LoRA-Ring: Locality-Preserving Consistent Hashing for Dynamic Adapter Routing in Large Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Parameter-efficient fine-tuning (PEFT) has become the dominant strategy for adapting large language models to specialized domains without retraining billions of parameters. Low-Rank Adaptation (LoRA; Hu et al., 2022) enables lightweight domain adapters, but large-scale deployments introduce a systems problem distinct from the adapters themselves: dynamic adapter routing. Existing approaches — learned classifiers, task-representation routers, and Mixture-of-Experts (MoE) gates — couple the routing decision to the adapter population at training time, so every adapter addition or retirement risks retraining the router. We present LoRA-Ring, a purely theoretical framework that reframes adapter routing as a consistent-hashing problem. Each adapter is represented by the centroid of its training corpus, projected onto a locality-preserving hash ring via a distortion-bounded map (as opposed to a cryptographic hash, which is unsuitable here precisely because it *destroys* locality by design); incoming queries are embedded into the same space and routed to the nearest ring position. Building on Karger *et al.*'s (1997) consistent-hashing theory, we formalize the routing function, state the structural properties it inherits — monotonicity under adapter churn, bounded load imbalance via virtual nodes, logarithmic lookup complexity — and extend the base construction with boundary-aware expertise blending, a cache-locality argument for adapter-serving systems, and hierarchical ring composition. We restrict every claim to what follows analytically from the construction and from established consistent-hashing and locality-preserving-hashing results. **No empirical measurements are reported**; the framework is presented strictly as a theoretical contribution, with an explicit account of which of its claims are unconditional, which depend on a stated assumption about the query distribution, and which require future empirical validation.

1 Introduction

Large language models are increasingly adapted to domain-specific tasks through PEFT rather than full retraining. Instead of maintaining independent checkpoints for legal, medical, financial, and software-engineering domains, an organization can maintain one frozen foundation model alongside a growing library of lightweight LoRA adapters, one per domain or task (Hu et al., 2022). This has motivated a wave of work on how such adapters should be composed, merged, or selected at inference time (Pfeiffer et al., 2021; Huang et al., 2023; Tian et al., 2025; Dhasade et al., 2026).

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

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.

As adapter libraries grow from a handful of curated modules to collections spanning hundreds of narrow domains, *routing* — deciding which adapter, or combination of adapters, should service a given request — becomes the central operational bottleneck. The dominant approach treats routing as a supervised learning problem: a router or task classifier is trained to predict which adapter best matches an incoming query (Tian et al., 2025; Dhasade et al., 2026), and MoE architectures generalize this by learning a gating function jointly with the experts themselves (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al., 2021). These learned approaches work well when the adapter population is relatively static, but they structurally couple the router to the adapter ecosystem: introducing, retiring, or substantially retraining one adapter can require revisiting the router's decision boundary, in the worst case from scratch. In production settings where adapters are added continuously — as new customers, clients, or subdomains are onboarded — this coupling is a maintenance liability, not a one-time cost.

LoRA-RING approaches routing from a different vantage point: distributed-systems theory rather than supervised learning. Consistent hashing (Karger et al., 1997) was developed to solve a structurally analogous problem in distributed caching — minimizing disruption when servers are added to or removed from a hash-based load-balancing scheme — and has since become foundational infrastructure for peer-to-peer lookup (Stoica et al., 2001) and highly available key-value storage (DeCandia et al., 2007). LoRA-RING transposes this machinery onto adapter routing: adapters, rather than server nodes, occupy positions on a hash ring; semantic query embeddings, rather than data keys, are the objects being routed; and an adapter's ring position is derived from a locality-preserving projection of its domain centroid rather than from a cryptographic hash of an identifier. The resulting routing mechanism's core properties — determinism, minimal disruption under adapter churn, and asymptotically balanced load — follow from the same analysis that underlies consistent hashing, rather than from empirical tuning of a learned model.

This paper makes five theoretical contributions: (i) a formal problem statement for adapter routing that separates the semantic representation problem from the placement-and-lookup problem; (ii) the LoRA-RING construction itself, together with the structural guarantees it inherits from consistent-hashing theory and the additional assumptions needed for those guarantees to transfer from arbitrary keys to semantic embeddings; (iii) boundary-aware expertise blending, a continuous relaxation of the discrete nearest-adapter rule analyzed as ring-distance interpolation rather than a learned MoE gate; (iv) a qualitative, property-based comparison against learned routing families, not a benchmark comparison; and (v) an explicit statement of the framework's limitations, including projection distortion, the cold-start problem for corpus-free adapters, and the absence of any mechanism for detecting when semantic locality is a poor proxy for task relevance.

2 Background and Related Work

2.1 Parameter-Efficient Fine-Tuning

The starting point for this work is the observation, formalized by Hu et al. (2022), that the weight updates required to specialize a pretrained model to a downstream task tend to have low intrinsic rank. LoRA exploits this by freezing a transformer's (Vaswani et al., 2017) pretrained weight matrices and injecting a pair of low-rank trainable matrices into selected layers, so only a small fraction of the original parameter count is trained or stored per task. LoRA belongs to a broader PEFT family that includes prompt tuning (Lester et al., 2021), prefix-tuning (Li and Liang, 2021), BitFit (Zaken et al., 2022), and $(IA)^3$ (Liu et al., 2022); Xu et al. (2023) survey these trade-offs systematically. Variants such as AdaLoRA (Zhang et al., 2023) and QLoRA (Dettmers et al., 2023) further improve efficiency but do not change the basic fact motivating this paper: once many independently trained adapters accumulate, a mechanism is required to decide which applies to a given input at inference time.

2.2 Adapter Routing, Selection, and Composition

Several lines of work address multi-adapter selection directly. AdapterFusion (Pfeiffer et al., 2021) learns a composition layer that attends over a fixed set of previously trained adapters, avoiding catastrophic interference between tasks. LoraHub (Huang et al., 2023) treats composition as a gradient-free coefficient-search problem, so a handful of examples from a new task suffices to assemble an effective combination of existing modules without backpropagation. Mixture of LoRA Experts (Wu et

al., 2024) and its derivatives extend MoE gating (Shazeer et al., 2017) specifically to LoRA modules, learning a routing function jointly with, or on top of, the expert adapters. Adapters Selector (Tian et al., 2025) trains a dedicated selector to identify the domain and task of an input, and a routing scheme built on task-level representations derived from small validation sets (Dhasade et al., 2026) aims to make routing cost scale with the number of tasks rather than adapters. All of these share a structural feature: the routing or composition function is itself learned, and is therefore coupled to the adapter set available at training time. LoRA-Ring instead decouples routing from learning, replacing the learned function with a fixed geometric rule over a semantic embedding space.

2.3 Hashing- and Similarity-Based Routing, and the Closest Analog

A natural question is whether hashing or nearest-neighbor structures have already been applied to adapter or expert routing. They have, but not in the form LoRA-Ring proposes. Semantic Library Adaptation (SemLA; Qorbani et al., 2025) is the closest existing analog, developed for open-vocabulary semantic segmentation rather than text: it represents each LoRA adapter by the centroid of CLIP embeddings over its training data — the same semantic-centroid idea underlying Eq. (1) — and retrieves an adapter at inference time by nearest-neighbor search over these centroids. SemLA validates the centroid representation itself, empirically, in a continual-adaptation setting; it does not, however, place centroids on a hash structure of any kind, and its retrieval cost grows with the adapter count with no stated disruption or load-balance guarantee under adapter churn. LoRA-Ring can be read as asking what changes if a centroid-based retrieval scheme of this kind is organized as a consistent-hashing ring instead of a flat nearest-neighbor index: query time becomes $O(\log(nv))$ rather than $O(n)$, and Proposition 1's disruption bound becomes available, at the cost of the ring's own locality distortion (Section 9) rather than exact Euclidean nearest-neighbor accuracy.

Hashing has also been applied to Mixture-of-Experts systems for purposes distinct from adapter routing. Hash Layers (Roller et al., 2021) route tokens to expert feedforward sublayers via a fixed, balanced hash function of local token features — a genuine hash-based routing mechanism, but the hash is not locality-preserving and the construction predates, and is unrelated to, consistent-hashing rings. LSH-MoE (Nie et al., 2024) uses locality-sensitive hashing, but to compress activation tensors for all-to-all GPU communication during MoE training, not to select experts or adapters. StableMoE (Dai et al., 2022) learns and then distills a frozen router, with no hashing at all. Router-Retriever (Lee et al., 2024) and similar similarity-based schemes select among expert models via learned similarity to reference embeddings, again without a hash structure. None of these combine a locality-preserving projection with a consistent-hashing ring and virtual nodes in the way Section 4 does; as far as we have been able to establish, that specific combination, applied to LoRA/PEFT adapter routing, is unaddressed in the existing literature.

2.4 Consistent Hashing and Distributed Hash Tables

Consistent hashing was introduced by Karger et al. (1997) to address hot-spot formation in web caching: a naive hash-based assignment of objects to servers remaps nearly every object when the server count changes, whereas a consistent hash function remaps only the fraction whose assignment must actually change. The same construction — mapping objects and servers onto a shared ring, and assigning each object to the nearest server in a fixed direction — underlies Chord (Stoica et al., 2001), a peer-to-peer lookup table guaranteeing logarithmic-hop lookup, and Dynamo (DeCandia et al., 2007), Amazon's key-value store, which pairs consistent hashing with virtual nodes to distribute data across heterogeneous servers while bounding data movement on server addition or removal. LoRA-Ring borrows this ring construction directly, substituting adapters for servers and semantic query embeddings for data keys.

2.5 Locality-Preserving Dimensionality Reduction

Consistent hashing, in its original form, relies on a cryptographic hash function precisely because such functions are designed to *destroy* structure in their input — similar keys should hash to unrelated positions, spreading load evenly regardless of input distribution. Adapter routing requires the opposite property: semantically similar queries should map to nearby, or identical, ring positions, so routing can proceed by proximity rather than exact match. This places LoRA-Ring's projection step in the tradition of locality-preserving dimensionality reduction rather than cryptographic

hashing. Principal component analysis (Hotelling, 1933) offers a linear projection preserving directions of maximal variance and is a natural default when centroids are drawn from a continuous embedding space such as Sentence-BERT (Reimers and Gurevych, 2019) or contextual representations generally (Mikolov et al., 2013; Devlin et al., 2019). Locality-sensitive hashing (Indyk and Motwani, 1998) offers an alternative, randomized family in which collision probability decreases monotonically with distance, giving a direct route to approximate nearest-neighbor retrieval; graph-based structures such as HNSW (Malkov and Yashunin, 2020) provide yet another route to the same goal. Space-filling curves such as the Hilbert curve (Moon et al., 2001) offer a third option, ordering multi-dimensional points one-dimensionally with near-optimal clustering behavior, making them a natural candidate for ring placement. LoRA-Ring is agnostic to which technique instantiates the projection; the properties argued in Section 5 depend only on locality-preservation, not on the specific method chosen.

2.6 Continual Learning, Forgetting, and Model Merging

A related line of work combines knowledge from multiple tasks without routing, by merging parameters directly. Task arithmetic (Ilharco et al., 2023) shows that the weight difference between a fine-tuned and pretrained model behaves like a vector that can be added or subtracted to compose or remove task-specific behavior; TIES-Merging (Yadav et al., 2023) refines this by resolving sign and magnitude conflicts before merging. Merging is complementary to routing: it produces a single weight set that must satisfy all merged tasks simultaneously, whereas routing preserves each adapter unmodified and selects which parameters to apply per request, avoiding the interference problem that motivates merging. That interference problem is a modern instance of catastrophic forgetting (McCloskey and Cohen, 1989), later addressed at the parameter level by Elastic Weight Consolidation (Kirkpatrick et al., 2017). LoRA-Ring's response to catastrophic interference is structural rather than algorithmic: because each adapter is retrieved and applied independently, no single weight set ever represents more than one domain at a time.

3 Formal Problem Definition

We state the routing problem precisely enough to support the theoretical claims of later sections. Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ denote a set of n LoRA adapters, each trained on a distinct domain or task corpus D_i. Let $\varphi : \mathcal{X} \to \mathbb{R}^d$ be a fixed embedding function mapping raw text inputs to a d-dimensional semantic space; in practice φ may be instantiated by a sentence-embedding model such as Sentence-BERT (Reimers and Gurevych, 2019), but the definitions below do not depend on a particular choice. For each adapter A_i, define its *centroid* $c_i \in \mathbb{R}^d$ as the mean embedding of its training corpus,

$$c_i = \frac{1}{|D_i|} \sum_{x \in D_i} \varphi(x). \tag{1}$$

A *routing function* is a map $R : \mathbb{R}^d \to \mathcal{A}$ that assigns every point in the semantic embedding space to exactly one adapter (Section 6 relaxes this to a distribution over adapters). We say a routing function is:

Definition 1 (Deterministic)**.** *R is a well-defined function of the query embedding alone, with no dependence on request order, system state, or randomness at inference time.*

Definition 2 (Locality-respecting)**.** *There exists a distance metric d_{sem} on $\mathbb{R}^d$ such that, for any query embedding q, $R(q) = \arg\min_i d_{\text{sem}}(q, c_i)$ up to the discretization introduced by the projection and ring-placement steps; that is, routing decisions approximate nearest-centroid assignment under d_{sem}.*

Definition 3 (Churn-stable with disruption bound $\varepsilon(n)$)**.** *For any single adapter insertion or removal, the set of queries whose routing decision changes has expected size at most $\varepsilon(n)$ under a stated distribution over queries, as n grows.*

Definition 4 (Load-balanced with imbalance bound α)**.** *Under a stated distribution over queries, the maximum number of queries routed to any single adapter is at most α times the expected number of queries routed to an average adapter.*

The learned-router methods surveyed in Section 2 optimize R directly against labeled data and therefore attain strong locality-respecting behavior with respect to the training distribution, but they

offer no analytical churn-stability bound: a single adapter insertion may require retraining R over its entire domain, in the worst case changing the routing decision for an unbounded fraction of the query space. LoRA-Ring instead constructs R as a fixed geometric rule derived from $\{c_i\}$ and a ring embedding, sacrificing some locality-respecting accuracy relative to a fully learned router in exchange for a provable churn-stability bound and a provable load-balance bound, both inherited from consistent hashing theory and stated formally in Section 5.

4 LoRA-Ring Architecture

The construction proceeds in four stages.

Stage 1: Centroid computation. For each adapter A_i, compute the centroid c_i as in Eq. (1). Since c_i is a simple average over embeddings that must already be computed to train or evaluate the adapter, this stage introduces no additional training cost beyond the embedding computation itself.

Stage 2: Locality-preserving projection. Project each centroid c_i onto a ring coordinate $\pi(c_i) \in \mathbb{S}$ (the unit circle) using a locality-preserving map $\pi : \mathbb{R}^d \to \mathbb{S}$. As discussed in Section 2, π may be instantiated via PCA followed by angular normalization, via a locality-sensitive hash family, or via a space-filling curve such as the Hilbert curve. The defining requirement, independent of instantiation, is that π be locality-preserving: centroids close under d_{sem} are, with high probability, close under the induced ring distance d_{ring}, and centroids far under d_{sem} are not systematically mapped close together. Direct cryptographic hashing violates this requirement by design and is therefore unsuitable for this stage, even though it is the standard choice in classical consistent hashing.

Stage 3: Ring placement with virtual nodes. Following Karger et al. (1997) and its refinement in Dynamo (DeCandia et al., 2007), each adapter is represented not by a single ring position but by a set of v_i virtual nodes, obtained by applying v_i independent locality-preserving perturbations around $\pi(c_i)$. Virtual nodes reduce the variance of the load each adapter receives, since the law of large numbers applies more favorably to v_i ring positions than to one, and they let heterogeneous adapters — for example, adapters whose training corpora span broader or narrower semantic regions — be allocated a proportionally larger or smaller share of the ring by varying v_i per adapter, directly analogous to weighted virtual-node schemes used in production key-value stores to accommodate heterogeneous server capacity.

Stage 4: Query-time lookup. At inference time, an incoming query q is embedded via φ, projected via the same map π used in Stage 2, and routed to the adapter owning the nearest virtual node in the clockwise direction around the ring. Because virtual-node positions are fixed once computed, this lookup is a binary search over a sorted array of ring positions, giving $O(\log(nv))$ query time ($v = \max_i v_i$), or a balanced binary search tree supporting $O(\log(nv))$ insertion and deletion when adapters are added or removed dynamically.

Unlike the learned routing methods of Section 2, no supervised training of a routing model is required at any stage: the routing decision is a deterministic geometric consequence of the centroids $\{c_i\}$ and the fixed map π. Adding a new adapter requires only computing its centroid, applying π, and inserting v_i new virtual nodes into the ring's sorted order — an $O(v_i \log(nv))$ operation that touches no other adapter's placement.

5 Theoretical Properties

This section states the structural properties of the LoRA-Ring construction that follow from consistent-hashing theory, together with the assumptions required for each to hold in the semantic-routing setting rather than the arbitrary-key setting for which consistent hashing was originally analyzed.

Proposition 1 (Monotonicity — minimal disruption under churn)**.** *In a standard consistent-hashing scheme with n nodes and K keys distributed uniformly around the ring, adding or removing a single node changes the assignment of an expected $O(K/n)$ keys, independent of n and K individually (Karger et al., 1997): only keys assigned to the affected node's ring segment are remapped,*

Algorithm 1 LoRA-Ring construction and lookup

Offline construction (per adapter A_i, on registration)

$c_i \leftarrow \frac{1}{|D_i|} \sum_{x \in D_i} \varphi(x)$ ▷ Eq. 1

for $j = 1, \ldots, v_i$ **do**

  insert $\pi_j(c_i)$ into the sorted ring array ▷ $O(\log(nv))$ each

end for

Online lookup (per query q)

$p \leftarrow \pi(\varphi(q))$

$i^* \leftarrow$ owner of $\mathrm{succ}_{\mathrm{ring}}(p)$ ▷ $O(\log(nv))$ binary search

return A_{i^*}

and no other assignment changes. This transfers directly to LoRA-Ring*: inserting or removing an adapter only perturbs the routing decision for queries whose projected embedding falls within the ring arc previously or newly owned by that adapter's virtual nodes.*

Justification. The property depends only on the ring-placement mechanism (Stage 3) and the lookup rule (Stage 4), and holds regardless of how π is instantiated, since it is a combinatorial property of nearest-clockwise-neighbor assignment on a ring rather than a property of the embedding geometry. We claim no new analysis beyond Karger *et al.*'s original argument, restated here for the semantic-routing setting. □

Proposition 2 (Load balance with virtual nodes)**.** *With a single ring position per adapter, the fraction of the ring — and hence the expected fraction of queries — assigned to each adapter can vary substantially, since ring positions are themselves random or projection-dependent. Introducing v virtual nodes per adapter reduces this variance: the total ring length owned by an adapter is a sum of v roughly independent contributions, so by standard concentration arguments the relative standard deviation of an adapter's ring share shrinks proportionally to $1/\sqrt{v}$, matching the classical analysis of virtual nodes in consistent hashing (Karger et al., 1997; DeCandia et al., 2007).*

Justification. Translating this into a load-balance guarantee for LoRA-Ring additionally requires an assumption on the query distribution: the bound controls the share of ring *length* owned by each adapter, and this share only translates into a share of *query traffic* if queries are not adversarially concentrated on a specific, narrow region of the embedding space relative to the adapters' centroid geometry. We state this as Assumption 1, since it is a property of the deployment workload rather than of the ring construction itself. □

Assumption 1 (Query-locality regularity)**.** *The distribution of incoming query embeddings, after projection through π, does not place disproportionate mass on an arbitrarily small arc of the ring relative to the arcs owned by individual adapters. Under this assumption, the virtual-node load-balance argument transfers from "share of ring length" to "share of query traffic" in expectation.*

Lookup and update complexity. As stated in Stage 4, routing lookup is $O(\log(nv))$ using a sorted array with binary search or a balanced search tree, and adapter insertion or removal is $O(v \log(nv))$, matching the complexity bounds established for Chord-style ring structures (Stoica et al., 2001). Centroid computation for a newly introduced adapter is a one-time cost proportional to the size of its training corpus and is incurred once, regardless of how many virtual nodes are subsequently placed.

Remark 1 (Non-transference of learned-router accuracy)**.** *It is important to state what* LoRA-Ring*'s guarantees do* not *include. Nothing in the construction guarantees that the nearest-centroid adapter is the task-optimal adapter for a given query in the sense that a supervised router, trained on labeled routing decisions, might achieve. The locality-respecting property of Section 3 is a property of the geometric construction relative to the chosen embedding and projection, not a statement about downstream task accuracy. Whether nearest-centroid routing is an adequate proxy for task-optimal routing is an empirical question about the semantic embedding space in use, and is explicitly outside the scope of the theoretical claims made here.*

6 Boundary-Aware Expertise Blending

Section 3 defined routing as a hard assignment of each query to a single adapter. Many realistic queries, however, draw on more than one domain simultaneously — a query about healthcare regulation may require medical, legal, and public-policy knowledge in combination, and a hard routing rule forces an arbitrary choice among adapters whose relevance may be nearly equal.

We relax R to a distribution over adapters, $R : \mathbb{R}^d \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the simplex of probability distributions over $\mathcal{A}$. Given the ring positions of the k adapters whose virtual nodes are nearest to a query's projected position $p = \pi(\varphi(q))$, we define blending weights as a monotonically decreasing function of ring distance,

$$w_i(q) \;=\; \frac{\exp\big(-d_{\text{ring}}(p,i)/\tau\big)}{\sum_{j=1}^{k}\exp\big(-d_{\text{ring}}(p,j)/\tau\big)}, \tag{2}$$

for a temperature parameter $\tau > 0$, normalized so that $\sum_i w_i(q) = 1$. As $\tau \to 0$, Eq. (2) recovers the hard nearest-neighbor rule of Section 4; as $\tau \to \infty$, it approaches a uniform blend over the k nearest adapters.

This construction has two properties worth stating explicitly. First, it is continuous in q away from ties: small perturbations to the query embedding produce small changes in the blending weights, whereas the hard rule of Section 4 is discontinuous at ring-arc boundaries — addressing the "abrupt domain transition" behavior that motivates boundary-aware blending in the first place. Second, it introduces a bias–variance trade-off governed by τ: smaller τ draws on more adapters and is therefore less sensitive to noise in any single centroid estimate, but risks diluting the contribution of the single most relevant adapter when one exists. Selecting τ is analogous to selecting the temperature of a softmax gate in an MoE layer (Shazeer et al., 2017), and the same qualitative considerations apply here; we do not propose a specific value, since doing so would require empirical tuning against a task distribution, outside the theoretical scope of this paper.

7 Cache-Aware Inference and Systems Benefits

Serving many LoRA adapters from a shared foundation model introduces a memory-management problem distinct from routing accuracy: adapters must be resident in accelerator memory to be applied efficiently, and swapping adapters in and out of memory incurs latency and reduces achievable throughput. Systems such as S-LoRA (Sheng et al., 2023) and Punica (Chen et al., 2023) address this directly, using unified memory pools and batched CUDA kernels to serve many adapters concurrently from a small number of accelerators, and PagedAttention (Kwon et al., 2023) addresses the closely related problem of managing KV-cache memory efficiently during autoregressive decoding. LoRA-RING is complementary to, rather than a replacement for, these systems: it does not itself manage GPU memory, but the monotonicity property of Proposition 1 has a direct consequence for the memory-management layer beneath it.

Because LoRA-RING's routing decisions are deterministic functions of query embeddings that vary only within the arc owned by an adapter, and because that arc changes only locally when adapters are added or removed, queries whose embeddings fall within a given semantic region are consistently routed to the same adapter across time, rather than being redistributed whenever the adapter population changes elsewhere on the ring. This consistency is precisely the property that memory-management systems such as S-LoRA and Punica can exploit through standard caching heuristics — a least-recently-used or frequency-based eviction policy is more effective when the mapping from request to adapter is stable, since an adapter's recent access frequency is then a more reliable predictor of its near-future access frequency. We emphasize that this is a structural argument about *why* cache locality should be easier to exploit under LoRA-RING than under a routing scheme whose decision boundaries move whenever the adapter population changes; we make no quantitative claim about the resulting cache hit rate, memory footprint, or latency, since any such claim requires an empirical evaluation against a concrete serving system and workload, outside the scope of this paper.

Table 1: Qualitative comparison of routing families. No cell is a measured quantity.

Property	**Learned selector** (Tian et al., 2025)	**Task-repr. router** (Dhasade et al., 2026)	**MoE gating** (Shazeer et al., 2017; Fedus et al., 2021)	**LoRA-Ring**
Router training required	Yes	Small per-task validation sets	Yes, jointly with experts	No
Cost of adding one adapter	Retrain/fine-tune selector	One task representation	Retrain or extend the gate	One centroid, $O(v \log(nv))$ insertion
Disruption on addition	Not analytically bounded	Not analytically bounded	Not analytically bounded	Bounded by reassigned arc (Prop. 1)
Soft multi-adapter blending	Method-dependent	Method-dependent	Yes, by construction	Yes, via ring-distance weighting (§6)
Basis for routing decision	Learned decision boundary	Learned task-embedding similarity	Learned gate over experts	Fixed geometric rule over centroids
Depends on labeled routing data	Yes	Partially (validation sets)	Yes	No

8 Comparative Theoretical Analysis

Table 1 summarizes, at a qualitative level, how LoRA-Ring's properties relate to those of the routing families surveyed in Section 2. The comparison is stated in terms of structural properties that follow from each method's construction, **not** in terms of measured performance, since no empirical evaluation is presented in this paper.

The qualitative pattern in Table 1 is that LoRA-Ring trades the potentially higher task-routing accuracy of a learned method — which can directly optimize routing against labeled outcomes — for analytically bounded behavior under adapter churn and for eliminating any training requirement on the routing mechanism itself. Which trade-off is preferable depends on the deployment setting: environments with a slowly changing, well-curated adapter set and abundant labeled routing data are better served by a learned router; environments with a rapidly growing or churning adapter population, or without labeled routing data, are better served by a construction such as LoRA-Ring whose guarantees do not depend on retraining.

9 Limitations and Assumptions

The theoretical claims of this paper rest on several assumptions worth stating explicitly rather than leaving implicit.

Projection fidelity. The monotonicity and complexity properties of Section 5 hold unconditionally, but the locality-respecting property of Section 3 depends entirely on π having low distortion. A projection from a high-dimensional embedding space onto a one-dimensional ring necessarily discards information; by the intuition underlying locality-sensitive hashing (Indyk and Motwani, 1998), some semantically distant centroids may be mapped close together on the ring even under a well-chosen projection, particularly as the adapter count grows relative to the projection's target dimensionality. Establishing a distortion bound specific to the adapter-routing setting is left as an open problem.

Centroid stability. The construction assumes an adapter's training corpus is a stable, representative summary of the queries it should serve. If the deployment-time query distribution drifts from the training corpus — as a domain's vocabulary evolves — the centroid computed at registration time becomes a stale proxy, and nothing internal to the ring construction detects or corrects this drift.

Cold-start for centroid-free adapters. Adapters introduced without an associated corpus — for example, via composition methods such as LoraHub (Huang et al., 2023) or task-vector arithmetic (Ilharco et al., 2023) rather than direct training — have no natural centroid and cannot be placed on the ring without an auxiliary mechanism for eliciting a representative embedding.

No resolution of semantic ambiguity. Boundary-aware blending (Section 6) softens the discontinuity at ring-arc boundaries but does not resolve whether ring distance is an adequate proxy for task relevance when a query genuinely straddles two domains; the blending weights reflect geometric proximity, not a learned relationship to downstream performance.

Absence of an adversarial-robustness analysis. The properties of Section 5 are stated relative to a query distribution satisfying Assumption 1. No claim is made about behavior under an adversarially concentrated query distribution, and no defense against one is proposed.

Because of these assumptions, the properties established in this paper should be read as theoretical guarantees of the ring-placement and lookup mechanism itself, conditional on a locality-preserving embedding and projection, and **not** as a proof that LoRA-Ring achieves task-routing accuracy comparable to a learned router in any specific deployment.

10 Extensions

The base construction of Section 4 admits several natural theoretical extensions, each preserving the core ring-lookup mechanism while relaxing one of its simplifying assumptions.

Hierarchical rings. Rather than placing all adapters on a single ring, adapters may first be grouped into coarse domain clusters — for example, a shallow taxonomy of legal, medical, financial, and technical domains — each represented by its own sub-ring, with a top-level ring routing to the appropriate cluster before a second lookup routes within it. This preserves $O(\log)$-style lookup at each level while letting the projection π within a sub-ring be tailored to narrower semantic variation, potentially reducing the distortion concern of Section 9 relative to one flat ring spanning all domains.

Multi-tenant namespaces. In a deployment serving multiple organizations from a shared foundation model, adapters belonging to different tenants may need to remain logically isolated even when their centroids are geometrically close. This can be represented as a namespace partition of the ring, in which lookup first restricts to a query's tenant arc before applying the nearest-clockwise-neighbor rule, at the cost of proportionally fewer effective virtual nodes per tenant.

Weighted virtual nodes for heterogeneous capacity. Section 4 already permits varying v_i per adapter; this generalizes to a formal capacity-weighting scheme, analogous to weighted consistent hashing in heterogeneous server fleets (DeCandia et al., 2007), in which v_i is set proportional to an explicit capacity or confidence score — training-corpus size or embedding-cluster tightness, for instance — rather than fixed uniformly.

Federated adapter ecosystems. Where adapters are contributed by independent parties without a shared training pipeline — the scenario implicit in adapter-sharing platforms such as LoraHub (Huang et al., 2023) — the ring can in principle be maintained as a federated structure, with each party publishing only the centroid and virtual-node positions of its own adapters into a shared, append-only ring. This raises trust and validation questions (verifying a published centroid faithfully represents its adapter) outside the present scope, noted here as a direction for future formalization.

11 Discussion: Routing as a Systems-Level Coordination Problem

The preceding sections position LoRA-Ring as a deliberate departure from the framing that dominates the adapter-routing literature. Existing methods, surveyed in Section 2, largely treat routing as a prediction problem: the goal is an increasingly accurate function from queries to adapters, with progress measured by proximity to an oracle assignment. LoRA-Ring instead treats routing as a coordination problem in the sense familiar from distributed systems: the goal is an assignment rule whose behavior under addition, removal, and re-weighting of resources is analytically predictable, even at some cost to per-decision accuracy.

This reframing does not claim predictive accuracy is unimportant; rather, it argues that accuracy and operational stability are properties of different components of a routing system, and that conflating them by asking one learned function to provide both is a design choice, not a necessity. The comparison in Section 8 makes this trade-off explicit rather than resolving it: a small, stable, well-labeled adapter ecosystem favors a learned router; a large, rapidly churning one, where relabeling after every addition is impractical, favors a construction whose worst-case disruption is bounded by design

rather than by retraining discipline. More broadly, the choice between learned and structural routing for PEFT ecosystems is a design axis worth studying in its own right, much as consistent hashing versus centrally coordinated load balancing is a long-studied axis in distributed systems (Karger et al., 1997; Stoica et al., 2001; DeCandia et al., 2007) rather than a question settled once by a single algorithm.

12 What Would Validate or Falsify These Claims

Every claim in this paper is either a cited prior theorem (Karger *et al.*'s monotonicity and load-balance results), a direct consequence of applying that theorem's construction to centroids instead of arbitrary keys, or an explicitly labeled open question. None of the following experiments have been run; each would test a specific, falsifiable claim rather than the framework in the abstract.

- *Locality-respecting accuracy*: on a labeled multi-domain query set, measure how often nearest-centroid routing under a chosen φ and π agrees with an oracle or learned-router assignment, as a function of adapter count and semantic overlap between domains.
- *Churn disruption*: directly count the fraction of a fixed query set whose routing changes when a single adapter is inserted or removed, and compare against the $O(K/n)$ prediction of Proposition 1.
- *Load balance under real traffic*: measure per-adapter query share under a real or replayed query distribution, with and without virtual nodes, and check whether Assumption 1 is satisfied in practice or is itself the binding constraint.
- *Blending quality*: compare task accuracy under boundary-aware blending (Eq. 2) against hard routing and against a learned MoE gate, across a range of τ, on queries that genuinely span multiple domains.
- *Cache-locality payoff*: measure adapter-swap rate and cache hit rate in a system such as S-LoRA or Punica under LoRA-Ring's routing versus a learned router with equivalent accuracy, holding the serving system fixed.

13 Conclusion and Future Work

This paper developed LoRA-Ring as a theoretical framework that reframes adapter routing for large-scale PEFT deployments as a consistent-hashing problem rather than a supervised-learning problem. By representing each adapter through a semantic centroid, placing centroids on a ring through a locality-preserving projection, and routing queries to their nearest ring neighbor, the construction inherits the monotonicity, load-balancing, and logarithmic-complexity properties established for consistent hashing, subject to the assumptions on projection fidelity and query-distribution regularity stated in Sections 5 and 9. Boundary-aware blending extends the construction from a discrete nearest-adapter rule to a continuous interpolation over ring distance, and Section 8 situates the resulting trade-offs relative to learned classifiers, task-representation routers, and MoE gating.

Because this paper is confined to theoretical claims, the most immediate direction for future work is empirical, along the lines of Section 12. Beyond empirical validation, the extensions of Section 10 — hierarchical rings, multi-tenant namespaces, weighted virtual nodes, and federated ecosystems — merit their own formal treatment, particularly regarding how the guarantees of Section 5 degrade once these constraints are imposed. A further open direction is adaptive centroid maintenance: a principled rule for recomputing an adapter's centroid as its query distribution drifts, addressing Section 9's centroid-stability limitation without reintroducing the retraining coupling this framework set out to avoid.

References

Chen, L., Ye, Z., Wu, Y., Zhuo, D., Ceze, L., and Krishnamurthy, A. (2023). Punica: Multi-tenant LoRA serving. *arXiv:2310.18547*.

Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., and Wei, F. (2022). StableMoE: Stable routing strategy for mixture of experts. *ACL 2022*.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *SOSP 2007*.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *arXiv:2305.14314*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT 2019*.

Dhasade, A., Kermarrec, A.-M., Pavlovic, I., Petrescu, D., Pires, R., Randl, M., and de Vos, M. (2026). Effective LoRA adapter routing using task representations. *arXiv:2601.21795*.

Fedus, W., Zoph, B., and Shazeer, N. (2021). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv:2101.03961*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24, 417–441.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *ICLR 2022*.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. (2023). LoraHub: Efficient cross-task generalization via dynamic LoRA composition. *arXiv:2307.13269*.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2023). Editing models with task arithmetic. *arXiv:2212.04089*.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. *STOC 1998*.

Karger, D., Lehman, E., Leighton, T., Levine, M., Lewin, D., and Panigrahy, R. (1997). Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *STOC 1997*, pp. 654–663.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13), 3521–3526.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. *SOSP 2023*.

Lee, H., Soldaini, L., Cohan, A., Seo, M., and Lo, K. (2024). RouterRetriever: Routing over a mixture of expert embedding models. *arXiv:2409.02685*.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2021). GShard: Scaling giant models with conditional computation and automatic sharding. *ICLR 2021*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *EMNLP 2021*, pp. 3045–3059.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *ACL 2021*.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeurIPS 2022*.

Malkov, Y. A. and Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE TPAMI*, 42(4), 824–836.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24, 109–165.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *NeurIPS 2013*.

Moon, B., Jagadish, H. V., Faloutsos, C., and Saltz, J. H. (2001). Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE TKDE*, 13(1), 124–141.

Nie, X., Liu, Q., Fu, F., Zhu, S., Miao, X., Li, X., Zhang, Y., Liu, S., and Cui, B. (2024). LSH-MoE: Communication-efficient MoE training via locality-sensitive hashing. *NeurIPS 2024*. arXiv:2411.08446.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. (2021). AdapterFusion: Non-destructive task composition for transfer learning. *EACL 2021*.

Qorbani, R., Villani, G., Panagiotakopoulos, T., Botet Colomer, M., Härenstam-Nielsen, L., Segu, M., Dovesi, P. L., Karlgren, J., Cremers, D., Tombari, F., and Poggi, M. (2025). Semantic library adaptation: LoRA retrieval and fusion for open-vocabulary semantic segmentation. *CVPR 2025*. arXiv:2503.21780.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *EMNLP-IJCNLP 2019*.

Roller, S., Sukhbaatar, S., Szlam, A., and Weston, J. (2021). Hash layers for large sparse models. *NeurIPS 2021*. arXiv:2106.04426.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., Gonzalez, J. E., and Stoica, I. (2023). S-LoRA: Serving thousands of concurrent LoRA adapters. *arXiv:2311.03285*.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM 2001*.

Tian, Y., Zhang, B., Tu, Z., and Chu, D. (2025). Adapters Selector: Cross-domains and multi-tasks LoRA modules integration usage method. *COLING 2025*, pp. 593–605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *NeurIPS 2017*.

Wu, X., Huang, S., and Wei, F. (2024). Mixture of LoRA experts. *arXiv:2404.13628*.

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. (2023). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv:2312.12148*.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). TIES-Merging: Resolving interference when merging models. *NeurIPS 2023*.

Zaken, E. B., Goldberg, Y., and Ravfogel, S. (2022). BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ACL 2022*.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. (2023). AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *ICLR 2023*.