LoRAHunter: Decoding Semantic Representations from Low-Rank Adapter Weights

Anonymous authors
Paper under double-blind review

Abstract

The rapid growth of community-trained Low-Rank Adaptation (LoRA) adapters creates a need for scalable adapter understanding and retrieval, yet existing systems rely heavily on noisy metadata, example images, or expensive generation-based profiling. We investigate whether LoRA weights can support text-aligned adapter representations learned from metadata and offline generation feedback. We propose LoRAHunter, a structured parameter representation framework that queries low-rank weight updates with learnable probes, projects module-specific responses into a shared token space, and aggregates them into a weight-derived adapter embedding. We align this space with language using noisy metadata as weak supervision and further calibrate it for generation-oriented retrieval using offline rewards. At retrieval time, database-side adapter embeddings are computed from LoRA parameters and support semantic clustering, text-to-LoRA retrieval, and multi-adapter selection. Experiments on Stable Diffusion and Qwen-Image show that the learned parameter representations exhibit meaningful semantic structure, support text-to-LoRA retrieval, and provide a useful signal for generation-oriented adapter selection. These results support supervised weight-derived representations as a complementary signal for adapter analysis and selection. Code is available at `https://anonymous.4open.science/r/LoRAHunter-ECFC`.

1 Introduction

Low-Rank Adaptation (LoRA) (Hu et al., 2022) has become a standard mechanism for personalizing modern image generation models. With only modest amounts of data and computation, users can steer a base diffusion model toward specific objects, identities, and artistic styles (Ruiz et al., 2023; Guo et al., 2024). At the same time, open communities such as ModelScope and Civitai have turned LoRA into a reusable building block within a modular ecosystem. As the number of public adapters grows into the hundreds of thousands, users increasingly need to understand what a given adapter has learned before they retrieve, combine, or deploy it. Reliable adapter understanding is therefore central to retrieval, composition, reuse, and model management.

Existing approaches infer adapter behavior primarily from signals external to the adapter parameters. User-provided titles, descriptions, and tags are inexpensive to index, but they are often incomplete, noisy, or inconsistent across repositories (Sarfaty et al., 2025; Sonmezer et al., 2025). Training data and curated example images provide more direct evidence (Muqeeth et al., 2024a), yet they are frequently unavailable for community-trained adapters. Generation-based profiling can probe an adapter through its outputs (Sarfaty et al., 2025), but synthesizing images for every candidate is costly when constructing or updating a large retrieval index. These limitations raise a natural question: can we learn useful semantic representations of adapters from their weights?

Prior work on weight-space learning suggests that model weights themselves form a learnable and comparable data modality (Han et al., 2026; Navon et al., 2023). We observe that A LoRA is not an arbitrary collection of tensors: it is a set of low-rank perturbations inserted into canonical modules of a shared base model. These perturbations directly modify the computation of the diffusion model, and any learned concept, style, or behavior must ultimately be expressed through their effects on the model's internal transformations. Because adapters for the same base model share a common backbone and module layout, their updates are defined in a partially aligned coordinate system. Adapters trained for related goals may therefore induce similar perturbation patterns across

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.

corresponding layers, leaving a recoverable functional footprint in weight space. This does not imply exact semantic invertibility, but it suggests that LoRA weight matrices contain semantic and functional signals that can be exposed through learned representations.

However, learning semantic representations from LoRA weights is non-trivial. LoRA parameters are heterogeneous low-rank matrices rather than fixed-length vectors; different layers have different shapes and roles, and community-trained adapters may modify different subsets of modules with different ranks. Moreover, parameter similarity does not automatically align with natural language, and textual relevance is not always a reliable proxy for generation quality. These challenges call for structured parameter encoding, language alignment, and generation-aware calibration within a common embedding space.

We address these challenges with **LoRAHunter**, a framework for learning semantic representations directly from LoRA parameters. Its Parameter Probing Network applies learnable probes to module-wise low-rank updates, and aggregates the ordered module tokens with a Transformer into a global adapter embedding. We first align this embedding space with language through contrastive learning, using adapter metadata as weak supervision. In the second stage, inspired by GCL (Zhu et al., 2025), preference-aware refinement uses offline rewards from generated images to calibrate prompt–adapter similarities for generation-oriented selection. The learned embedding space supports both retrieval and composition: text prompts can retrieve semantically aligned and preference-aware LoRAs, and candidate subsets can be scored by concept coverage, global prompt relevance, and redundancy penalties to balance semantic relevance and diversity. Empirically, we find that the learned weight-derived embeddings form meaningful semantic clusters, support text-to-LoRA retrieval against a strong metadata-text baseline, and remain useful when adapter metadata is degraded. In downstream composition, the same representation provides a practical signal for selecting adapters that are semantically relevant and empirically helpful for generation.

In summary, our main contributions are as follows:

- We formulate LoRA understanding as a parameter representation learning problem and study whether low-rank adapter weights contain recoverable semantic signals.
- We introduce LoRAHunter, a structured weight encoder that combines learnable parameter probing and Transformer aggregation, and learns a text-aligned embedding space through weak semantic supervision and preference-aware refinement.
- Experiments show semantic structure and text-to-LoRA retrieval in the learned weight embeddings, and evaluate their use for generation-oriented multi-adapter selection with separately trained Stable Diffusion and Qwen-Image encoders.

2 Related Work

Weight-Space Representations. Task embeddings and weight-space learning together suggest that model behavior can be summarized geometrically. Task2Vec (Achille et al., 2019) represents a task through Fisher information, and FUTE (Wang et al., 2024) extends task embeddings to a shared space across models and prompts. In parallel, weight-space studies show that model properties can be inferred directly from parameters: Classifying the Classifier (Eilertsen et al., 2020) and Predicting Neural Network Accuracy from Weights (Unterthiner et al., 2020) predict network quality from weights, Self-Supervised Representation Learning on Neural Network Weights for Model Characteristic Prediction (Schürholt et al., 2021) learns compact hyper-representations, and symmetry-aware encoders such as Equivariant Architectures for Learning in Deep Weight Spaces (Navon et al., 2023) and Permutation Equivariant Neural Functionals (Zhou et al., 2023) respect the structure of parameter space. Recent works further argue that weights are a learnable data modality and that useful representations can be learned from heterogeneous open collections (Han et al., 2026; Falk et al., 2025). These results support the premise behind LoRAHunter, but they mostly target full checkpoints or curated model zoos; our setting is different in that we encode LoRA updates directly and align them with natural-language semantics for retrieval, clustering, and composition.

Adapter Retrieval. Large adapter libraries have motivated methods that automatically route inputs to suitable adapters. MoE-style approaches such as SMEAR (Muqeeth et al., 2024b) and MoLE (Wu et al., 2024) learn routing mechanisms over a fixed set of experts. For LLMs, PHATGOOSE

(Muqeeth et al., 2024a) learns adapter signatures from training data, whereas RAMoLE (Zhao et al., 2024) instruction-tunes a sentence encoder to match input text with task-specific LoRAs. These approaches are effective when expert sets or adapter-side supervision are available, but are less suited to continually expanding community repositories with incomplete auxiliary information. Adapter retrieval has also been studied for vision models. SemLA (Qorbani et al., 2025) retrieves segmentation adapters using similarities to their training data. Stylus (Luo et al., 2024) summarizes diffusion adapters with enriched descriptions and embeddings, then retrieves and composes them according to prompt keywords. LoRAverse (Sonmezer et al., 2025) clusters adapters using metadata embeddings and applies submodular optimization to select subsets that balance relevance and diversity. In contrast, LoRAHunter uses metadata only as weak supervision during training and constructs database-side representations directly from LoRA parameters for retrieval and downstream composition.

3 METHOD

3.1 PROBLEM FORMULATION

Let $\mathcal{D} = \{a_i\}_{i=1}^{N}$ denote a repository of LoRA adapters associated with a fixed diffusion backbone, and let $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$ be its canonical target-module set. Adapter a_i modifies a subset $\mathcal{L}_i \subseteq \mathcal{L}$. For module $\ell \in \mathcal{L}_i$, the adapter stores factors $A_{i,\ell} \in \mathbb{R}^{r_{i,\ell} \times d_{\text{in},\ell}}$ and $B_{i,\ell} \in \mathbb{R}^{d_{\text{out},\ell} \times r_{i,\ell}}$, which induce the effective update $\Delta W_{i,\ell} = B_{i,\ell} A_{i,\ell}$.

Our goal is to learn a weight encoder f_θ that maps the LoRA parameters to a semantic embedding $z_i^w = f_\theta(a_i) \in \mathbb{R}^d$. Training uses adapter metadata and offline preference rewards as supervision. At retrieval time, however, the indexed representation z_i^w is computed from the adapter parameters without adapter-side metadata or image rendering. Let g denote the text-query encoder. Given query q, we compute $z_q^t = g(q)$ and rank adapters by normalized cosine similarity $s(q, a_i) = \langle \bar{z}_q^t, \bar{z}_i^w \rangle$.

3.2 PARAMETER PROBING NETWORK

A LoRA adapter comprises heterogeneous, module-specific low-rank updates that are poorly represented by direct flattening. As shown in Figure 1, we instead probe each update as a linear operator and aggregate the resulting module tokens in canonical order.

Learnable weight probing. For each canonical module ℓ, we maintain P trainable probes $X_\ell \in \mathbb{R}^{P \times d_{\text{in},\ell}}$. For an adapter that contains module ℓ, the probes query its low-rank update without explicitly materializing a flattened $d_{\text{out},\ell} d_{\text{in},\ell}$-dimensional vector:

$$R_{i,\ell} = X_\ell \Delta W_{i,\ell}^\top = X_\ell A_{i,\ell}^\top B_{i,\ell}^\top, \qquad R_{i,\ell} \in \mathbb{R}^{P \times d_{\text{out},\ell}}. \tag{1}$$

Their responses provide a compact, learnable sketch of the update as a linear operator. We model interactions among the P responses with a residual MLP applied along the probe dimension:

$$\widehat{R}_{i,\ell} = R_{i,\ell}^\top + \text{MLP}_{\text{probe}}\big(\text{LN}(R_{i,\ell}^\top)\big), \qquad \widehat{R}_{i,\ell} \in \mathbb{R}^{d_{\text{out},\ell} \times P}. \tag{2}$$

A trainable linear map $\Pi_P : \mathbb{R}^P \to \mathbb{R}^T$ then compresses the probe axis into T response tokens: $V_{i,\ell} = \Pi_P(\widehat{R}_{i,\ell})^\top \in \mathbb{R}^{T \times d_{\text{out},\ell}}$. Here, T is fixed across modules, while the last dimension still reflects the output structure of module ℓ.

Type-specific projection. The response tokens produced by different modules have different output dimensions and architectural roles. For each module type $c = \text{type}(\ell)$, we therefore use a trainable projection Φ_c:$E_{i,\ell} = \Phi_{\text{type}(\ell)}(V_{i,\ell}) \in \mathbb{R}^{T \times D}$. They normalize heterogeneous module responses into a shared hidden space before semantic supervision is applied. Modules assigned to the same type share a projection, whereas modules of different types use separate parameters.

Cross-module aggregation. We concatenate module tokens in the canonical order $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$, prepend a learnable class token, and add positional embeddings:

$$Z_i^{(0)} = \big[[\text{CLS}]; E_{i,\ell_1}; \ldots; E_{i,\ell_M}\big] + P_{\text{pos}}, \tag{3}$$

where $Z_i^{(0)} \in \mathbb{R}^{(MT+1) \times D}$. A K-block Transformer contextualizes responses across modules, $Z_i^{(K)} = \text{Transformer}_\theta(Z_i^{(0)})$, and the final class token is mapped to the adapter embedding: $z_i^w = W_o \,\text{LN}\Big(Z_{i,\text{cls}}^{(K)}\Big) \in \mathbb{R}^d$. The complete encoder f_θ includes the probes, probe MLP, response projection, type-specific projections, Transformer blocks, and output head.

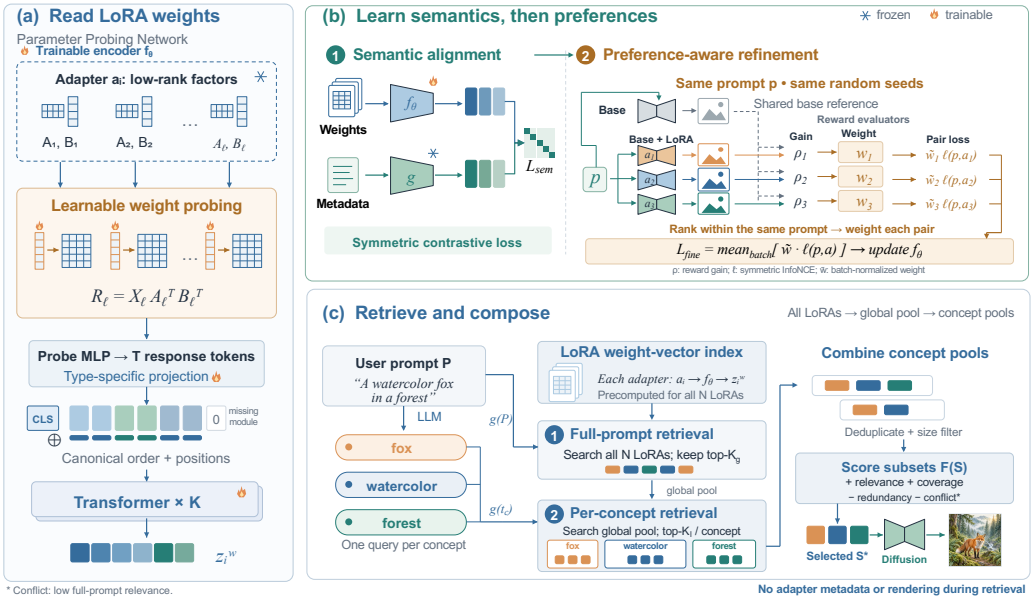

Figure 1: **Overview of LoRAHunter. (a)** Learnable probes and type-specific projections convert module-wise LoRA updates into tokens, which a Transformer aggregates into a weight embedding. **(b)** Metadata aligns weight embeddings with text; Reward-based weights then refine same-prompt adapter pairs while the text encoder remains frozen. **(c)** Full-prompt and concept queries search the weight index, and merged candidates are scored for relevance, coverage, and redundancy.

3.3 Weakly Supervised Semantic Alignment

Although community-provided metadata is noisy, it provides a useful semantic prior for aligning the weight-derived embedding space with language. Given a batch $\mathcal{B}$ of adapters and their corresponding metadata texts, we compute the adapter embeddings and metadata text embeddings as:

$$z_i^w = f_\theta(a_i), \qquad z_i^m = g(m_i), \qquad a_i \in \mathcal{B}, \tag{4}$$

where m_i denotes the metadata text (e.g., title, description, and tags) of adapter a_i and $g(\cdot)$ is a frozen text encoder such as CLIP. The parameter encoder f_θ is then optimized with a symmetric contrastive loss :

$$\mathcal{L}_{sem} = \frac{1}{2}\left[\mathrm{CE}(\mathrm{sim}(Z^w, Z^t)/\tau, y) + \mathrm{CE}(\mathrm{sim}(Z^t, Z^w)/\tau, y)\right], \tag{5}$$

where $Z^w = [z_1^w, \ldots, z_B^w]^\top$, $Z^t = [z_1^t, \ldots, z_B^t]^\top$, $\mathrm{CE}(\cdot)$ is cross-entropy loss and τ is a learnable temperature parameter. This stage organizes the weight-derived space around textual semantics and provides the initialization for both general retrieval and the subsequent preference refinement.

3.4 Preference-Aware Refinement

The Stage 1 representation already supports semantic analysis and retrieval. For generation-oriented selection, however, two semantically relevant adapters can differ in how much they improve the rendered result. Inspired by generalized contrastive learning Zhu et al. (2025), We therefore use an additional preference stage to calibrate prompt–adapter similarities with offline generation feedback.

Preference Dataset Construction. We construct a preference dataset consisting of weighted prompt-adapter pairs $(p_u, a_{u,v}, w_{u,v})$. For each prompt p_u, we first retrieve a candidate pool $\mathcal{C}_u = \{a_{u,v}\}_{v=1}^{M}$ using the semantically aligned retriever, where each $a_{u,v} \in \mathcal{D}$ is a candidate adapter. We then generate images with both the base model and each candidate adapter under the same random seeds. For each offline evaluator M_e such as CLIPScore Radford et al. (2021) or PickScore Kirstain et al. (2023), we compute the average score improvement over the base model:

$$\Delta m_{u,v}^{(e)} = \mathbb{E}_{s \in \mathcal{S}_{\mathrm{seed}}}\left[M_e(I_{a_{u,v}}^{(s)}, p_u) - M_e(I_{\mathrm{base}}^{(s)}, p_u)\right]. \tag{6}$$

The evaluator-specific improvements are fused into a scalar preference reward:

$$\rho_{u,v} = \sum_e \eta_e \tanh\left(\Delta m_{u,v}^{(e)} / \tau_e\right), \tag{7}$$

where η_e controls the contribution of evaluator M_e, and τ_e normalizes the scale of its score differences. Within each prompt-specific candidate pool, we convert the fused rewards into loss weights according to their intra-pool ranking:

$$w_{u,v} = \begin{cases} 1/\sqrt{\mathrm{rank}_{u,v}}, & \rho_{u,v} > 0, \\ \epsilon, & \rho_{u,v} \le 0, \end{cases} \tag{8}$$

where $\mathrm{rank}_{u,v}$ is the descending rank of $\rho_{u,v}$ within $\mathcal{C}_u$, and ϵ is a small positive weight assigned to candidates that do not improve over the base model.

Reward-weighted contrastive learning. Given a batch of weighted prompt-adapter pairs $\mathcal{B} = \{(p_k, a_k, w_k)\}_{k=1}^{|\mathcal{B}|}$, we compute adapter embeddings $z_k^w = f_\theta(a_k)$ and frozen text embeddings $z_k^t = g(p_k)$. Both embeddings are ℓ_2-normalized before computing cosine similarities. The parameter encoder is fine-tuned with a reward-weighted symmetric InfoNCE loss:

$$\mathcal{L}_{\text{fine}} = -\frac{1}{2|\mathcal{B}|} \sum_{k=1}^{|\mathcal{B}|} \tilde{w}_k \left(\log \frac{\exp((z_k^w)^\top z_k^t / \tau)}{\sum_{m=1}^{|\mathcal{B}|} \exp((z_k^w)^\top z_m^t / \tau)} + \log \frac{\exp((z_k^t)^\top z_k^w / \tau)}{\sum_{m=1}^{|\mathcal{B}|} \exp((z_k^t)^\top z_m^w / \tau)} \right), \tag{9}$$

where τ is a learnable temperature parameter and $\tilde{w}_k = w_k / (\frac{1}{|\mathcal{B}|} \sum_{m=1}^{|\mathcal{B}|} w_m)$is the batch-normalized reward weight. The reward weights modulate the contribution of each prompt-adapter pair to the contrastive objective. Pairs with larger reward improvements exert stronger alignment pressure, while non-improving candidates contribute weakly. This biases the learned retrieval space toward adapters that are both semantically relevant and empirically helpful for generation.

3.5 Adapter Retrieval and Composition

Concept Extraction and Constrained Local Retrieval. We first use an LLM to decompose a prompt P into semantic concepts $\mathcal{T}(P) = \{t_c\}_{c=1}^C$. The complete prompt is scored against all adapters to obtain the global pool $\mathcal{A}_g(P) = \mathrm{Top}_{K_g}\{\widetilde{s}(P, a_i) : a_i \in \mathcal{D}\}$. For each concept, we first retrieve an expanded full-index shortlist $\mathcal{B}_c(P) = \mathrm{Top}_{K_l}\{\widetilde{s}(t_c, a_i) : a_i \in \mathcal{D}\}$, and filter it by membership in the full-prompt pool. Let $\mathcal{H}_c(P)$ denote this filtered list in descending concept-score order. In our setting, $K_g = 150$ and $K_l = 10$. The local pool is

$$\mathcal{A}_c(P) = \begin{cases} \mathrm{First}_{K_l}(\mathcal{H}_c(P)), & |\mathcal{H}_c(P)| \ge K_l, \\ \mathrm{Top}_{K_l}\{\widetilde{s}(t_c, a_i) : a_i \in \mathcal{A}_g(P)\}, & |\mathcal{H}_c(P)| < K_l. \end{cases} \tag{10}$$

The fallback reranks the entire global pool and therefore guarantees up to K_l full-prompt-constrained candidates even when the expanded shortlist contains too few global-pool members.

Subset construction and scoring. We form the Cartesian product of the concept-specific pools $\mathcal{X}(P) = \mathcal{A}_1(P) \times \cdots \times \mathcal{A}_C(P)$, where a tuple $(a_{j_1}, \ldots, a_{j_C})$ assigns one candidate to each concept. We collapse every tuple into the unique adapter subset $\mathcal{S} = \mathrm{unique}(a_{j_1}, \ldots, a_{j_C})$, retain subsets satisfying $s_{\min} \le |\mathcal{S}| \le s_{\max}$, and remove duplicate subsets produced by different assignments. We use $s_{\min} = 1$, $s_{\max} = 4$, and abort enumeration when the Cartesian product exceeds 200,000 tuples.

After candidate generation, we score each unique subset independently of the tuple that produced it. The score combines full-prompt relevance, concept coverage, pairwise redundancy, and a low-context-relevance penalty. These are embedding-based selection criteria rather than direct estimates of functional interference between adapters::

$$\mathcal{F}(\mathcal{S}; P, \mathcal{T}) = \lambda_{\text{rel}} \Phi_{\text{rel}}(\mathcal{S}, P) + \lambda_{\text{cov}} \Phi_{\text{cov}}(\mathcal{S}, \mathcal{T}) - \lambda_{\text{red}} \Omega_{\text{red}}(\mathcal{S}) - \lambda_{\text{ctx}} \Omega_{\text{ctx}}(\mathcal{S}, P). \tag{11}$$

The full-prompt term balances the strongest selected adapter with the average relevance of the subset: $\Phi_{\text{rel}}(\mathcal{S}, P) = \frac{1}{2} \max_{a_i \in \mathcal{S}} \widetilde{s}(P, a_i) + \frac{1}{2|\mathcal{S}|} \sum_{a_i \in \mathcal{S}} \widetilde{s}(P, a_i)$, whereas concept coverage allows a

different selected adapter to explain each extracted concept: $\Phi_{\text{cov}}(\mathcal{S}, \mathcal{T}) = \sum_{c=1}^{C} \max_{a_i \in \mathcal{S}} \widetilde{s}(t_c, a_i)$. The redundancy term discourages selecting adapters that occupy nearly identical regions of the learned space: $\Omega_{\text{red}}(\mathcal{S}) = \frac{2}{|\mathcal{S}|(|\mathcal{S}|-1)} \sum_{\substack{a_i, a_j \in \mathcal{S} \\ i<j}} \langle \bar{z}_i^w, \bar{z}_j^w \rangle$, with $\Omega_{\text{red}}(\mathcal{S}) = 0$ for $|\mathcal{S}| = 1$. We also penalize selected adapters whose normalized relevance to the complete prompt falls below τ_{full}: $\Omega_{\text{ctx}}(\mathcal{S}, P) = \frac{1}{|\mathcal{S}|} \sum_{a_i \in \mathcal{S}} \mathbf{1}[\widetilde{s}(P, a_i) < \tau_{\text{full}}]$.

We first retain the top M subsets according to $\mathcal{F}$. To return multiple composition candidates without near duplicates, we define the normalized mean embedding of a subset as $\bar{z}_{\mathcal{S}} = \text{norm}\Big(\frac{1}{|\mathcal{S}|} \sum_{a_i \in \mathcal{S}} \bar{z}_i^w\Big)$ and the similarity between two subsets as $\text{Sim}(\mathcal{S}, \mathcal{S}') = \beta_{\text{set}} \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|} + \beta_{\text{emb}} \langle \bar{z}_{\mathcal{S}}, \bar{z}_{\mathcal{S}'} \rangle$. The first returned subset maximizes $\mathcal{F}$. Each subsequent subset is selected by maximal marginal relevance,

$$\mathcal{S}^{(r)} = \arg \max_{\mathcal{S} \in \mathcal{Q} \setminus \mathcal{R}_{r-1}} \left[\lambda_{\text{MMR}} \mathcal{F}(\mathcal{S}) - (1 - \lambda_{\text{MMR}}) \max_{\mathcal{S}' \in \mathcal{R}_{r-1}} \text{Sim}(\mathcal{S}, \mathcal{S}') \right], \tag{12}$$

where $\mathcal{Q}$ is the top-M quality pool and $\mathcal{R}_{r-1}$ contains the previously returned subsets.

4 Experiments

4.1 Experimental Setup

Data. For Stable Diffusion (SD) (Rombach et al., 2022), we use StylusDocs-v2 (Luo et al., 2024), retaining 56K valid adapters and using 51K for representation training. For Qwen-Image, we retain approximately 10K community adapters and use 8K for training. The LoRA encoder is trained separately on their respective LoRA corpora. For semantic analysis, we use a diagnostic set of 206 SD adapters from 12 categories, excluded from both training stages. Text-to-LoRA retrieval uses 120 manually written category queries under different candidate-pool settings. Generation experiments use 500 prompts from COCO (Lin et al., 2014) and PartiPrompts (Esser et al., 2024), with no overlap with preference-training prompts. Metadata-degradation evaluation uses 250 COCO prompts.

Baselines. For semantic analysis and retrieval, we compare against CLIP Metadata Text, which encodes each adapter's title, description, and tags with the same frozen text encoder used by LoRAHunter. Because some evaluation labels are derived from metadata, this baseline also shares information with the labeling process. For generation, we compare with the base model, Stylus (Luo et al., 2024), and LoRAverse (Sonmezer et al., 2025).

Metrics. We evaluate semantic structure using ARI, NMI, purity, kNN@5, cosine Silhouette, and the intra–inter class similarity gap. Retrieval is measured by Hit@K, Recall@K, MRR, and nDCG@10. For generation, we report CLIPScore (Radford et al., 2021), PickScore (Kirstain et al., 2023), and ImageReward (Xu et al., 2023), together with TCE Ibarrola & Grace (2024), TIE Ibarrola & Grace (2024), and image-to-image similarity (I2I).

4.2 Semantic Structure of LoRA Weight Embeddings

We first test whether the final weight encoder, trained with metadata supervision and preference refinement, recovers semantic organization from unseen adapters. We evaluate on the 206-adapter, 12-class diagnostic set, which is excluded from both training stages. We apply K-means with $K = 12$ and evaluate both clustering structure and local neighborhoods.

Table 1: Semantic-structure evaluation on the 12-class LoRA set.

Method	ARI	NMI	Purity	kNN@5	Sil.	I–I Gap
CLIP (Metadata)	0.799	0.882	0.876	**0.848**	0.205	0.201
LoRAHunter (Weights)	**0.810**	**0.884**	**0.893**	0.797	**0.229**	**0.347**

Table 1 shows that LoRAHunter achieves slightly higher ARI and NMI than CLIP (Metadata), while improving purity, cosine Silhouette, and the intra–inter similarity gap. The gap increases from 0.201 to 0.347, whereas kNN@5 decreases from 0.848 to 0.797. The weight-derived space therefore exhibits stronger class separation under these measures, while clean metadata provides more accurate local neighborhoods. The similarity heatmap (Figure 3) shows clear category separation while preserving meaningful cross-category affinities, such as watercolor–oil painting and

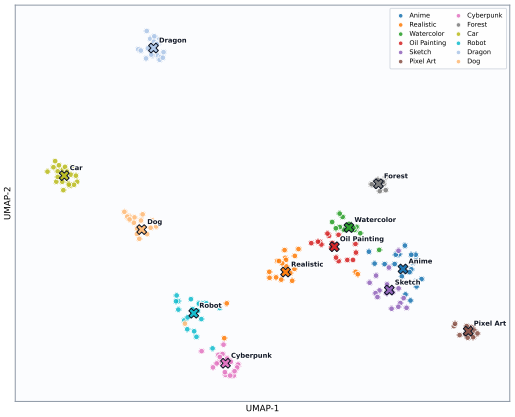

Figure 2: UMAP projection of the encoded LoRA representations. Adapters from the same category form compact and separable clusters.

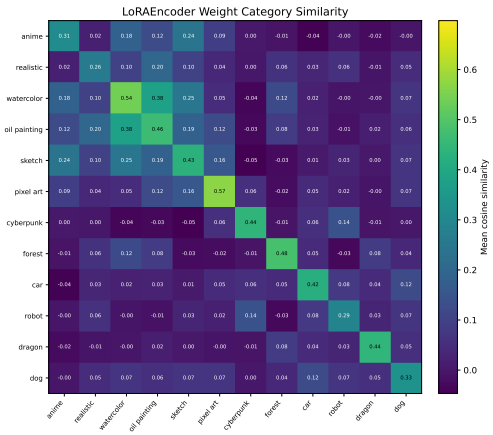

Figure 3: Pairwise embedding similarity heatmap. The diagonal indicates strong intra-class cohesion.

Table 2: Text-to-LoRA retrieval results under two settings.

Setting	Method	Hit@1	Hit@5	Hit@10	Recall@10	Recall@20	MRR	nDCG@10
Main	LoRAHunter	**0.567**	0.742	**0.967**	**0.270**	**0.362**	0.639	0.412
	CLIP	0.558	**0.787**	0.900	0.264	0.327	**0.667**	**0.424**
Stress	LoRAHunter	**0.525**	0.683	0.783	0.245	0.353	**0.634**	0.336
	CLIP	0.425	**0.780**	**0.875**	**0.284**	**0.389**	0.565	**0.355**

cyberpunk–robot. This indicates that the learned space is not merely orthogonalizing category labels, but also retains semantic relationships between related adapter types. UMAP (Figure 2) likewise reveals compact category clusters. Taken together, these results suggests that LoRA parameters do indeed encode rich, structured semantic information. By interacting with these weights, LoRAHunter effectively decodes this implicit knowledge, establishing a well-structured metric space that provides a foundational tool for downstream tasks such as LoRA retrieval and combination. We further compare LoRAHunter with a baseline that directly flattens the LoRA matrices (Appendix A.2.2), and LoRAHunter achieves better performance.

4.3 Text-to-LoRA Retrieval

We next test whether natural-language queries can retrieve relevant adapters from the weight-derived index. We focus on category-level queries with multiple relevant adapters and therefore better reflect practical text-to-LoRA search. We evaluate the same 120 category queries in two settings. Main uses a candidate pool of 1,093 adapters, while Stress expands the pool to 1,959 adapters with additional distractors. Both methods use the same queries, candidate pool, and relevance labels.

The results in Table 2 show that LoRAHunter can consistently retrieve relevant adapters from the top-ranked candidate and maintains stable retrieval capabilities even in a more challenging candidate pool. CLIP Metadata Text typically performs better on Hit@5 and ranking list metrics, which is expected since it directly indexes textual metadata. Therefore, we think LoRAHunter as a parameter-based retrieval signal, particularly suitable for situations where metadata is noisy, incomplete, or unavailable, rather than a general alternative to metadata retrieval.

4.4 Preference-Aware Retrieval and Composition

We evaluate multi-adapter selection on 500 COCO and PartiPrompts prompts that do not overlap with preference-training prompts. Each method produces 10 images per prompt. Adapter-selection methods share the candidate pool and concept decomposition, and all methods use matched seeds

and the same generation settings within each backbone. For LoRAHunter, the 10 images correspond to 10 returned adapter subsets, with one image generated per subset (see Appendix A.1 for details).

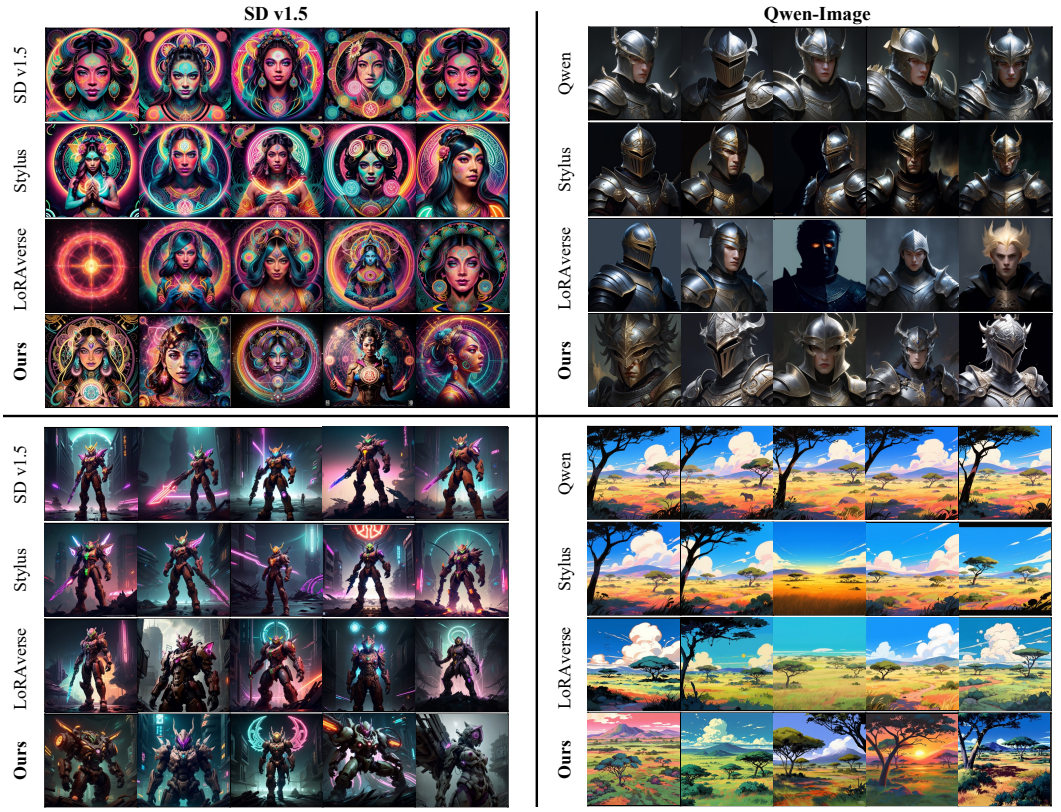

Figure 4: **Qualitative Comparison.** Each row shows multiple outputs for the same prompt. The prompts are in appendix A.1. LoRAHunter maintains diversity while showcasing better image quality.

Comparison of generation results. On SD, LoRAHunter improves image-quality metrics over the baselines (Table 3), while LoRAverse shows a slight advantage in diversity. One possible explanation is that selecting adapters from different semantic groups encourages broader visual variation, but does not necessarily ensure that the selected combination satisfies the full prompt. LoRAHunter explicitly incorporates full-prompt relevance alongside concept coverage and embedding redundancy, which may help balance these objectives; Relative to the base model, ImageReward increases from 0.5095 to 0.5413, while I2I similarity decreases from 0.855 to 0.825. On Qwen-Image, LoRAHunter improves both image-quality and diversity metrics. This result suggests that the parameter-based retrieval pipeline can transfer beyond SD. It also provides additional evidence that LoRA parameters trained on a different diffusion architecture still contain recoverable semantic information. We provide further VLM-Judge results in the Appendix A.2.1. Figure 4 shows some visual examples. In the Appendix A.3, we present more detailed case studies on concept decomposition and LoRA retrieval.

Preference-aware refinement ablation. To validate the impact of our Preference-Aware Refinement, we conduct an ablation study in Table 4. Preference-aware refinement improves CLIP, PickScore, and ImageReward over semantic alignment alone, where ImageReward does not participate in the reward construction process. The result indicates that reward-weighted alignment can bias retrieval toward adapters that are empirically more helpful for generation. It also suggests that textual relevance and final generation quality are not necessarily equivalent; retrieving and analyzing LoRAs only through text metadata is therefore insufficient for generation-oriented adapter selection.

Table 4: Ablation of preference-aware refinement on the SD v1.5 generation benchmark.

Method	CLIP ↑	PickScore ↑	IR ↑
w/o Preference Distillation	26.053	22.470	0.511
w/ Preference Distillation	**26.416**	**22.614**	**0.541**

Table 3: Quantitative multi-adapter generation evaluation on Stable Diffusion v1.5 and Qwen-Image. We report diversity metrics and image-quality metrics.

Backbone	Method	Generation Diversity			Image Quality		
		TCE ↑	TIE ↑	I2I ↓	CLIP ↑	PickScore ↑	IR ↑
SD v1.5	SD v1.5	2.399	1.155	0.855	26.286	22.503	0.509
	Stylus	2.432	1.198	0.847	25.964	22.399	0.380
	LoRAverse	**2.492**	**1.215**	**0.813**	25.767	22.267	0.316
	LoRAHunter	2.434	1.202	0.825	**26.416**	**22.614**	**0.541**
Qwen-Image	Qwen-Image	2.025	1.543	0.912	27.881	23.649	1.432
	Stylus	2.059	1.561	0.884	28.012	23.619	1.391
	LoRAverse	2.060	1.567	0.882	27.501	23.584	1.408
	LoRAHunter	**2.063**	**1.581**	**0.866**	**28.276**	**23.697**	**1.499**

Robustness to Metadata Degradation. Open LoRA repositories often contain incomplete or noisy metadata, such as missing descriptions, sparse tags, or uninformative titles. To test whether retrieval and composition remain reliable under this realistic failure mode, we construct a metadata-degradation benchmark. For adapters in the candidate pool, we randomly remove the description field with 50% probability and perturb the title field with 50% probability by replacing it with irrelevant distractor terms or meaningless identifiers. We then evaluate downstream generation on 250 COCO prompts.

As shown in Table 5, metadata degradation noticeably affects methods that rely on textual adapter descriptions. Stylus shows a clear drop in CLIP, PickScore, and ImageReward under the perturbed setting, while LoRAverse also degrades on most metrics. These results indicate that retrieval pipelines based primarily on external metadata can be sensitive to incomplete or corrupted adapter descriptions. LoRAHunter remains more stable because its adapter-side representations are computed from LoRA weights rather than from the corrupted metadata fields. Under the same perturbed setting, LoRAHunter achieves the best results among the perturbed-metadata methods on all reported metrics. These results support the role of weight-derived adapter embeddings as a metadata-robust retrieval signal for generation-oriented composition. In the Appendix A.2.3, we also analyze how prompt complexity affects LoRA retrieval.

Table 5: Robustness evaluation. LoRAHunter does not use adapter metadata at retrieval time; its original and perturbed conditions therefore coincide, and only one row is reported.

Method	Metadata	TCE ↑	TIE ↑	I2I ↓	CLIP ↑	PickScore ↑	IR ↑
Stylus	Original	1.676	0.979	0.853	25.427	22.275	0.264
	Perturbed	1.684	0.947	0.859	25.145	22.081	0.175
LoRAverse	Original	1.713	0.958	0.831	25.694	22.351	0.260
	Perturbed	1.701	0.934	0.827	25.481	22.241	0.253
LoRAHunter	Perturbed	**1.733**	**0.968**	**0.826**	**25.873**	**22.485**	**0.371**

5 Conclusion

We introduced LoRAHunter, a structured parameter encoder that transforms module-wise LoRA updates into language-queryable embeddings. Through metadata-supervised semantic alignment and preference-aware refinement using CLIPScore and PickScore, LoRAHunter learns representations whose database-side inputs are derived solely from adapter weights. Experiments show that these representations exhibit meaningful semantic structure, support competitive text-to-LoRA retrieval, and provide a useful signal for multi-adapter selection. Preference refinement further improves generation-oriented selection, including on ImageReward, which is excluded from reward construction and model selection. Applying the same design and training procedure to separately trained Stable Diffusion and Qwen-Image encoders also demonstrates its applicability across diffusion architectures. These results establish LoRA parameters as a useful source of semantic information for adapter analysis, retrieval, and composition.

AI USE STATEMENT

We used generative AI tools for language polishing, translation, and assistance with drafting and improving the clarity and organization of the manuscript. The core research work, including the development of the research ideas and methodology, experimental work, and interpretation of the results, was conducted by the authors. The authors take responsibility for the final manuscript and all reported methods, results, and conclusions.

ETHICS STATEMENT

This work studies representations and retrieval of community-trained LoRA adapters for image generation. Such adapters may inherit biases, sensitive content, and intellectual-property concerns from their training data and underlying models. Our retrieval method does not verify the provenance, licensing status, or safety of an adapter, and relevance or preference scores should not be interpreted as evidence of responsible use. Deployments should incorporate appropriate license checks, content safeguards, and privacy protections. More effective adapter retrieval may also facilitate the discovery of adapters that enable harmful or unauthorized image generation; this risk should be considered when deploying the system or distributing adapter collections.

REPRODUCIBILITY STATEMENT

Section 3 describes the weight encoder, training objectives, and adapter selection procedure. Appendix A.1 provides implementation details, including corpus preprocessing, encoder configurations, optimization settings, preference-data construction, prompt decomposition, and generation protocols. Section 4 and Appendix A.2 report the evaluation settings and ablation studies. Together, these descriptions document the main components and experimental procedures needed to reproduce the reported results.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 6429–6438. IEEE, 2019.

Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: Dissecting the weight space of neural networks. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1119–1126. IOS Press, 2020.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Damian Falk, Konstantin Schürholt, Konstantinos Tzevelekakis, Léo Meynent, and Damian Borth. Learning model representations using publicly available model hubs. *CoRR*, abs/2510.02096, 2025.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, Rose Yu, Haggai Maron, Yanfang Ye, Lu Yin, and Ferrante Neri. A survey of weight space learning: Understanding, representation, and generation. *CoRR*, abs/2603.10090, 2026.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Francisco Ibarrola and Kazjon Grace. Measuring diversity in co-creative image generation. In Kazjon Grace, Maria Teresa Llano, Pedro Martins, and Maria M. Hedblom (eds.), *Proceedings of the 15th International Conference on Computational Creativity, ICCC 2024, Jönköping, Sweden, June 17-21, 2024*, pp. 254–261. Association for Computational Creativity (ACC), 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014.

Michael Luo, Justin Wong, Brandon Trabucco, Yanping Huang, Joseph E. Gonzalez, Zhifeng Chen, Ruslan Salakhutdinov, and Ion Stoica. Stylus: Automatic adapter selection for diffusion models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *Trans. Mach. Learn. Res.*, 2024, 2024b.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25790–25816. PMLR, 2023.

Reza Qorbani, Gianluca Villani, Theodoros Panagiotakopoulos, Marc Botet Colomer, Linus Härenstam-Nielsen, Mattia Segù, Pier Luigi Dovesi, Jussi Karlgren, Daniel Cremers, Federico Tombari, and Matteo Poggi. Semantic library adaptation: Lora retrieval and fusion for open-vocabulary semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 9804–9815. Computer Vision Foundation / IEEE, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 10674–10685. IEEE, 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 22500–22510. IEEE, 2023.

Shahar Sarfaty, Adi Haviv, Uri Hacohen, Niva Elkin-Koren, Roi Livni, and Amit H. Bermano. Carlos: Retrieval via concise assessment representation of loras at scale. *CoRR*, abs/2512.08826, 2025.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 16481–16493, 2021.

Mert Sonmezer, Matthew Zheng, and Pinar Yanardag. Loraverse: A submodular framework to retrieve diverse adapters for diffusion models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 17879–17888. IEEE, 2025.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya O. Tolstikhin. Predicting neural network accuracy from weights. *CoRR*, abs/2002.11448, 2020.

Xinyu Wang, Hainiu Xu, Lin Gui, and Yulan He. Towards unified task embeddings across multiple models: Bridging the gap for prompt-based large language models and beyond. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 8324–8340. Association for Computational Linguistics, 2024.

Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 893–911. Association for Computational Linguistics, 2023.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Yuwei Hu, Tao Shen, Hongxia Yang, Kun Kuang, and Fei Wu. Retrieval-augmented mixture of lora experts for uploadable machine learning. *CoRR*, abs/2406.16989, 2024.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Tianyu Zhu, Myong Chol Jung, and Jesse Clark. Generalized contrastive learning for multi-modal retrieval and ranking. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov (eds.), *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pp. 661–670. ACM, 2025.

A Appendix

A.1 Implementation Details

A.1.1 LoRA corpus and preprocessing.

For Stable Diffusion (SD), we retain approximately 56K valid adapters from StylusDocs-v2 and use 51K for representation training. For Qwen-Image, we retain approximately 10K community adapters and use 8K for training. The two backbones use the same encoder design and training objectives, but their encoders are trained separately on the corresponding LoRA corpora. The 206-adapter, 12-category SD diagnostic set is excluded from both training stages. Adapter metadata is used for Stage 1 supervision and metadata-based baselines; the indexed LoRAHunter representation is computed from adapter parameters.

For each backbone, we parse the low-rank factors of its target modules in a fixed canonical order. Missing target modules are represented by zero-valued placeholder tokens at their canonical positions, yielding a fixed-length token sequence for that backbone. The SD and Qwen encoders use backbone-specific module lists; the exact lists and preprocessing rules are provided in Table 6.

A.1.2 Weight encoder architecture.

The Stable Diffusion lora encoder uses learnable probing to convert each module-wise low-rank update into local response tokens. In our implementation, each target module uses $P = 48$ learnable probes and produces $T = 2$ local tokens. Local tokens are projected to a common hidden dimension $D = 768$ using type-specific projection heads, where module types share projection parameters according to their architectural role and output dimensionality. The module tokens are concatenated in canonical order, prepended with a learnable class token, and augmented with positional embeddings.

Table 6: Weight-encoder configurations. The Qwen encoder reads LoRA updates from all 60 backbone Transformer blocks. Parameter counts exclude the frozen text encoder.

Setting	SD v1.5	Qwen-Image
Module families	Attention/FFN	Attention
Projection types	69	8
Canonical positions M	150	480
Probes P	48	48
Tokens per module T	2	2
Hidden dimension D	768	768
Encoder layers K	24	24
Trainable parameters	$\sim$320M	$\sim$650M

The SD encoder reads 23 target-module patterns in each of three down blocks and three up blocks, plus 12 mid-block patterns, giving 150 canonical positions. The Qwen-Image encoder reads eight attention-projection patterns from all 60 transformer blocks (indices $0, \ldots, 59$), giving 480 positions. With two tokens per position, the corresponding Transformer input sequences contain 301 and 961 tokens, respectively, including the class token. The Qwen-Image MLP targets shown in the implementation comments are not active in this configuration.

A.1.3 Semantic initialization and preference refinement.

Training stages. Stage 1 aligns LoRA-weight embeddings with embeddings of each adapter's title, description, and tags using symmetric contrastive learning. The CLIP ViT-L/14 text encoder is frozen; only the weight encoder is optimized. Stage 2 starts from this checkpoint and refines the weight encoder with offline prompt–adapter preference feedback. Unless explicitly labeled otherwise, the clustering, retrieval, and generation results use Stage 2 weight-encoder checkpoints.

Preference refinement. For each preference prompt, we retrieve 10 candidate LoRAs and compare each LoRA-conditioned image with a base-model image generated under the same prompt and seed. We compute reward improvements using CLIPScore and PickScore, fuse them as defined in the main text, and convert within-prompt reward ranks into prompt–adapter loss weights. Stage 2

Table 7: Stage 2 preference data. Both backbones use the same 20K prompts; candidates are retrieved only from their respective Stage 1 training adapters. Candidates per prompt denotes the initial retrieval count.

Field	SD v1.5	Qwen-Image
Distinct training prompts	20K	20K
Distinct candidate LoRAs used	∼20K	∼5.5K
Initial candidates per prompt	10	10
Shared seeds per comparison	5	5
Valid prompt–adapter pairs	∼150K	∼120K

Table 8: Optimization settings shared by SD and Qwen-Image in both stages.

Setting	Value
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay (encoder)	1×10^{-4}
Batch size per device	64
Global batch size	64×4
Training epochs	200
LR schedule	Cosine
Warmup	0.5% of training steps
Training precision	bf16

fine-tunes only the weight encoder; the CLIP text encoder remains frozen. ImageReward is reserved for evaluation and is not used to train the encoder or select its checkpoint or hyperparameters.

Preference reward settings. We use CLIPScore and PickScore with equal raw weights of 1.0, normalized to 0.5 each. Both reward scales are 0.03, giving

$$\rho_{u,v} = 0.5 \tanh\left(\frac{\Delta m_{u,v}^{(\mathrm{CLIP})}}{0.03}\right) + 0.5 \tanh\left(\frac{\Delta m_{u,v}^{(\mathrm{Pick})}}{0.03}\right).$$

For non-positive rewards, we retain the prompt–adapter pair with a small positive weight of $\epsilon = 0.05$; positive-reward pairs receive the rank-based weight defined in the main text.

Preference-data construction. We sample a shared set of 20K prompts from COCO-30K (Lin et al., 2014) and DiffusionDB (Wang et al., 2023) for the SD and Qwen-Image preference datasets. For each prompt and backbone, the Stage 1 retriever initially selects 10 candidate LoRAs exclusively from that backbone's Stage 1 training adapters; held-out test adapters are excluded. Across the resulting preference records, approximately 20K distinct SD LoRAs and 5.5K distinct Qwen-Image LoRAs appear. We generate a base-model image and an image with each candidate LoRA under the same five random seeds, and compute prompt-specific improvements from the paired outputs. Approximately 120K valid prompt–adapter pairs per backbone are retained for Stage 2 training after data filtering. The 500 prompts used for the final generation evaluation do not overlap with the 20K preference prompts. Table 7 summarizes the two datasets.

Optimization and checkpoint selection. Both stages train the LoRA weight encoder while keeping the CLIP ViT-L/14 text encoder frozen. Stage 1 uses symmetric metadata–weight contrastive learning; Stage 2 initializes from Stage 1 and uses the reward-weighted objective in the main text. We use the run-specific settings in Table 8. ImageReward is not used for reward construction, checkpoint selection, or hyperparameter selection.

A.1.4 LLM-ASSISTED PREPROCESSING AND BASELINES

We use the same fixed LLM (Qwen-3.6) configuration for LoRAHunter's prompt decomposition and for the LLM calls required by Stylus. LoRAHunter sends only the extracted `keyword` strings as

concept-level retrieval queries; the returned `explanation` fields are not used for scoring. Stylus additionally uses the LLM to select adapters from its prompt-specific candidate catalogue. LoRAHunter's indexed adapter embeddings and retrieval scores use LoRA parameters rather than adapter-side text.

For concept extraction, we use the following system prompt and user prompt template, where `[PROMPT]` is replaced by the input text-to-image prompt.

System prompt for concept extraction

You are a specialist to extract meaningful keywords and provide explanations for prompts designed for text-to-image models. Your task is to identify keywords or phrases that significantly impact the visual content generated by such models.

User prompt template for concept extraction

You are a keyword extractor specialized in identifying concepts and their descriptive adjectives from prompts used in text-to-image models. Your task is to detect key concepts, i.e., keywords with their adjectives, that significantly influence the content, style, or composition of the generated image. Ensure that the extracted concepts are relevant to the visual elements of the scene and describe distinct entities or attributes. Do not split related concepts that describe a single entity. Avoid extracting verbs describing actions.

Key Instructions:

1) Extract descriptive concepts. Always extract keywords that include any relevant descriptive adjectives or qualifiers that modify them in the prompt. If adjectives or modifiers are provided, ensure they are part of the concept, as they provide crucial detail to the entity. For example, for "a large, ancient stone castle," return "large, ancient stone castle" rather than just "castle." For "a vibrant painting of a tropical sunset," return "vibrant painting of a tropical sunset."
2) Focus on impactful concepts. Keywords should describe specific entities, concepts, styles, or attributes central to the generated image. Avoid overly general terms, such as "thing" or "place." Ignore verbs describing an action.
3) Combine related concepts. If multiple keywords collectively describe a single entity, treat them as one unified concept.
5) The extracted concept should not be a verb phrase or a phrase describing the positional relationship between different objects. For example, "standing together side by side" is not allowed because it describes the action "standing" and the positional relationship between objects.
6) Provide explanations. For each keyword, provide a concise explanation of why it was chosen.
7) A prompt can extract a maximum of four concepts. If there are more than four concepts, retain the most critical ones, as these concepts play a key role in the image content.
8) Return the response as a JSON list. Each element must contain an `explanation` field and a `keyword` field.

Extract and explain keywords for the prompt enclosed within the literal tags `<prompt>` and `</prompt>`:

```
<prompt>
[PROMPT]
</prompt>
```

Provide the response in the JSON format specified above without Markdown code fences.

The extracted concepts are then used as local text queries in constrained multi-adapter retrieval.

A.1.5 Retrieval and composition configuration

At inference, we search the precomputed Stage 2 weight-embedding index using cosine similarity. The full prompt retrieves a global top-150 pool. For each extracted concept, we retrieve a top-100 shortlist from the full index, retain members of the global pool, and keep up to 10 candidates. If fewer than 10 survive, we instead rank the global pool by the concept score and take its top 10. Per-query dense scores are min–max normalized before subset scoring.

We enumerate one adapter choice per concept, collapse each tuple to a unique subset, and keep subsets of size 1–4. Enumeration aborts if the Cartesian product exceeds 200,000 tuples. We score each distinct subset using the relevance, coverage, redundancy, and context terms defined in the main text, with coefficients $(0.50, 0.50, 0.10, 0.05)$ and a low-full-prompt threshold of 0.30. We retain the top 200 scored subsets and apply MMR only when returning multiple results, using $\lambda = 0.70$,

Table 9: Evaluation-set and label provenance.

Setting	Queries/adapters	Pool	Label source
Semantic diagnostic	206 adapters	206	Manual annotation
Main retrieval	120 queries	1,093	Manual annotation and LLM
Stress retrieval	same 120 queries	1,959	Manual annotation and LLM

Table 10: Exact prompts used in the qualitative comparison in Figure 4.

Backbone	Example	Prompt
SD v1.5	Top	`Portrait of a transcendent being symbolizing global consciousness, in the vibrant styles, featuring intricate multicultural motifs and cosmic elements blending into a unified whole`
SD v1.5	Bottom	`A hyper-detailed sci-fi Kshatriya warrior in a stylized fusion of Cytus and Deemo aesthetics, glowing with mysterious energy, set within the dystopian yet beautiful world of Half-Life 2...`
Qwen-Image	Top	`Avalon knight, close-up portrait, fierce, intricate, elegant, volumetric lighting, scenery, digital painting, highly detailed, artstation, sharp focus, illustration, concept art, ruan jia, steve mccurry`
Qwen-Image	Bottom	`beautiful serengeti landscape, illustration, concept art, pixiv, art by wlop and greg rutkowski and makoto shinkai and studio ghibli and kyoto animation, vivid, colorful, cel - shaded, blackoutlines`

overlap weight 0.30, and embedding-similarity weight 0.70. At most 10 subsets are returned. The same settings are used for SD and Qwen-Image.

Image generation protocol. All image generation experiments are implemented with the `diffusers` framework. For Stable Diffusion image generation, we use an SD v1.5-compatible checkpoint and generate images at 512×512 resolution with a DPMSolver++ scheduler. Unless otherwise stated, we use 35 denoising steps and classifier-free guidance scale 7.0. For Qwen-Image generation, we generate images at 1024×1024 resolution with 20 denoising steps. The same generation protocol is applied to all methods compared under the same backbone.

For each retrieved combination, we load the selected LoRA weights into the diffusion pipeline and activate the corresponding adapters. Adapter weights follow a fixed heuristic used throughout the experiments: a single adapter uses weight 0.8; two adapters use weights 0.5 and 0.5; three adapters use 0.33, 0.33, and 0.33; and four adapters use uniform weight 0.25. This fixed weighting rule avoids introducing an additional learned tuning stage and keeps the comparison focused on retrieval and composition.

Evaluation labels and query sets. The 206-adapter SD diagnostic set contains 12 semantic categories and is excluded from both training stages. Its category labels were manually annotated. Our final text-to-LoRA benchmark contains 120 manually written category queries. The same queries are used in Main and Stress, with candidate pools of 1,093 and 1,959 adapters. For each query, the relevant-adapter labels were manually annotated or use LLM. Both LoRAHunter and CLIP (Metadata) are evaluated with the same query text, candidate pool, and positive sets. Database-side retrieval for LoRAHunter uses only Stage 2 weight embeddings and frozen text-query embeddings.

Prompts for the qualitative examples. Table 10 lists the exact prompts shown in Figure 4. Each group compares outputs for one fixed prompt across methods. For LoRAHunter, the displayed outputs correspond to different returned adapter subsets rather than repeated seeds for a single subset.

A.2 Ablation and Additional Analysis

A.2.1 VLM-Based Pairwise Preference Evaluation

To complement metric-based generation evaluation, we conduct an auxiliary pairwise preference study on Qwen-Image. We use the 200 prompts from diffusionDB and generate one image per prompt for each method with the same shared random seed. The evaluator is the Qwen2.5-VL. For each comparison, the two images are shown with randomized, method-blind labels, and the evaluator selects the image with better visual quality and the image it prefers overall. Each method pair is evaluated on 200 prompt-level comparisons.

Table 11: VLM-based pairwise voting on 200 Qwen-Image prompts. Each entry reports win rates as *Ours / Opponent / Tie*.

Comparison	**Overall preference**
Ours vs. Stylus	72.5% / 27.5% / 0.0%
Ours vs. LoRAverse	70.5% / 28.5% / 1.0%
Ours vs. Base	69.0% / 29.5% / 1.5%

As shown in Table 11, LoRAHunter receives a clear pairwise advantage over both adapter-selection baselines. Against Stylus, it wins 86.5% of visual-quality comparisons and 72.5% of overall preference comparisons. Against LoRAverse, it wins 84.0% and 70.5%, respectively. These results suggest that the proposed retrieval and composition pipeline is more likely to produce visually preferred images than the two metadata-driven adapter-selection baselines under this VLM voting protocol.

A.2.2 Ablation: Is Flattening LoRA Weights Sufficient?

We compare LoRAHunter with Flat-Sketch MLP, a baseline that replaces structured module probing and Transformer aggregation with a fixed-length sketch and a two-layer MLP. For each canonical SD module, Flat-Sketch pads or truncates the LoRA rank to 32, flattens the two low-rank factors, and accumulates their coordinates into an 8192-dimensional deterministic bucketed sketch; its MLP outputs a 768-dimensional embedding. Flat-Sketch and the SD LoRAHunter encoder have approximately 400M and 320M trainable parameters, respectively. Both follow the Stage 1 semantic-alignment and Stage 2 preference-refinement objectives, and we evaluate their Stage 2 checkpoints.

Both encoders receive LoRA weights as input and are evaluated on the same 206 SD adapters from 12 categories; these adapters are excluded from both training stages. LoRAHunter and Flat-Sketch MLP have approximately 320M and 400M trainable parameters. We ℓ_2-normalize the Stage 2 embeddings and report ARI, NMI, kNN@5, cosine Silhouette, the intra–inter class similarity gap, and leave-one-out nearest-centroid accuracy. This comparison evaluates structured encoding against a flattened baseline of similar scale, although it does not isolate the contribution of any single encoder component.

Table 12: Flattened-weight ablation on the 12-class LoRA diagnostic set. Both methods compute adapter embeddings from LoRA weights and are evaluated on the same 206 adapters. Flat-Sketch MLP uses fixed-rank A/B flattening with deterministic bucketed sketching, while LoRA Weight denotes the structured LoRAHunter encoder.

Method	ARI	NMI	kNN@5	Silhouette	Gap	Centroid Acc.
Flat-Sketch MLP	0.240	0.428	0.470	0.034	0.097	0.801
LoRA Weight	**0.810**	**0.884**	**0.797**	**0.229**	**0.347**	**0.942**

Table 12 shows a large and consistent gap between the two encoders. The flattened baseline retains some signal: its nearest centroid accuracy reaches 0.801, suggesting that coarse category information is not completely lost when LoRA factors are treated as a single vector. However, its clustering quality is substantially weaker than the structured encoder, with ARI dropping from 0.810 to 0.240 and NMI dropping from 0.884 to 0.428. The same trend appears in neighborhood and geometry metrics. kNN@5 decreases from 0.797 to 0.470, while the intra-inter similarity gap decreases from 0.347 to

0.097. These results indicate that simple vectorization can expose limited semantic regularities in LoRA weights, but it does not provide a well-separated metric space.

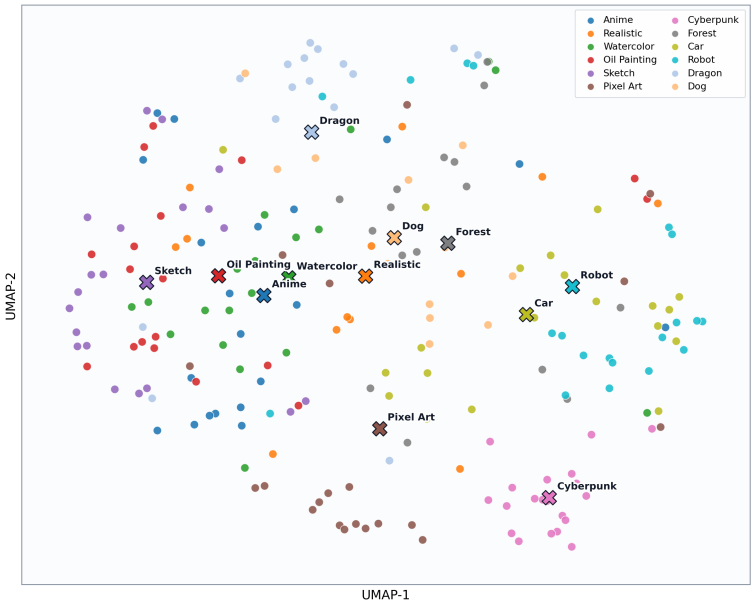

Figure 5: UMAP projection of the Flat-Sketch MLP embeddings on the 12-class diagnostic set.

Figure 5 visualizes the Flat-Sketch embeddings. Several categories form local neighborhoods, but the overall separation is weaker than the LoRAHunter visualization in Figure 2. Together with Table 12, this supports the utility of the complete structured encoder over the evaluated flattened baseline.

A.2.3 Performance across Varying Conceptual Complexities.

We further analyze how prompt complexity affects LoRA retrieval and composition. As shown in Table 13, the benefit of adapter selection is most visible for prompts with two or three concepts, where the base model often struggles to satisfy multiple requirements simultaneously. For single-concept prompts, the base model remains competitive, while for four or more concepts both methods become less stable. This suggests that composition is most useful in moderate-complexity prompts.

Table 13: Performance stratification across varying conceptual complexities.

# Concepts	Method	CLIP ↑	PickScore ↑	IR ↑
1	SD v1.5	**25.505**	22.16	**0.8190**
	Ours	25.491	**22.34**	0.8001
2	SD v1.5	26.342	22.65	0.5678
	Ours	**26.483**	**22.69**	**0.617**
3	SD v1.5	26.418	22.30	0.1580
	Ours	**26.602**	**22.32**	**0.2516**
4+	SD v1.5	27.258	**22.48**	**0.2180**
	Ours	**27.378**	22.42	0.1926

A.2.4 Time Cost Analysis.

We distinguish offline preparation from online retrieval. In our measured configuration, encoder training converges in approximately 24 hours on four NVIDIA A100 GPUs. Encoding a single LoRA takes approximately 0.5 seconds, excluding weight loading from disk. Building an index of 5,000

adapters on a single GPU takes approximately two hours including disk loading, with weight loading accounting for a substantial portion of the preparation time.

The resulting embeddings are cached and reused across queries while the encoder and indexed adapter weights remain unchanged. Newly added adapters can be encoded incrementally. Table 14 reports online retrieval and composition latency using these precomputed indices, separately from offline training and index construction. These measurements show that index construction is an upfront cost that can be amortized over repeated queries.

All evaluated methods pre-construct their embedding indices offline. Table 14 reports mean online retrieval and composition time over ten trials. LoRAHunter has the lowest measured latency in this comparison. These measurements do not include offline encoder training, preference-data generation, or index construction and therefore do not establish a total-cost advantage.

Table 14: Comparison of time consumption during the adapter retrieval and composition phase.

Method	Stylus	LoRAverse	**Ours**
Time (s) $\downarrow$	1.032	1.812	**0.368**

A.2.5 Metadata Retrieval Evaluation

To validate the effectiveness of our encoder in bridging the modality gap, we conduct a text-to-adapter retrieval experiment using the original adapter descriptions as queries (Table 15). As a supplementary sanity check, we select 500 LoRAs from a held-out test pool of 5K LoRAs and use each selected adapter's original description as a query. We rank all 5K test LoRAs by cosine similarity between the query-text embedding and their Stage 2 weight embeddings, treating the query's source adapter as the relevant item. These results provide a supplementary check of description-to-adapter matching using weight-derived embeddings in the held-out pool.

Notably, the retrieval accuracy does not reach absolute saturation. Because the continuous metric space is explicitly regularized by offline visual rewards, the representations are optimized for actual generative fidelity and physical parameter compatibility instead of rigid textual matching. This outcome highlights a critical insight in multi-adapter composition: pure textual semantic similarity is not strictly positively correlated with superior human-aligned generation quality.

Table 15: Metadata retrieval performance using adapter descriptions as queries.

Method	**Recall@1** $\uparrow$	**Recall@5** $\uparrow$	**Recall@10** $\uparrow$
Representations Retrieval	0.696	0.812	0.855

A.3 Qwen-Image Concept-to-LoRA Qualitative Cases

We provide four qualitative examples from the 200-prompt Qwen-Image evaluation to inspect the relation between prompt decomposition, adapter retrieval, and final generation. Each example uses the same seed for Base and LoRAHunter. These examples are qualitative illustrations rather than an unbiased estimate of aggregate results.

Case A: Serene pond with water lilies. **Prompt:** Perspective by Alice Pasquini, intricate. A peaceful mixed-media artwork that shows a pond with water lilies floating on the surface. The colors are soft and calming, and the overall effect is one of serenity and relaxation.

As show in figure 6 and table 16. The Base image contains the broad pond-and-lily subject but is heavily distorted and lacks a stable composition. Ours preserves the requested pond and water lilies while adding a coherent mixed-media appearance, soft colors, and a calm visual hierarchy.

Case B: Concentric triangle and sacred geometry. **Prompt.** Concentric triangle, sacred geometry, octane render, mesmerizing patterns.

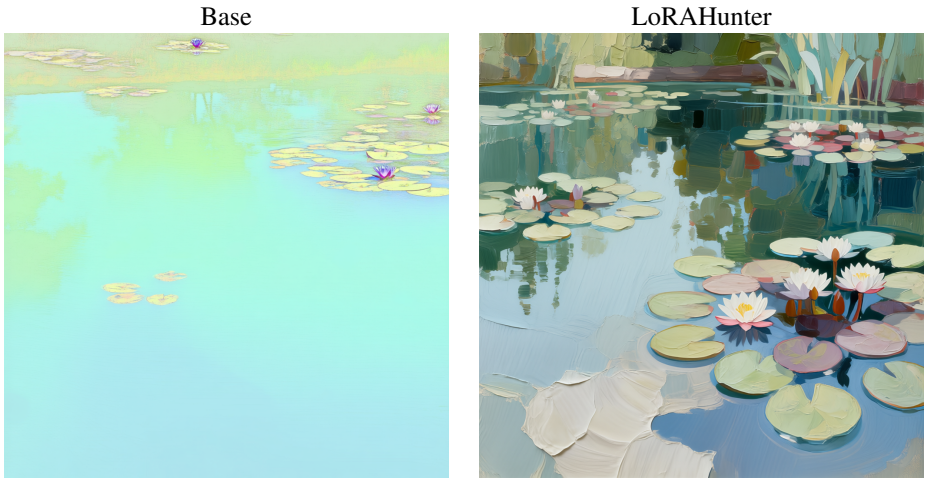

Figure 6: Case A. LoRAHunter preserves the pond and water-lily structure while adding a coherent mixed-media appearance.

Table 16: Case A concept decomposition and retrieved adapters.

Extracted concept	Retrieved LoRA title	LoRA Description summary	Score
peaceful mixed media	Oil Painting	Mixed-media and oil-paint-like rendering with tactile layered surfaces and soft directional lighting.	0.987
pond with water lilies	Lemon LinJiangXian	Serene scenic environments with reflective water, lily pads, pavilions, and quiet natural compositions.	0.948
soft calming colors	Healing picture-book style	Gentle illustrative treatment with simplified forms and a calm decorative mood.	0.904

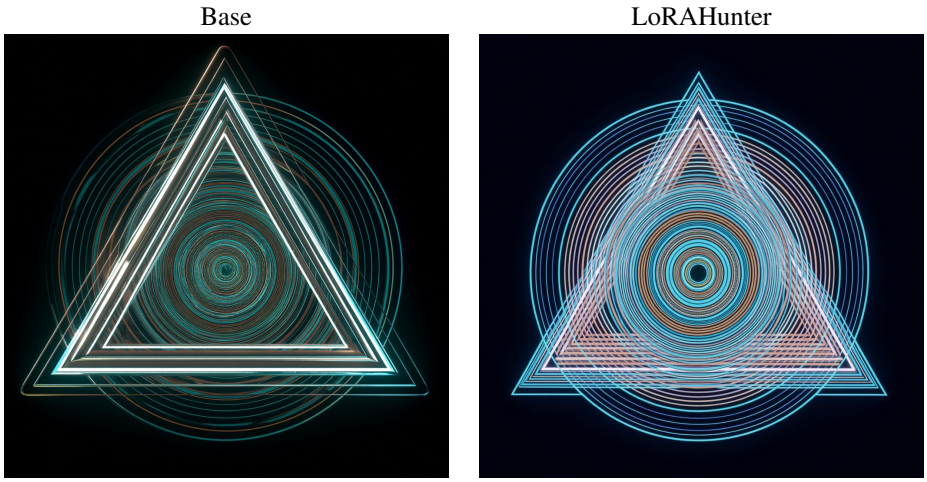

Figure 7: Case B. Both images match the abstract prompt, while LoRAHunter gives the triangular and circular motifs a more centered and balanced structure.

Table 17: Case B concept decomposition and retrieved adapters.

Extracted concept	Retrieved LoRA title	LoRA Description summary	Score
concentric triangle	Miao batik vortex pattern	Symmetric wax-dye-inspired patterns with concentric lines, vortex motifs, and decorative repetition.	0.976
sacred geometry	Eastern Roman style	Geometric and ornamental rendering with strong symmetry and high-contrast decorative surfaces.	0.944
mesmerizing patterns	Adelis pattern	Rhythmic textile-like patterns with symmetry, vivid color blocks, and structured geometric bands.	0.985

As show in figure 7 and table 17. Both generations satisfy the abstract prompt. The difference is compositional rather than semantic: Ours gives the concentric triangles and circular pattern a more centered and balanced alignment, while the Base image has a slightly less stable geometric arrangement.

Case C: Tech company office in a downtown building. **Prompt.** A tech company office in a downtown class-A building.



Figure 8: Case C. LoRAHunter retrieves adapters corresponding to the office concept and downtown-building concept, producing a more polished glass facade and clearer urban context.

Table 18: Case C concept decomposition and retrieved adapters.

Extracted concept	Retrieved LoRA title	Description summary	Score
tech company office	Corporate culture wall design	Modern high-tech corporate interiors with clean geometric layouts and professional office design.	0.998
downtown class-A building	Issaquah downtown	Realistic urban commercial architecture with glass facades, sidewalks, signage, and downtown context.	0.991

As show in figure 8 and table 18. The two retrieved adapters map cleanly to the two main concepts: one supplies the office and corporate-technology context, while the other supplies the downtown architectural setting. Both images satisfy the prompt, but Ours produces a more modern glass-fronted facade with clearer reflections and a more legible interior.

Case D: Serene landscape with a futuristic metropolis. **Prompt.** A hyper-realistic detailed matte painting of a serene landscape with a futuristic metropolis foreground and a super advanced city in the background, barometric projection, featured on ArtStation and Behance, ultrawide angle, f16, dof-1.

Table 19: Case D concept decomposition and retrieved adapters.

Extracted concept	Retrieved LoRA title	Description summary	Score
hyper-realistic detailed matte painting	4K game scene	Detailed photorealistic urban scenes with layered architecture, atmospheric lighting, and game-quality detail.	0.925
serene landscape	Flower	Serene scenic environments with natural scenery, waterways, soft lighting, and picturesque composition.	0.969
futuristic metropolis foreground	Futuristic surreal Shanghai	Futuristic urban architecture with dense layered skyscrapers and a strong metropolitan identity.	0.958

As show in figure 9 and table 19. This example illustrates composition across multiple spatial concepts. The retrieval process separates the urban, futuristic, and serene landscape components, and the generated image highlights the metropolis in the foreground while preserving a greener,

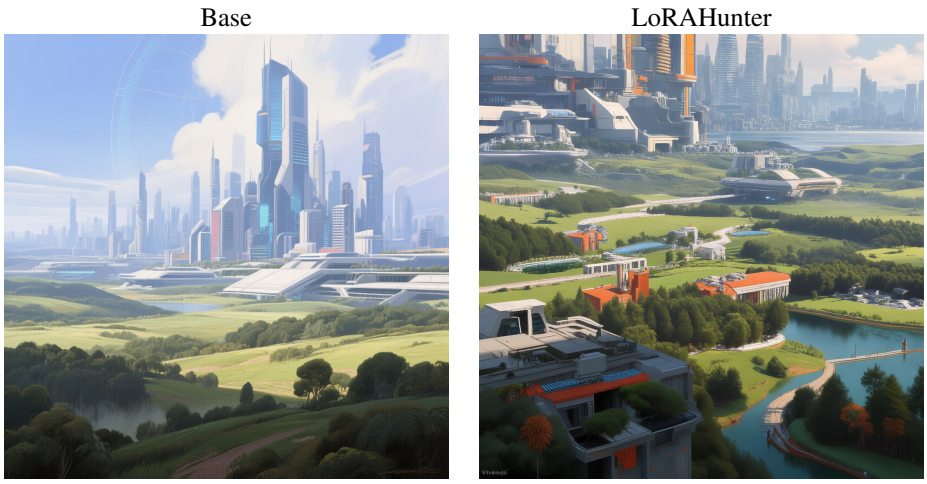

Figure 9: Case D. LoRAHunter separates the urban, futuristic, and serene landscape components and produces a clearer foreground-background hierarchy.

calmer landscape in the surrounding area. Our method excels in foreground-background layout, detail rendering, and overall coherence.