction Errors Under Reduced Guidance

We examine rank prediction errors under classifier-free guidance values of 5.0 and 12.0 using DINOv2 fingerprints in the blind setting. Table 16 aggregates predictions from five seeded fits over the same 135 held-out victims. Each configuration uses its best-performing classifier, consistent with Table 13.

At guidance 5.0, the selected random forest never predicts rank 4 across any of the five fits. All 240 predictions for victims with true rank 4 are assigned to higher-rank classes, while 205 of the 225 predictions for true rank 64 are correct. Exact rank accuracy is 51.0%, whereas accuracy within one rank-grid step remains 96.3%, compared with a uniform-chance adjacent accuracy of 77.8%.

At guidance 12.0, the selected XGBoost classifier achieves 77.8% exact accuracy and 95.6% adjacent accuracy. Unlike the reduced-guidance setting, it produces predictions for all three rank classes.

These results show that substantial information about relative rank persists despite errors in exact identification. Because the two configurations use different selected classifiers, their confusion matrices do not isolate the effect of guidance alone.

Table 16: LoRA rank confusion matrices under two guidance settings. Rows indicate true ranks; columns indicate predicted ranks. Counts aggregate five fits over 135 victims. The selected classifiers are random forest (guidance 5.0) and XGBoost (guidance 12.0).

	Predicted rank		
True rank	4	16	64
Guidance = 5.0 (RF)			
4	0	215	25
16	0	139	71
64	0	20	205
Guidance = 12.0 (XGBoost)			
4	235	5	0
16	65	105	40
64	30	10	185

E Additional Evaluation of Deployment-Level Defenses

We provide additional experimental details and results for the output-perturbation evaluation discussed in Section 5.10.

E.1 Experimental Protocol

We evaluate four output transformations: Gaussian noise, JPEG compression, color quantization, and spatial downscaling. These transformations modify the generated images returned to the attacker without changing the fine-tuned model.

We use 162 shadow models for training and 135 independently held-out AdamW victim models for evaluation. The shadow-model generations and base-model reference images remain unmodified. Perturbations are applied only to the victim's generated images.

This setting evaluates whether output modification can disrupt a previously trained attack without requiring the attacker to retrain its classifier using similarly perturbed shadow-model outputs.

We consider the following perturbation configurations:

- **Gaussian noise:** Standard deviations of $\{0, 2, 4, 8, 16, 32\}/255$.
- **JPEG compression:** Compression settings corresponding to JPEG qualities down to 10.
- **Color quantization:** Removal of up to six bits per color channel.
- **Spatial downscaling:** Reduction of image resolution followed by upsampling to the original resolution.

We measure image degradation using peak signal-to-noise ratio (PSNR) and evaluate recovery using DINOv2-based behavioral fingerprints.

The unmodified victim images reproduce the original held-out transfer results: 77.9% for LoRA rank, 67.9% for LoRA scaling, and 56.3% for learning rate.

E.2 Recovery Under Output Perturbations

Table 17 reports recovery accuracy under the evaluated perturbation configurations. The table includes the unmodified baseline and the measured settings reported in our perturbation sweep.

The 19 perturbation settings reveal substantial differences in leakage suppression. Mild transformations provide little protection: Gaussian noise with $\sigma = 2$ yields 80.0% rank accuracy, compared with 77.9% for unmodified images. Even at $\sigma = 32$, rank accuracy remains 75.1%.

Spatial downscaling produces the largest observed reduction in rank accuracy, reaching 46.2% when images are reduced to one-eighth of their original width and height. Strong color quantization and JPEG compression reduce rank accuracy to 51.6% and 54.8%, respectively.

Table 17: Hyperparameter recovery accuracy (%) under output perturbations. PSNR measures image degradation relative to the original generated images. All results use DINOv2 fingerprints and 135 held-out victim models. An asterisk indicates significance after Benjamini–Hochberg correction; "ns" denotes a non-significant result.

Perturbation	PSNR	Rank	Scaling	LR	Steps
None	∞	77.9*	67.9*	56.3*	53.9*
Gaussian $\sigma = 2$	41.8	80.0*	67.7*	56.3*	57.8*
Gaussian $\sigma = 4$	36.1	77.2*	63.7*	52.1*	52.6*
Gaussian $\sigma = 8$	30.2	78.2*	61.6*	51.1*	50.4*
Gaussian $\sigma = 16$	24.3	78.1*	61.9*	49.6*	48.1*
Gaussian $\sigma = 32$	18.5	75.1*	51.4*	51.1*	45.9*
JPEG quality 90	37.9	79.3*	69.5*	53.8*	56.3*
JPEG quality 75	34.6	76.3*	65.8*	52.1*	53.6*
JPEG quality 50	32.3	76.3*	61.5*	50.5*	53.3*
JPEG quality 30	30.5	73.3*	63.7*	46.7*	49.6*
JPEG quality 10	26.6	54.8*	55.4*	43.7 ns	50.4*
Quantization -2 bits	46.3	77.0*	64.9*	47.6*	52.3*
Quantization -3 bits	40.6	73.6*	63.0*	49.8*	54.1*
Quantization -4 bits	34.7	63.7*	63.7*	49.0*	51.1*
Quantization -5 bits	28.7	61.5*	67.4*	48.0*	49.6*
Quantization -6 bits	22.7	51.6*	57.8*	42.8*	46.1*
Downscale $\times 0.75$	30.8	74.1*	57.8*	48.9*	52.6*
Downscale $\times 0.50$	27.5	70.4*	58.5*	48.9*	53.3*
Downscale $\times 0.25$	24.4	49.2*	55.6*	43.1*	51.1*
Downscale $\times 0.125$	22.3	46.2*	49.0*	44.0*	49.6*

Table 18: Image quality and rank-leakage retention at the strongest evaluated setting of each output transformation. These operating points have different PSNR values and should not be interpreted as an exactly quality-matched comparison.

Method	PSNR	Rank (%)	Retention
Gaussian	18.5	75.1	0.94
JPEG	26.6	54.8	0.48
Quantization	22.7	51.6	0.41
Downscaling	22.3	46.2	0.29

Nevertheless, rank remains statistically detectable across all 19 perturbation settings. Among the four hyperparameters reported, learning rate is the only one to lose statistical significance, occurring at JPEG quality 10.

These results characterize non-adaptive output perturbations; the attacker is not retrained using similarly perturbed shadow-model generations.

E.3 Image Quality and Leakage Retention

We compare the strongest setting of each transformation using the fraction of above-chance rank performance retained:

$$R = \frac{a_{\text{perturbed}} - a_{\text{chance}}}{a_{\text{original}} - a_{\text{chance}}}, \tag{23}$$

where $a_{\text{original}} = 0.779$ and $a_{\text{chance}} = 1/3$.

Table 18 shows that the strongest Gaussian perturbation retains 94% of the original above-chance rank performance, despite reducing PSNR to 18.5 dB.

By comparison, downscaling retains 29% at 22.3 dB. The two settings therefore differ substantially in both image quality and leakage suppression.

The complete perturbation curves also indicate that downscaling generally produces greater rank suppression than quantization, JPEG compression, or Gaussian noise at comparable image-quality levels.

However, even the strongest downscaling configuration leaves rank accuracy above the 33.3% uniform-chance baseline.

E.4 Scope and Limitations

These experiments evaluate non-adaptive output perturbations using one visual encoder and one held-out victim population. An attacker who anticipates the transformation could potentially train its shadow-model classifier using similarly perturbed images; we do not evaluate that setting.

Furthermore, PSNR measures pixel-level distortion and does not fully capture perceptual quality or downstream image utility. Consequently, the reported results characterize the measured leakage–distortion trade-off rather than establishing a universal deployment threshold.

Finally, we do not evaluate training-time privacy mechanisms such as differentially private fine-tuning. Whether such methods can suppress the observed fingerprints while preserving generation quality remains outside the scope of this study.