PAIR: POOL-ADAPTER INFERENCE FOR ROUTING IN TRAINING-FREE LORA COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are increasingly deployed with growing libraries of task-specific low-rank adaptation (LoRA) adapters that share a backbone. A practical router must operate without task labels or examples, accommodate newly arriving adapters, and avoid scoring the entire pool for every input. We introduce PAIR (Pool-Adapter Inference for Routing), a training-free hierarchical router that treats a LoRA library as a searchable index rather than a flat list. PAIR constructs weight-only fingerprints from each adapter's last-block LoRA factors and organizes them with an online Dirichlet-process Gaussian mixture. New adapters are incorporated from their LoRA weights alone; no learned router is trained or retrained. For routing, a single adapter-free backbone forward produces the hidden state used to rank cluster representatives. PAIR then evaluates additional candidates only within selected clusters and composes the selected adapters with a calibrated compatibility score. We evaluate PAIR on 26 datasets, three 7B/8B backbones, and three backbone-specific pools of 260 adapters. On LLaMA-3.1-8B, PAIR achieves a five-family macro score of 47.9, versus 43.6 for the strongest evaluated non-PAIR baseline. Holding top-20 uniform composition fixed, hierarchical routing alone improves the macro by 2.4 points while reducing adapter-score evaluations from 260 to a median of 190 per input. In an isolated single-threaded CPU benchmark, median per-adapter insertion time is 20–21 ms across the tested pool sizes of 100–260 adapters, with no backbone forward.

1 INTRODUCTION

Large language models are increasingly paired with libraries of hundreds of task-specific low-rank adaptation (LoRA) adapters (Hu et al., 2022). Practitioners can train a small adapter for each downstream task while reusing the same backbone across many tasks (Ostapenko et al., 2024; Sheng et al., 2024). As these adapters accumulate, deployment introduces a different problem: for each input, which adapters should be selected and combined? We study backbone-specific LoRA pools whose adapters share the same rank, insertion pattern, and adapted layers, so that their updates are directly comparable within one serving stack. In this setting, a practical router should adapt to individual inputs, require no task labels, task examples, or learned gating parameters, accommodate a changing adapter pool, and avoid a mandatory scan of every adapter on every request.

Existing approaches satisfy only subsets of these requirements. Static parameter-space merging constructs task- or domain-level compositions outside the per-input routing loop, but reuses the resulting composition across individual inputs (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Chronopoulou et al., 2023). Learned or fitted routers provide input adaptivity, but require labelled examples, few-shot fitting, trained routing parameters, or mechanisms tied to a fixed pool (Pfeiffer et al., 2021; Huang et al., 2024; Zhao et al., 2024; Wu et al., 2024; Feng et al., 2024; Buehler & Buehler, 2024; Xu et al., 2025; Wang et al., 2024; Li et al., 2025). Training-free methods remove this router-training step, yet their per-input computation can still scale directly with the pool. LoRA-on-the-Go (LoGo) (Lee et al., 2026), the closest activation-based baseline in our experiments, scores all M adapters for every input. HiLoRA (Han et al., 2025) is also hierarchical and training-free, but its hierarchy operates over adapter-level Gaussian scoring and token-level rank selection rather than a reusable pool index, and its adapter-level routing remains pool-size dependent. The missing capability is therefore input-adaptive, training-free routing over a growing adapter library without mandatory full-pool scoring.

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.

PAIR starts from a simple observation: the adapter pool itself is searchable. We find empirically that last-block LoRA fingerprints exhibit task-coherent local structure, so real adapters can serve as representatives of different regions of the pool. This suggests a different routing strategy: rather than ranking every adapter first, identify promising regions of the pool and then inspect adapters only within those regions. PAIR turns this structure into a searchable pool index and routes each input through the index before selecting its final active set.

PAIR implements this idea in three stages. First, it maps each adapter to a compact fingerprint computed only from its last-block LoRA factors, without running the backbone. Second, an online Dirichlet-process Gaussian mixture (Ferguson, 1973; Sethuraman, 1994; Blei & Jordan, 2006) organises these fingerprints into a pool index and represents each cluster using real adapters from the library. Third, for each input, one adapter-free backbone pass produces the hidden state used to score cluster representatives. PAIR then selects promising clusters, probes candidates within them, and composes the final active set using an input-dependent compatibility score. The routing procedure uses no task labels, task examples, or learned gating parameters; the pool index is constructed from adapter weights alone.

We evaluate PAIR on 26 datasets, three 7B/8B backbones, and 260-adapter pools. On LLaMA-3.1-8B, PAIR achieves a five-family macro score of 47.9, compared with 43.6 for the strongest evaluated non-PAIR baseline. More importantly, a controlled comparison isolates the effect of route selection itself: with the same top-20 cardinality and uniform composition, hierarchical routing improves the macro from 45.2 to 47.6 while reducing adapter-score evaluations from 260 to a median of 190 per input. Holding the routed set fixed, PAIR's nonuniform weighting adds a further 0.3 points over uniform composition. The fingerprint index also forms task-coherent groups, providing empirical support for the premise that representative-first search can narrow the pool before per-adapter ranking.

Contributions. (i) **Weight-only pool indexing.** We introduce a compact last-block LoRA fingerprint and a Bayesian nonparametric index that organises a backbone-specific adapter pool using adapter weights alone and supports streaming adapter arrivals. (ii) **Representative-first training-free routing.** We introduce a two-level routing procedure that first scores real adapter representatives and then probes candidates within selected clusters, requiring no task labels, task examples, or learned gating parameters. (iii) **Controlled empirical attribution.** Across 26 datasets and three 7B/8B backbones, we evaluate PAIR end to end and separately isolate the effects of hierarchical route selection and generation-side composition, showing that the hierarchy itself improves downstream quality under matched composition while reducing adapter-score evaluations.

2 Related work

Static parameter-space merging. Static merging collapses several adapters or task vectors into one set of weights. Task Arithmetic (Ilharco et al., 2023) composes task vectors by addition, negation, and scalar scaling; TIES-Merging (Yadav et al., 2023) trims small updates, resolves sign conflicts, and averages sign-consistent entries; DARE (Yu et al., 2024) drops and rescales delta parameters before merging. AdapterSoup (Chronopoulou et al., 2023) averages a subset of domain adapters selected for a target domain; once the target-domain soup is chosen, the resulting merge is reused across inputs from that domain. These methods are cheap at inference and perform no input-conditional routing, but the composition cannot vary across individual inputs.

Learned routing over a fixed pool. Learned routing trains or fits a mechanism for selecting or mixing adapters. AdapterFusion (Pfeiffer et al., 2021) learns per-layer attention over adapters; LoRAHub (Huang et al., 2024) fits task-specific scalar weights by few-shot gradient-free search; LoRARetriever (Zhao et al., 2024) trains an embedder for inference-time retrieval. Mixture-of-LoRAs (Feng et al., 2024), MoLE (Wu et al., 2024), X-LoRA (Buehler & Buehler, 2024), MeteoRA (Xu et al., 2025), and LoRA-Flow (Wang et al., 2024) place learned gates inside the transformer, ranging from sequence- or layer-level routing to token-level selection or fusion. ELREA (Li et al., 2025) clusters training examples by gradient features, trains expert adapters per cluster, and weights experts by gradient similarity at inference. These methods adapt per input but require labelled data, few-shot fitting, trained experts, or routing parameters tied to a fixed pool.

Training-free routing. Training-free methods route over an existing pool using adapter-derived signals. Arrow (Ostapenko et al., 2024) scores SVD-based LoRA prototypes by hidden-state align-

ment at token and layer level. LoGo (Lee et al., 2026) probes adapter activations in one forward pass with all pool adapters attached, then composes the top-scoring ones. HiLoRA (Han et al., 2025) selects LoRAs by Gaussian likelihood at sequence level and routes rank-one components at token level. Belofsky (2023) compute adapter weights from cosine similarities between prompt or token representations and adapter centroids. These approaches avoid a trained router, but their rules are adapter-, component-, or centroid-level rather than an online posterior over the pool, and several require pool-wide or pool-size-dependent scoring.

Bayesian-nonparametric pool indexing. Dirichlet-process Gaussian mixtures (Ferguson, 1973; Sethuraman, 1994) place a posterior over a variable number of clusters, with truncated stick-breaking variational inference (Blei & Jordan, 2006) and streaming Chinese-restaurant assignments (Blackwell & MacQueen, 1973; Aldous, 1985) as standard tools. While typically used for clustering, PAIR uses this posterior as an online routing index for a LoRA library and as an explicit factor in per-input LoRA compatibility. App. F writes the methods above as instances of the canonical form in §3.

3 PAIR

Adapters trained on related tasks act similarly on the backbone and concentrate in nearby regions of representation space, so per-input routing is cheaper if it first picks the most relevant region and scores only inside it (Fig. 1). PAIR turns this intuition into a training-free, two-level search over a Bayesian-nonparametric pool index, learning no task-level parameters. Derivations are deferred to App. B–D and the symbol table to App. G.

3.1 Setup and notation

We consider a reusable library of M LoRA adapters $\{(A^{(m)}, B^{(m)})\}_{m=1}^{M}$ (Hu et al., 2022) for a shared backbone W_0, with a common insertion pattern, rank r, and adapted layers $\ell = 1, \dots, L$. We write $A_\ell^{(m)} \in \mathbb{R}^{r\times n_\ell}$, $B_\ell^{(m)} \in \mathbb{R}^{d_\ell \times r}$, fold the LoRA scaling into $A_\ell^{(m)}$, and denote the layer-wise update by $\Delta W_\ell^{(m)} = B_\ell^{(m)} A_\ell^{(m)} \in \mathbb{R}^{d_\ell \times n_\ell}$. The rank-$r$ chain $B_\ell^{(m)}(A_\ell^{(m)}\mathbf{h})$, evaluated as two small matmuls without ever forming $\Delta W_\ell^{(m)}$, is the *factor chain* of adapter m on a hidden state $\mathbf{h}$; one such evaluation is a *factor-chain evaluation*. Throughout, a *score evaluation* denotes the full computation of Eq. (4) for one adapter across all configured channels. When several projections in a block are adapted, we write $(A_\ell^{(m,\omega)}, B_\ell^{(m,\omega)})$ with a channel index ω.

For each input x, we construct a per-position active set $\mathcal{A}^\star_{\ell,i}(x) \subseteq \{1, \dots, M\}$ and composition coefficients $\pi_{\ell,i,m}(x) \in \mathbb{R}$ under a controlled adapter-score budget. Most *LoRA composition* methods, after optional preprocessing of each $\Delta W^{(m)}$, at each layer ℓ and position i,

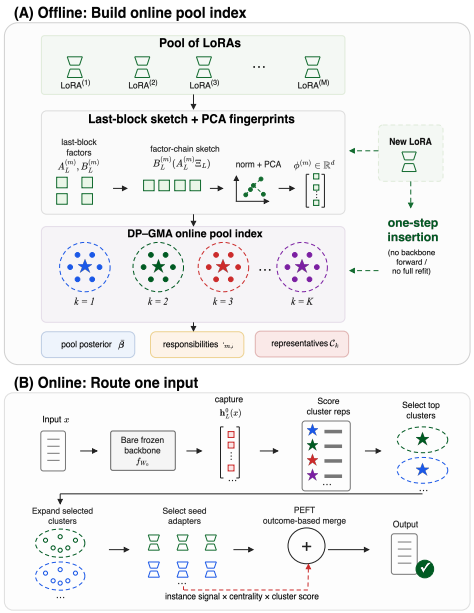

Figure 1: Overview of PAIR: the adapter pool is indexed offline and updated as adapters arrive, then each input is routed hierarchically through cluster representatives and selected candidate adapters.

$$y_{\ell,i}(x) \;=\; W_{0\ell}\,\mathbf{h}_{\ell,i}(x) \;+\; \sum_{m\in\mathcal{A}^\star_{\ell,i}(x)} \pi_{\ell,i,m}(x)\, B_\ell^{(m)}\big(A_\ell^{(m)}\,\mathbf{h}_{\ell,i}(x)\big), \tag{1}$$

where $y_{\ell,i}(x)$ is the output of adapted layer ℓ and $\mathbf{h}_{\ell,i}(x)$ is the residual stream during the composed forward, distinct from the bare-backbone capture $\mathbf{h}^0_L(x)$ used at routing time (§3.4). A method is identified by its $\mathcal{A}^\star$ and π; subscripts on π collapse when the method does not condition on them. PAIR specialises to input-level routing with normalised nonnegative weights $\tilde\pi_m(x) = \pi_m(x)/\sum_{j\in\mathcal{A}^\star(x)}\pi_j(x) \geq 0$. App. F writes the baselines in this form.

3.2 ADAPTER FINGERPRINT

The pool index needs each adapter mapped to a vector $\boldsymbol{\phi}^{(m)} \in \mathbb{R}^d$ that (i) preserves task-relevant similarity, (ii) is cheap in memory, and (iii) is computed from adapter weights alone, so new adapters fold into the index incrementally. We summarise each adapter by the action of its update on a random subspace, restricted to the last adapted block, which already separates task families (Lee et al., 2026). Concretely, draw once a Gaussian sketch $\Xi_L \in \mathbb{R}^{n_L \times N}$ with $N \ll n_L$ and entries $\Xi_{L,ij} \sim \mathcal{N}(0, 1/n_L)$, shared across the pool, and define the per-adapter sketched fingerprint

$$U^{(m)} \;=\; B_L^{(m)}\big(A_L^{(m)}\,\Xi_L\big) \;=\; \Delta W_L^{(m)}\,\Xi_L \;\in\; \mathbb{R}^{d_L \times N}, \tag{2}$$

computed in $\mathcal{O}((d_L + n_L)\, r\, N)$ FLOPs through the two factors. By the Johnson–Lindenstrauss lemma applied row-wise (Johnson & Lindenstrauss, 1984), $N = \mathcal{O}(\varepsilon^{-2} \log(d_L M))$ preserves pairwise Frobenius distances within $1 \pm \varepsilon$ up to a global scale (App. B). We then ℓ_2-normalise, centre, and project $U^{(m)}$ through a frozen orthonormal PCA basis (Pearson, 1901; Jolliffe, 2002) $P \in \mathbb{R}^{d_L N \times d}$ fit once over the initial pool:

$$\boldsymbol{\phi}^{(m)} \;:=\; P^\top\big(\tilde{\mathbf{u}}^{(m)} - \bar{\mathbf{u}}\big) \in \mathbb{R}^d, \qquad \tilde{\mathbf{u}}^{(m)} = \mathbf{u}^{(m)} \big/ \left\|\mathbf{u}^{(m)}\right\|_2, \tag{3}$$

where $\mathbf{u}^{(m)}$ flattens $U^{(m)}$ and $\bar{\mathbf{u}}$ is the empirical mean of $\{\tilde{\mathbf{u}}^{(m)}\}$. The frozen apparatus $(\Xi_L, P, \bar{\mathbf{u}})$ is reused bit-for-bit when new adapters arrive, so insertion is deterministic and backbone-free.

3.3 ONLINE POOL INDEX VIA DP-GMA

The appropriate cluster count is not known a priori, so a fixed-K mixture requires an externally chosen K, and new LoRAs arrive over time, so the index must absorb each arrival without an iterative refit over all stored fingerprints. We therefore use a Dirichlet-process Gaussian mixture over adapter (DP-GMA) fingerprints $\{\boldsymbol{\phi}^{(m)}\}$ (Ferguson, 1973; Blei & Jordan, 2006) whose active component count adapts to the fingerprint cloud under a shared prior, paired with a Chinese-restaurant-process (CRP) predictive rule (Blackwell & MacQueen, 1973; Aldous, 1985) that assigns each arrival in one soft-assignment step.

Online state and CRP update. At time t the index state $\mathcal{G}_t = \{(a_k, b_k, N_k, \boldsymbol{\mu}_k, \widehat{\boldsymbol{\sigma}}_k^2, \mathcal{I}_k)\}_{k=1}^{K_t}$, with K short for K_t, packs, per component, variational stick parameters (a_k, b_k), soft count $N_k = \sum_m r_{m,k}$ over the responsibilities $r_{m,k}$ of component k for adapter m, running centroid $\boldsymbol{\mu}_k$, prior-blended diagonal variance $\widehat{\boldsymbol{\sigma}}_k^2$, and MAP hard partition $\mathcal{I}_k = \{m : \arg\max_j r_{m,j} = k\}$. The CRP responsibility for a new fingerprint $\boldsymbol{\phi}^{(+)}$ is proportional to $N_k\, \mathcal{N}(\boldsymbol{\phi}^{(+)}; \boldsymbol{\mu}_k, \mathrm{diag}(\widehat{\boldsymbol{\sigma}}_k^2))$ for existing clusters and to $\alpha\, \mathcal{N}(\boldsymbol{\phi}^{(+)}; \boldsymbol{\mu}_0, \mathrm{diag}(\boldsymbol{\nu}_0))$ for a brand-new component, with DP concentration α and base measure $(\boldsymbol{\mu}_0, \boldsymbol{\nu}_0)$ given by the mean and variance of the initial fingerprints (App. C). A new component is spawned when the new-cluster responsibility crosses the spawn threshold τ_{spawn} and K_t is below the truncation cap $K_{\max}$; otherwise the responsibilities are renormalised over existing components. Running statistics $(a_k, b_k, N_k, \boldsymbol{\mu}_k, \widehat{\boldsymbol{\sigma}}_k^2)$ are then updated by closed-form weighted Welford recursions (Welford, 1962). Because later components can change where earlier adapters belong, the soft assignments of all stored fingerprints are recomputed once after the initial pool and after every later insertion, which also rebuilds the running statistics, in a single pass (App. C.8).

Pool posterior and cluster representatives. Routing consumes the posterior mixture weight $\bar{\beta}_k = \mathbb{E}_q[v_k] \prod_{j<k}(1 - \mathbb{E}_q[v_j])$ of each cluster, where v_k is the stick-breaking variable of component k with variational posterior $q(v_k) = \mathrm{Beta}(a_k, b_k)$, so $\bar{\beta}_k$ depends on (a, b) alone. Because a centroid $\boldsymbol{\mu}_k$ need not be a pool member, we represent each cluster by its *medoid* $c_k = \arg\max_{m \in \mathcal{I}_k} r_{m,k}$ and keep its top k_{alt} members $\mathcal{C}_k$ as *representatives*, which are real adapters that plug into Eq. (1). We additionally attach a per-adapter *centrality* $\nu_m = \exp\!\big(-\left\|\boldsymbol{\phi}^{(m)} - \boldsymbol{\mu}_{k(m)}\right\|_2^2 / (2\ \mathrm{tr}\ \widehat{\boldsymbol{\sigma}}_{k(m)}^2)\big) \in (0, 1]$ to the home cluster $k(m) = \arg\max_k r_{m,k}$, rewarding adapters that are typical of their cluster rather than at the periphery (App. C.9).

3.4 TWO-LEVEL TRAINING-FREE ROUTING

PAIR instantiates Eq. (1) with a two-level active set drawn from the pool index of §3.3 and a three-factor compatibility score. An adapter is a good match for an input when (i) its rank-r output aligns

Algorithm 1 PAIR: per-input routing and composition.

Require: Backbone W_0; pool $\{(A_\ell^{(m)}, B_\ell^{(m)})\}_{m=1}^{M}$; index $\mathcal{G}_t$ with $\bar{\boldsymbol{\beta}}$, $\{\mathcal{C}_k\}$, $\{\nu_m\}$; input x.
1: Forward x through bare W_0 to obtain $\mathbf{h}_L^0(x)$. ▷ 1 backbone forward
2: $s_m(x) \leftarrow$ Eq. (4) for $m \in \bigcup_k \mathcal{C}_k$. ▷ R score evaluations
3: Compute $\hat{s}_k(x), \tilde{s}_k(x)$ via Eq. (5); $\mathcal{K}^\star(x) \leftarrow \mathrm{TopK}_{k_c}\{\tilde{s}_k(x)\}$.
4: **for** $k \in \mathcal{K}^\star(x)$ in decreasing order of $\tilde{s}_k(x)$ **do**
5: $\quad \mathcal{M}_k \leftarrow \mathrm{TopK}_{k_{\mathrm{probe}}}\{r_{m,k} : m \in \mathcal{I}_k\}$.
6: $\quad s_m(x) \leftarrow$ Eq. (4) for $m \in \mathcal{M}_k$. ▷ $\le k_{\mathrm{probe}}$ score evaluations, reusing cached Level-1 outputs
7: $\quad \mathcal{A}_k^\star(x) \leftarrow \mathrm{TopK}_{k_s}\{\nu_m\, s_m(x) : m \in \mathcal{M}_k\}$.
8: **end for**
9: $\mathcal{A}^\star(x) \leftarrow$ first k_{top} adapters of the $\mathcal{A}_k^\star(x)$; $\tilde{\pi}_m(x) \leftarrow$ Eq. (6).
10: Generate with each $m \in \mathcal{A}^\star(x)$ scaled by $\tilde{\pi}_m(x)$. ▷ 1 generation forward
11: **return** generated response.

strongly with the input's hidden state, (ii) it is typical of its home cluster, and (iii) its home cluster is relevant, judged by its posterior weight and by how well its representatives match the input. The centrality ν_m of §3.3 encodes (ii), and the instance signal and cluster score below encode (i) and (iii). Level 1 scores the representatives of every cluster to select clusters, and Level 2 scores candidates inside the selected clusters.

Instance signal. Capture $\mathbf{h}_L^0(x)$ at the last input position of block-L's adapted projection on one forward through W_0 with all adapters disabled. This bare capture makes the signal independent of M, in contrast to the augmented capture used by Lee et al. (2026) (App. D, Prop. D.1 bounds the score gap by $\mathcal{O}(M\delta^2)$, where δ bounds the spectral norm of every LoRA update). The per-adapter instance factor is the per-call standardised factor-chain score

$$s_m(x) \;=\; \max_{\omega \in \Omega} \exp\Big(\mathrm{clamp}\Big(\frac{\big\|B_L^{(m,\omega)}(A_L^{(m,\omega)}\mathbf{h}_L^0(x))\big\|_2 - \mu^\omega(x)}{\sigma^\omega(x)},\; \pm\tau_z\Big)\Big), \tag{4}$$

where Ω is the set of adapted projection channels at block L, with $\Omega = \{\mathrm{Q}, \mathrm{V}\}$ in our experiments, and $(\mu^\omega(x), \sigma^\omega(x))$ are the channel-wise mean and standard deviation of the inner factor-chain norms across the adapters scored in the same call, namely all cluster representatives at Level 1 and the (up to k_{probe}) candidates of one cluster at Level 2. The standardisation removes per-adapter scale heterogeneity, so $s_m(x)$ reflects relative alignment, and the saturation clamp $\tau_z > 0$ bounds single-adapter dominance.

Two-level active set. Because a representative indicates whether its cluster as a whole is relevant, per-adapter scoring is spent only inside the clusters that the representatives select. At Level 1, each cluster k has a *prototype score* $\hat{s}_k(x)$, the maximum instance signal across its representatives, and a *tempered cluster score* $\tilde{s}_k(x)$, the weighted geometric mean of the prototype score and the cluster's posterior weight $\bar{\beta}_k$:

$$\hat{s}_k(x) = \max_{m \in \mathcal{C}_k} s_m(x), \qquad \tilde{s}_k(x) = \bar{\beta}_k^{\,\eta}\,\big[\hat{s}_k(x)\big]^{1-\eta}, \quad \eta \in [0,1]. \tag{5}$$

The exponent η controls the blend: at $\eta = 0$ the cluster score reduces to pure data signal, at $\eta = 1$ to the pure pool prior. The top k_c clusters form $\mathcal{K}^\star(x) = \mathrm{TopK}_{k_c}\{\tilde{s}_k(x)\}$. At Level 2, each $k \in \mathcal{K}^\star(x)$ is pre-filtered to its top k_{probe} members $\mathcal{M}_k$ by responsibility $r_{m,k}$, those members are ranked by $\nu_m\, s_m(x)$, and the top k_s form $\mathcal{A}_k^\star(x)$. The per-input active set $\mathcal{A}^\star(x)$ collects the $\mathcal{A}_k^\star(x)$ in decreasing order of $\tilde{s}_k(x)$, up to k_{top} adapters.

Compatibility score. This score is the product of the three factors, normalised over $\mathcal{A}^\star(x)$:

$$\pi_m(x) \;=\; \tilde{s}_{k(m)}(x)\,\nu_m\, s_m(x), \qquad \tilde{\pi}_m(x) \;=\; \pi_m(x) \Big/ \textstyle\sum_{j \in \mathcal{A}^\star(x)} \pi_j(x). \tag{6}$$

Since $\tilde{s}_{k(m)}(x)$ is constant within a cluster, Level-2 ranking uses only $\nu_m\, s_m(x)$, and the cluster factor gives adapters in higher-scoring clusters proportionally more mass at composition.

3.5 Per-input procedure and cost

Algorithm 1 folds the pieces above into a per-input procedure: one bare-backbone forward to capture $\mathbf{h}_L^0(x)$, two waves of score evaluations to build $\mathcal{A}^\star(x)$, and one composed forward at generation.

Output-side composition. At generation, PAIR realises Eq. (1) via a batched mixed-adapter forward: at each adapted layer the active adapters in $\mathcal{A}^\star(x)$ contribute their scaled rank-r outputs $\tilde{\pi}_m(x)\, B_\ell^{(m)}(A_\ell^{(m)}\mathbf{h}_{\ell,i}(x))$ to the residual stream, which by distributivity equals applying the patched weight $W_{0\ell} + \sum_m \tilde{\pi}_m(x)\Delta W_\ell^{(m)}$. Each prompt in a batch carries its own weight vector, and the model is restored after the forward to keep the pool clean across inputs.

Per-input cost and design knobs. Let $R = |\bigcup_{k=1}^{K} \mathcal{C}_k|$ be the number of unique Level-1 representatives. With Level-1 factor-chain outputs cached and reused at Level 2, the number of unique score evaluations for input x satisfies $N_{\text{score}}(x) \le \min\big(R + |\mathcal{K}^\star(x)|\, k_{\text{probe}}, M\big)$ (Prop. E.1), and the corresponding factor-chain count is at most $|\Omega|\, N_{\text{score}}(x)$, set by the fitted representative set and the executed candidate sets instead of a full-pool scan. LoGo, the closest training-free activation baseline, scores all M adapters per input; App. E gives the broader cost taxonomy. The hyperparameters $(K_{\max}, k_{\text{alt}}, k_c, k_{\text{probe}}, k_s, k_{\text{top}}, \eta, \alpha, \tau_z)$ control the cost-recall tradeoff.

4 Experiments

We evaluate PAIR in the large-pool setting that motivates the method: a fixed decoder-only backbone paired with a reusable library of task-specific LoRA adapters. The experiments are organised around three questions. First, does PAIR improve aggregate quality and the measured serving profile over existing composition methods? Second, under matched uniform composition, does hierarchical routing improve active-set quality while reducing adapter-score evaluations (App. A.1)? Third, does the online index provide useful representatives and low-cost streaming insertion, and which score choices support routing?

4.1 Experimental setup

Backbones and adapter pool. Following LoGo (Lee et al., 2026), we evaluate three 7–8B decoder-only backbones: LLaMA-3.1-8B (Grattafiori et al., 2024), Qwen-2.5-7B (Yang et al., 2024), and DeepSeek-LLM-7B-Base (DeepSeek-AI et al., 2024). For each backbone, we use the corresponding pool of $M = 260$ rank-8 LoRA adapters trained on distinct Flan-v2 (Longpre et al., 2023) task splits. All adapters for a given backbone share the same insertion pattern and are attached to the Q/V projections at every decoder block. Thus all methods operate on the same fixed adapter library and differ only in how they select, merge, or route adapters at inference time.

Benchmarks and metrics. We evaluate on the complete LoGo benchmark suite: 26 datasets and 60,619 evaluation examples per method/backbone. The suite covers five task families: BBH, Machine Translation, Struct-to-Text, Closed-Book QA, and NLI. We report exact match for BBH, Closed-Book QA, and NLI, BLEU for Machine Translation, and ROUGE-L for Struct-to-Text. Family scores are macro-averaged over datasets, and the overall score is the macro-average over the five families.

Baselines. We compare with the frozen Base model and three classes of adapter-composition methods. Static merging baselines include TIES (Yadav et al., 2023) and AdapterSoup (Chronopoulou et al., 2023). In our evaluation, AdapterSoup performs task-level soup selection on the training split and reuses the resulting weight-averaged adapter for all evaluation inputs of that task; it performs no per-input routing. Fitted or trained routing baselines include LoRAHub (Huang et al., 2024), LoRARetriever (Zhao et al., 2024), and LoRAFlow (Wang et al., 2024). The main training-free routing baseline is LoGo (Lee et al., 2026), which performs augmented hidden-state capture and scores the full adapter pool for each input.

Protocol. All methods use the same prompt templates, deterministic decoding, and offline evaluator. PAIR builds its pool index only from adapter weights: it uses no task labels, task-specific training examples, or evaluation examples to train a router, tune routing weights, or select adapters. Unless otherwise stated, PAIR uses one fixed routing configuration across tasks and backbones: last-block LoRA fingerprints for the pool index, sample-normalised last-block factor-chain norms for instance scoring, max aggregation over Q/V channels, top-20 composition, and $(k_c, k_{\text{probe}}, k_s) = (4, 15, 5)$. In the main LLaMA-3.1-8B evaluation, the fitted index contains K_{active}=80 clusters and R=178 unique Level-1 representatives under the multi-medoid configuration. With $(k_c, k_{\text{probe}}) = (4, 15)$, the four-cluster routing bound is $R + k_c k_{\text{probe}} = 178 + 4 \times 15 = 238$ unique adapter-score evaluations

Table 1: Main results and serving cost on LLaMA-3.1-8B.

	Score						Efficiency / cost		
Method	BBH	MT	S2T	CBQA	NLI	Avg.	Adapter-score evals/input	Routing+Activation ms ↓	Tok/s ↑
Base	32.2 [29.8,34.6]	27.3 [27.0,27.6]	46.6 [46.2,47.0]	62.3 [61.5,63.0]	22.3 [21.4,23.3]	38.1 [37.6,38.7]	0	0.0	90.6
TIES	40.9 [38.3,43.4]	27.7 [27.3,28.0]	43.8 [43.4,44.2]	50.7 [49.7,51.6]	35.9 [34.7,37.2]	39.8 [39.2,40.4]	0	0.0	38.0
AdapterSoup	52.8 [49.9,55.6]	26.5 [26.1,26.8]	39.9 [39.5,40.3]	58.7 [57.8,59.6]	**40.2** [38.9,41.5]	43.6 [42.9,44.3]	0	0.0	37.4
LoRARetriever	42.8 [40.1,45.5]	**28.1** [27.7,28.4]	47.2 [46.8,47.6]	65.4 [64.6,66.1]	32.9 [31.6,34.0]	43.3 [42.6,43.9]	20	29.0	27.2
LoRAHub	46.3 [43.4,49.0]	27.3 [27.0,27.6]	46.8 [46.4,47.2]	54.6 [53.6,55.4]	37.2 [36.0,38.5]	42.4 [41.8,43.1]	260	0.0	38.3
LoRAFlow	37.0 [34.5,39.5]	27.5 [27.2,27.8]	47.1 [46.7,47.5]	62.6 [61.8,63.4]	23.5 [22.5,24.4]	39.5 [39.0,40.1]	260	0.0	36.4
LoGo (Norm)	44.5 [41.8,47.2]	27.9 [27.6,28.3]	45.5 [45.1,45.9]	65.1 [64.3,65.8]	22.3 [21.3,23.3]	41.1 [40.5,41.6]	260	2936.5	3.8
PAIR	**56.8** [54.0,59.7]	27.5 [27.1,27.8]	**50.0** [49.6,50.4]	**67.4** [66.6,68.1]	38.0 [36.7,39.2]	**47.9** [47.3,48.6]	185–200; med. 190	15.7	74.4

Note: Bold marks the best score per column, and underline marks the second-best score. Brackets give 95% percentile bootstrap confidence intervals, resampling examples within each task and recomputing task, family, and average macros.

per input. After Level-1 score reuse, candidate de-duplication, and small-cluster truncation, PAIR scores 185–200 adapters per input (median 190).

Efficiency measurement. We report observed adapter-score evaluations and measured serving cost. Adapter-score evaluations count unique adapter indices submitted to the scoring function, with cached Level-1 scores reused at Level 2. Routing/activation overhead is defined as the per-input median time spent on routing and adapter activation before generation; generation throughput is measured in tokens per second. Full implementation details, including DP-GMA hyperparameters, score-evaluation accounting, baseline fitting protocols, evaluator settings, and profiling hardware, are reported in App. A.

4.2 Main results: two-level routing improves the quality-cost tradeoff

Table 1 reports the main LLaMA-3.1-8B comparison, including both task performance and measured serving cost. Adapter-score evals/input reports only input-conditional adapter scoring. Static or task-fitted compositions such as TIES and AdapterSoup therefore incur zero per-input adapter-score evaluations; their task-level fitting or merge sizes are reported separately in App. A. Per-task breakdowns and full Qwen-2.5-7B and DeepSeek-LLM-7B-Base results are reported in App. A.4. PAIR obtains the best average score while remaining training-free: 47.9, compared with 43.6 for AdapterSoup, 43.3 for LoRARetriever, 42.4 for LoRAHub, and 41.1 for LoGo. The bootstrap intervals in Table 1 show that the overall gain over the strongest non-PAIR baselines is stable.

The gains are largest on families where related-adapter transfer appears most useful. On BBH, PAIR improves over LoGo by 12.3 points and over the strongest non-PAIR baseline by 4.0 points. On Struct-to-Text, it improves over LoGo by 4.5 points and obtains the best score in the table. PAIR is also best on Closed-Book QA (67.4) and improves NLI from LoGo's 22.3 to 38.0.

The most direct comparison is LoGo, since both methods are training-free and activation-based. LoGo scores all $M = 260$ adapters for every input, whereas PAIR routes through cluster representatives and selected within-cluster candidates. This yields a better serving profile: PAIR scores 185–200 adapters per input (median 190), rather than all M=260, and reduces measured routing/activation overhead from 2936.5 ms to 15.7 ms. Median throughput increases from 3.8 tok/s to 74.4 tok/s. The evaluation count is controlled by the representative and within-cluster budgets rather than by a full M-adapter scan (§3.5, App. E). Thus the complete two-level design improves the quality-cost tradeoff over full-pool training-free scoring. A controlled ablation in App. A.1 isolates the contributions of hierarchical route selection and generation-side composition weighting.

4.3 Online pool indexing: task-coherent grouping and streaming insertion

DP-GMA supplies the three properties required by routing: task-coherent groups, data-adaptive granularity, and streaming insertion without full index refitting. Task-family labels provide a held-out evaluation of cluster coherence, separate from index fitting and test-time routing.

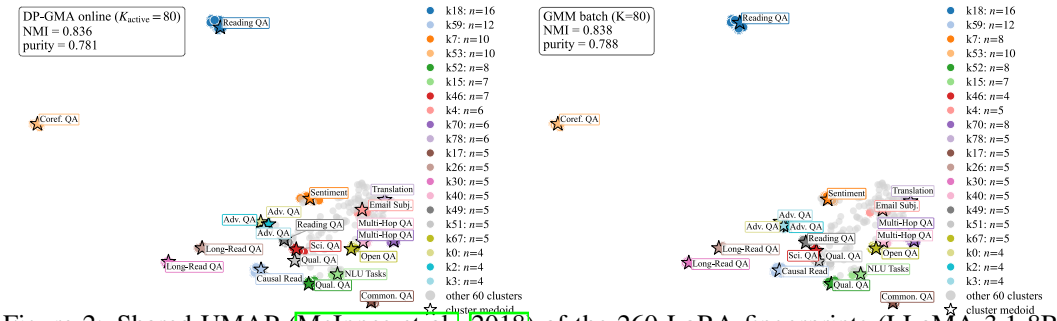

Figure 2: Shared UMAP (McInnes et al., 2018) of the 260 LoRA fingerprints (LLaMA-3.1-8B backbone). **Left**: DP-GMA online fit (streaming CRP, K_{active}=80). **Right**: batch Gaussian-mixture EM fit at the same K=80, Hungarian-relabelled (Kuhn, 1955) to DP-GMA ids by maximum cluster overlap; matched ids share colour. **The two partitions agree on** $237/260$ **adapters** (91%). Only the top 20 clusters by membership are shown; stars mark cluster medoids and coloured boxes give the dominant task family per cluster.

Table 2: Isolated pool-index update benchmark under streaming adapter arrivals.

M	Insert p50 (ms)	Insert p95 (ms)	Fingerprint p50 (ms)	CRP update p50 (ms)	Batch refit (s)	Batch / insert p50	K_β
50	–	–	–	–	0.40 ± 0.02	–	25.0 ± 0.0
100	20.15 ± 1.01	22.31 ± 2.09	3.53	4.27	0.82 ± 0.01	$40.6\times$	11.0 ± 2.0
150	20.00 ± 0.49	22.38 ± 1.26	3.57	4.48	1.26 ± 0.29	$62.8\times$	12.3 ± 2.9
200	20.59 ± 0.91	23.47 ± 2.62	3.62	4.69	1.38 ± 0.01	$66.9\times$	12.3 ± 2.1
260	20.58 ± 0.53	21.68 ± 0.58	3.63	4.79	1.73 ± 0.04	$84.2\times$	13.0 ± 1.7

K_β counts components whose posterior mass exceeds the threshold used in this insertion benchmark; it is distinct from K_{active} used in the partition-quality analysis.

Task-coherent fingerprints. Figure 2 shows that last-block fingerprints expose task-coherent structure. On the LLaMA-3.1-8B pool, adapters from related families form visible islands, and cluster medoids inherit the dominant family labels of their neighbourhoods. This supports the central premise of two-level routing: a small number of real adapter representatives can summarize regions of the pool.

Data-adaptive granularity without a K sweep. A fixed-K GMM recovers similar structure at a well-chosen cluster count. Sweeping batch GMM on the same fingerprints (Fig. A.1, App. A.3) shows that AMI and ARI rise sharply up to $K \approx 40$ and plateau over $K \in [80, 120]$, peaking at $K = 100$ with AMI 0.600 and ARI 0.461. DP-GMA selects $K_{\text{active}} = 80$, which lies on this plateau and is within 0.01 AMI and 0.02 ARI of the peak. Thus the nonparametric index lands on the same high-quality granularity plateau without an explicit sweep over K.

Streaming quality and insertion cost. At the same granularity, the online fit closely matches a batch diagonal GMM. After Hungarian matching, the online DP-GMA and batch EM partitions agree on $237/260$ adapters, and all 20 of the top 20 clusters share the same dominant family label. Aggregate clustering scores (Vinh et al., 2010) are nearly identical: NMI is 0.836 for online DP-GMA and 0.838 for batch EM, while purity is 0.781 versus 0.788. DP-GMA thus matches batch GMM's partition quality while additionally supporting online insertion and posterior-weighted routing, the two properties PAIR relies on. App. A.3 extends this analysis to DeepSeek-LLM-7B-Base and Qwen-2.5-7B.

For insertion, after computing the fixed-basis fingerprint, the CRP predictive step evaluates K_t diagonal-Gaussian component likelihoods and one new component likelihood, costing $O(K_t d)$ for fingerprint dimension d. One insertion therefore costs $O(C_{\text{fp}} + K_t d)$, where C_{fp} is the fingerprint/projection cost and is independent of the current pool size M_t. By contrast, refitting a batch diagonal GMM costs $O(I M_t K d)$ for I EM iterations. This mirrors the motivation of online variational inference for Bayesian-nonparametric models: streaming updates avoid repeated passes over all accumulated data while preserving data-adaptive component counts (Wang et al., 2011). Full derivations are given in App. A.6.

Table 2 confirms this scaling in an isolated pool-index benchmark. Inserting one adapter takes roughly 20–21 ms as the pool grows from $M = 100$ to $M = 260$, while batch refitting grows from 0.82 s to 1.73 s. At $M = 260$, refitting is $84.2\times$ slower than one streaming insertion, and the CRP update itself remains below 5 ms. The benchmark isolates index maintenance on one CPU thread using LoRA factor matrices.

Table 3: Seed-score ablations on LLaMA-3.1-8B. Avg. is the five-family macro-average with 95% bootstrap confidence intervals.

Variant	BBH	MT	S2T	CBQA	NLI	Avg. (CI)	Evals/input
Single-medoid representatives							
Raw Q	54.29	27.41	49.15	65.91	37.90	46.93 [46.25,47.60]	116
Sketch-z Q	57.64	27.50	47.45	67.73	37.63	47.59 [46.95,48.22]	124
Sample-z Q	58.23	27.46	48.82	65.79	37.88	47.64 [46.99,48.29]	93
Sample-z max(Q,V)	53.92	27.23	50.77	67.52	37.66	47.42 [46.74,48.01]	92
Multi-medoid representatives							
Sample-z Q + multi-medoid	56.67	27.57	46.46	65.12	37.72	46.71 [46.02,47.33]	94
PAIR: Sample-z max(Q,V) + multi-medoid	56.84	27.49	50.01	67.37	37.98	47.94 [47.29,48.58]	92

4.4 Calibrated seed scoring and representative expansion

Table 3 ablates the seed-score construction used by PAIR on the LLaMA-3.1-8B. The variants test score calibration, Q/V channel aggregation, and medoid expansion. The final row, Sample-z max(Q,V) + multi-medoid, is the PAIR configuration used in all main experiments. Single-medoid variants use one Level-1 representative per active cluster (R=80), whereas multi-medoid variants use R=178 unique representatives in the fitted LLaMA index. Their four-cluster routing bounds are therefore 140 and 238 adapter-score evaluations, respectively. Executed counts reflect Level-1 score reuse, candidate de-duplication, and small-cluster truncation. The two blocks therefore separate score construction from the effect of expanding cluster representation. Full diagnostics are reported in App. A.5.

Calibration strengthens the raw factor-chain signal. Both normalised Q variants improve the five-family macro over Raw Q (46.93 $\rightarrow$ 47.59–47.64), showing that per-input normalisation makes the factor-chain signal more useful in aggregate.Q/V aggregation and medoid expansion interact. Adding $\max(Q, V)$ to the sample-normalised Q score shifts the gain toward Struct-to-Text and Closed-Book QA. Medoid expansion is strongest when combined with calibrated Q/V scoring: the PAIR configuration combines sample-level normalisation, Q/V aggregation, and medoid expansion, yielding the highest observed five-family macro (47.94). Oracle diagnostics are reported in App. A.5. Once the oracle cluster enters the executed cluster set, the oracle adapter survives Level 2 in every observed mapped example.

5 Discussion

PAIR treats a backbone-specific LoRA library as a searchable pool index rather than a flat list. Under matched top-20 uniform composition, hierarchical routing improves the five-family macro from 45.2 to 47.6 while reducing the median number of adapter-score evaluations from 260 to 190. With the routed set fixed, nonuniform weighting adds a further 0.3 points. These controlled ablations indicate that the larger gains come from selecting a stronger active set and composing multiple selected adapters, while nonuniform weighting provides a smaller refinement. The matched-composition result also shows that the hierarchy is not merely a cheaper approximation to flat top-k ranking: in the evaluated setting, it selects a stronger set.

PAIR assumes that adapters within an index share the backbone, rank, insertion pattern, and adapted layers; we therefore fit a separate index for each backbone while keeping the index-fitting and routing hyperparameters fixed. Its efficiency benefit depends on a compact representative set, with $N_{\text{score}}(x) \leq \min(R + q(x)k_{\text{probe}}, M)$, and sublinear scaling requires $R + q_{\max}k_{\text{probe}} = o(M)$. In the main LLaMA setting, $R = 178$ for $M = 260$, so the current results establish a finite reduction to a median of 190 score evaluations rather than asymptotic sublinear scaling. The evaluation is limited to homogeneous 260-adapter Flan-v2 pools, and robustness to larger, heterogeneous pools and different arrival distributions remains open.

Overall, when a LoRA library contains reusable fingerprint-space structure, indexing the pool can improve route quality while avoiding a mandatory flat per-input scan.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing, finding related work, and assisting with the implementation. We have not used generative AI tools for developing the method, designing the experiments, or analysing the results. All suggested references were checked against their published records, and AI-assisted code was reviewed and tested by the authors before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithms 1 and 2 specify the full routing and index-update procedures. App. A.2 lists all hyperparameters of PAIR, the decoding and scoring settings, the baseline fitting protocols, and the profiling hardware. App. C derives the online updates, and Apps. B, D and E state the assumptions and proofs of each formal claim. Per-task results for all three backbones are in App. A.4. The backbones and all 26 evaluation datasets are publicly available.

REFERENCES

David J. Aldous. Exchangeability and related topics. In *Ecole d'Ete de Probabilites de Saint-Flour XIII – 1983*, volume 1117 of *Lecture Notes in Mathematics*, pp. 1–198. Springer, Berlin, 1985. doi: 10.1007/BFb0099421.

Joshua Belofsky. Token-level adaptation of LoRA adapters for downstream task generalization. In *Proceedings of the 2023 6th Artificial Intelligence and Cloud Computing Conference (AICCC)*, pp. 168–172, 2023. doi: 10.1145/3639592.3639615.

David Blackwell and James B. MacQueen. Ferguson distributions via polya urn schemes. *The Annals of Statistics*, 1(2):353–355, 1973. doi: 10.1214/aos/1176342372.

David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006. doi: 10.1214/06-BA104.

Eric L. Buehler and Markus J. Buehler. X-LoRA: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2):026119, 2024. doi: 10.1063/5.0203126.

Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2054–2063, 2023. doi: 10.18653/v1/2023.findings-eacl.153.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. doi: 10.1002/rsa.10073.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhihong Sha, Zhuoshu Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint*

International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING), pp. 11371–11380, 2024.

Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973. doi: 10.1214/aos/1176342360.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ziyi Han, Huanyu Wang, Zeyu Zhang, Xiangxiang Dai, Xutong Liu, and John C. S. Lui. HiLoRA: Adaptive hierarchical LoRA routing for training-free domain generalization. *arXiv preprint arXiv:2510.12266*, 2025.

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

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. In *Conference on Language Modeling (COLM)*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, Providence, RI, 1984. doi: 10.1090/conm/026/737400.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, 2 edition, 2002. doi: 10.1007/b98835.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi: 10.1002/nav.3800020109.

Seungeon Lee, Soumi Das, Manish Gupta, and Krishna P. Gummadi. LoRA on the go: Instance-level dynamic LoRA selection and merging. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39583–39601, 2026. doi: 10.18653/v1/2026.acl-long.1837.

Yinghao Li, Vianne Gao, Chao Zhang, and MohamadAli Torkamani. Ensembles of low-rank expert adapters. In *International Conference on Learning Representations (ICLR)*, 2025.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22631–22648, 2023.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861.

Oleksiy Ostapenko, Zhan Su, Edoardo M. Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. Towards modular LLMs by building and reusing a library of LoRAs. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 38885–38904, 2024.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi: 10.1080/14786440109462720.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 487–503, 2021. doi: 10.18653/v1/2021.eacl-main.39.

Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. SLoRA: Scalable serving of thousands of LoRA adapters. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 6, 2024.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

Chong Wang, John Paisley, and David M Blei. Online variational inference for the hierarchical dirichlet process. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 752–760, 2011.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. LoRA-Flow: Dynamic LoRA fusion for large language models in generative tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12871–12882. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.695.

B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. doi: 10.1080/00401706.1962.10490022.

David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014. doi: 10.1561/0400000060.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of LoRA experts. In *International Conference on Learning Representations (ICLR)*, 2024.

Jingwei Xu, Junyu Lai, and Yunpeng Huang. MeteoRA: Multiple-tasks embedded LoRA for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 7093–7115, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning (ICML)*, 2024.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462, 2024. doi: 10.18653/v1/2024.findings-acl.263.