CAST-LoRA: Cross-Attentive Soft-Transition MoE-LoRA for Multi-Task Adaptation

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient multi-task fine-tuning is important for deploying large language models (LLMs) across diverse tasks under limited training, storage, and serving budgets. However, existing Low-Rank Adaptation (LoRA) methods for multi-task adaptation still have the following limitations that require further improvement in performance: (1) training a single adapter across all tasks induces severe task interference in multi-task scenarios. (2) routing over a flat expert pool neglects dedicated shared capacity, failing to capture common knowledge across tasks. To address these problems, this paper proposes CAST-LoRA, a novel multi-task Mixture-of-Experts Low-Rank Adaptation (MoE-LoRA) framework. By clustering training data to guide early routing and then annealing cluster-guided assignments into input-dependent soft routing, CAST-LoRA stabilizes task-to-expert specialization and mitigates task interference. CAST-LoRA further introduces role-separated task-specific, entropy-aligned shared, and floor-preserved shared experts, providing dedicated capacity for routing uncertainty and reusable cross-task knowledge. Additionally, a cross-attentive task encoder extracts compact task representations with $\mathcal{O}(Ld)$ rather than $\mathcal{O}(L^2d)$ attention interactions, where L is the source-sequence length and d is the hidden dimension. Extensive experiments with 10 baselines on 7 benchmarks demonstrate that CAST-LoRA achieves the optimal average accuracy, reduces gradient conflict to an optimal level of 1.09%, cuts task-encoder latency by 86.9%, cuts task-encoder peak memory by 23.8%, lowers expert-parameter redundancy by 26.8%, and increases effective expert directions by 26.8%.

1 Introduction

Large language models (LLMs) provide a common foundation for language understanding, generation, reasoning, and knowledge-intensive applications. Rather than maintaining a separately tuned model for every capability, multi-task fine-tuning (MTFT) jointly optimizes one pretrained model on a mixture of instruction-formatted tasks. By sharing representations across tasks, MTFT can transfer useful knowledge, improve generalization, and consolidate many capabilities into one deployable model (Wei et al., 2022; Sanh et al., 2022; Chung et al., 2024; Longpre et al., 2023). However, full-parameter fine-tuning updates the entire backbone and therefore incurs substantial training, memory, storage, and deployment costs. Parameter-efficient fine-tuning (PEFT) instead freezes the backbone and trains compact adapters (Houlsby et al., 2019), soft prompts (Lester et al., 2021), activation-scaling vectors (Liu et al., 2022), or quantized low-rank updates (Dettmers et al., 2023). In particular, Low-Rank Adaptation (LoRA) (Hu et al., 2022) represents each weight update with two low-rank matrices that can be merged into the backbone for deployment. These properties make LoRA a practical basis for multi-task adaptation under limited compute and storage budgets.

Existing methods for multi-task adaptation can be organized into two broad families. The first shares one trainable parameter set across tasks through full fine-tuning or a single adapter. LoRA (Hu et al., 2022) constrains updates to a low-rank subspace; LoRA+ (Hayou et al., 2024) assigns different learning rates to the two factors; AdaLoRA (Zhang et al., 2023) allocates rank by parameter importance; PiSSA (Meng et al., 2024) initializes adapters from principal singular components; and DoRA (Liu et al., 2024) separates weight magnitude from direction. These methods improve efficiency, but mixed-task examples still update the same dense or low-rank parameter space. A single shared adapter forces heterogeneous tasks to compete within one low-rank update space, inducing severe task interference in multi-task scenarios.

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.

The second family builds on conditional expert computation (Jacobs et al., 1991; Shazeer et al., 2017; Lewis et al., 2021; Du et al., 2022) and routes each input through a subset of LoRA branches. Extremely parameter-efficient MoE (Zadouri et al., 2024) shows that sparse expert adaptation can operate under a small trainable budget. MoA (Feng et al., 2024) combines domain-specific adapters through a supervised selector; MixLoRA (Li et al., 2024) places TOP-K LoRA experts in feed-forward blocks; and LoRAMoE (Dou et al., 2024) assigns different experts to adaptation and knowledge preservation. More recent designs refine specialization: HMoRA (Liao et al., 2025) combines token- and task-level routing, FlyLoRA (Zou et al., 2025) introduces implicit rank-wise routing, and HotMoE (Huang et al., 2026) models layer-dependent expert interactions. However, most input-routed MoE-LoRA methods select from a flat or functionally homogeneous expert pool without jointly reserving dedicated cross-task branches for routing uncertainty and reusable knowledge, thereby failing to capture common knowledge across tasks.

To address these problems, this work presents CAST-LoRA (**C**ross-**A**ttentive **S**oft-**T**ransition LoRA), a multi-task MoE-LoRA framework that separates task-specific adaptation from cross-task sharing. First, a learned cross-attention query condenses the input sequence into a compact task representation, which is combined with token-level states to produce input-dependent routing distributions. Next, semantic clusters constructed from training prompts provide an informative task-to-expert assignment at the start of optimization. An annealing schedule then transfers routing control progressively from this cluster-guided assignment to the learned soft router, stabilizing early specialization while removing the need for task labels at inference. Finally, the expert bank is divided into task-specific and shared branches. Task-specific experts isolate cluster-related update directions to reduce interference, whereas an entropy-aligned shared expert serves routing-uncertain inputs and a floor-preserved shared expert maintains an active path for reusable knowledge. Together, these components establish stable specialization, preserve shared capacity, and mitigate task interference.

The primary contributions are summarized as follows:

- CAST-LoRA introduces a role-separated MoE-LoRA architecture with task-specific, entropy-aligned shared, and floor-preserved shared experts. Task-specific branches isolate cluster-related low-rank directions, while two regularized shared branches support uncertain routing and reusable knowledge.
- Two additional designs improve routing practicality and efficiency. Annealed routing uses semantic clusters only as temporary training guidance before switching to input-dependent soft routing. A single-query task encoder reduces attention interactions from $\mathcal{O}(L^2d)$ to $\mathcal{O}(Ld)$ for source length L and hidden dimension d, consequently delivering lower latency and a smaller memory, especially for longer sequences and larger batches.
- Extensive experiments conducted on multiple benchmark datasets demonstrate the superiority of the proposed framework: CAST-LoRA (a) achieves optimal performance with a 64.95 average on seven benchmarks; (b) reduces gradient conflict to an optimal level of 1.09%, 78.5% below the best baseline value; (c) reduces expert-parameter redundancy by 26.8% while increasing effective expert directions by 26.8% relative to a matched flat expert pool; and (d) cuts task-encoder latency by 86.9% and peak memory by 23.8%.

2 PRELIMINARIES

Let $\mathcal{D}$ contain mixed-task pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, \ldots, x_L)$ is the source and $\mathbf{y}$ is the response. PEFT freezes backbone Θ and trains adapter parameters Φ for $p(\mathbf{y} \mid \mathbf{x}; \Theta, \Phi)$.

For a frozen weight $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA (Hu et al., 2022) learns the rank-r update

$$\Delta\mathbf{W} = \frac{\alpha}{r}\mathbf{B}\mathbf{A}, \qquad \mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}, \quad \mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{1}$$

where α scales the update and only $\mathbf{A}$ and $\mathbf{B}$ are trainable. MoE-LoRA routes inputs to K such updates (Shazeer et al., 2017; Fedus et al., 2022). For activation $\mathbf{h} \in \mathbb{R}^{d_{\text{in}}}$, it computes $\mathbf{o} = \mathbf{W}\mathbf{h} + \sum_{i=1}^{K} g_i(\mathbf{x}, \mathbf{h})\Delta\mathbf{W}_i\mathbf{h}$, where each $\Delta\mathbf{W}_i$ follows Eq. (1). The gates satisfy $g_i \geq 0$ and $\sum_i g_i = 1$, sparse routing normalizes the TOP-K_{act} entries.

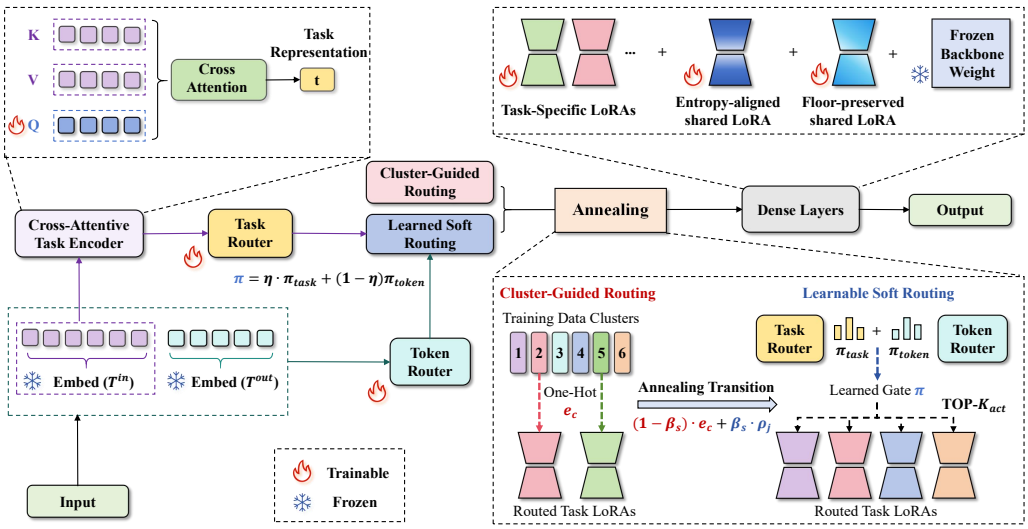


Figure 1: Overview of CAST-LoRA.

3 METHOD

CAST-LoRA combines a cross-attentive task encoder, annealed routing, and task-specific/shared experts (Figure 1). Training prompts define K_t semantic clusters, with $\mathcal{C}_i$ mapped to task expert E_i^{task}. Cluster indices $c(\mathbf{x}) \in \{1, \ldots, K_t\}$ guide early routing, inference uses only $\mathbf{x}$.

3.1 CROSS-ATTENTIVE TASK ENCODER

The task-level router consumes one global vector, although a Transformer task encoder updates every source position alongside a learned task token (Liao et al., 2025). CAST-LoRA instead uses one learned query to read the source sequence directly.

Let $\mathbf{H} = [\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_L^{(0)}]^\top \in \mathbb{R}^{L \times d}$ contain source embeddings from the frozen input-embedding layer, where d is the hidden dimension and superscript (0) denotes the embedding level. Let $\mathbf{q} \in \mathbb{R}^d$ be a learned query shared across examples. For attention head $m \in \{1, \ldots, M\}$ with width $d_h = d/M$, define $\mathbf{q}_m = (\mathbf{W}_m^Q)^\top \mathbf{q}$, $\mathbf{K}_m = \mathbf{H}\mathbf{W}_m^K$, and $\mathbf{V}_m = \mathbf{H}\mathbf{W}_m^V$. Here, $\mathbf{W}_m^Q, \mathbf{W}_m^K, \mathbf{W}_m^V \in \mathbb{R}^{d \times d_h}$ are trainable projection matrices. The head then computes

$$\mathbf{a}_m = \text{softmax}\left(\frac{\mathbf{K}_m \mathbf{q}_m}{\sqrt{d_h}} + \mu\right), \qquad \mathbf{z}_m = \mathbf{V}_m^\top \mathbf{a}_m, \tag{2}$$

where $\mu \in \{0, -\infty\}^L$ masks padding positions, $\mathbf{a}_m \in \mathbb{R}^L$ contains attention weights, and $\mathbf{z}_m \in \mathbb{R}^{d_h}$ is the head output. The concatenated heads pass through an output projection and layer normalization (LN), $\mathbf{z} = \text{LN}(\mathbf{q} + \mathbf{W}^O[\mathbf{z}_1; \ldots; \mathbf{z}_M])$, followed by a feed-forward network (FFN), $\mathbf{t} = \text{LN}(\mathbf{z} + \text{FFN}(\mathbf{z}))$. Here, $\mathbf{W}^O \in \mathbb{R}^{d \times M d_h}$ and $[\,;\,]$ denotes vector concatenation. The resulting task representation $\mathbf{t} \in \mathbb{R}^d$ is reused by the task-level routers for that example.

The router needs one global task state rather than a contextualized representation at every source position. The learned query therefore concentrates capacity on evidence that changes expert selection while avoiding unnecessary source-to-source interactions, which improves both routing efficiency and robustness to long prompts.

For M heads, the cross-attention map contains ML entries rather than the ML^2 entries required by source self-attention. Its attention-score and value-aggregation cost is $\mathcal{O}(Ld)$ with $\mathcal{O}(ML)$ attention memory, compared with $\mathcal{O}(L^2 d)$ computation and $\mathcal{O}(ML^2)$ attention memory for a Transformer task encoder. Including dense projections, the respective total costs are $\mathcal{O}(Ld^2 + Ld + d^2)$ and $\mathcal{O}(Ld^2 + L^2 d)$. The design removes the quadratic interaction term and lowers the source-length dependence from quadratic to linear while preserving a content-dependent global summary.

3.2 ANNEALED TASK-TO-SOFT ROUTING

Task-level routing promotes coherent expert selection, while token-level routing captures local adaptation needs. For token position j, let $\gamma^{\text{task}} \in \mathbb{R}^K$ and $\gamma_j^{\text{token}} \in \mathbb{R}^K$ denote the router logits computed from $\mathbf{t}$ and $\mathbf{h}_j$, respectively. Their normalized outputs are $\boldsymbol{\pi}^{\text{task}} = \text{softmax}(\gamma^{\text{task}})$ and $\boldsymbol{\pi}_j^{\text{token}} = \text{softmax}(\gamma_j^{\text{token}})$. For the current routed layer, a learned coefficient $\eta \in [0, 1]$ combines them as $\boldsymbol{\pi}_j = \eta\boldsymbol{\pi}^{\text{task}} + (1-\eta)\boldsymbol{\pi}_j^{\text{token}}$. Here, $\boldsymbol{\pi}_j \in \mathbb{R}^K$ is the learned distribution over all $K = K_t + K_s$ experts, where K_s is the number of shared experts. Its normalized task-specific slice is $\boldsymbol{\rho}_j = \boldsymbol{\pi}_{j,1:K_t} / \sum_{i=1}^{K_t} \pi_{j,i} \in \mathbb{R}^{K_t}$.

Early in training, unstructured routing can mix unrelated task updates before experts specialize. To provide a coherent starting assignment, CAST-LoRA interpolates between cluster guidance and learned routing at optimizer step s:

$$\widehat{\boldsymbol{\rho}}_{j,s} = (1-\beta_s)\mathbf{e}_{c(\mathbf{x})} + \beta_s\boldsymbol{\rho}_j, \qquad \beta_s = \min\left(1, \frac{s}{S_{\text{ann}}}\right), \tag{3}$$

where $\mathbf{e}_{c(\mathbf{x})} \in \{0,1\}^{K_t}$ is the one-hot vector for semantic cluster $\mathcal{C}_{c(\mathbf{x})}$, β_s is the annealing coefficient, and S_{ann} is the transition length. The main configuration sets $S_{\text{ann}} = 1000$ optimizer steps.

Eq. (3) initially assigns related examples to the same expert, then transfers control linearly to learned gates. After S_{ann} steps, routing is fully input-dependent, inference directly uses $\boldsymbol{\rho}_j$ without cluster labels. This gradual transition limits early task mixing while allowing specialization beyond the training clusters, as examined in Section 4.3. Without cluster supervision, learning the router from the start, when untrained experts are nearly homogeneous and routing decisions are highly random, can produce unstable early assignments and leave some experts with too few examples to specialize. Cluster supervision provides a stable initial assignment, allowing the router and experts to learn more consistently before the learned gate takes full control.

3.3 TASK-SPECIFIC AND SHARED EXPERTS

In a flat pool, task-specific and reusable directions compete for the same sparse routing slots. CAST-LoRA separates these roles by partitioning the bank into $\mathcal{E}_{\text{task}} = \{E_i^{\text{task}}\}_{i=1}^{K_t}$ and $\mathcal{E}_{\text{shared}} = \{E_u^{\text{shared}}\}_{u=1}^{K_s}$. The main configuration uses $K_t = 6$ task-specific experts and $K_s = 2$ shared experts. For one routed linear module, the input-dependent low-rank update at token j is

$$\Delta\mathbf{W}_j = \sum_{i=1}^{K_t} g_{j,i}\Delta\mathbf{W}_i^{\text{task}} + \sum_{u=1}^{K_s} g_{j,K_t+u}\Delta\mathbf{W}_u^{\text{shared}}. \tag{4}$$

Task experts separate cluster updates. Shared experts capture common knowledge and uncertainty.

To keep shared experts active, their gates are protected outside task-specific TOP-K_{act} selection. Let $\boldsymbol{\tau}_j \in \mathbb{R}^{K_t}$ be the normalized task-specific gate obtained by sparsifying $\widehat{\boldsymbol{\rho}}_{j,s}$ during training or $\boldsymbol{\rho}_j$ during inference. Shared expert u uses floor $\delta > 0$ and the same transition horizon:

$$\bar{\omega}_{j,u} = \max(\pi_{j,K_t+u}, \delta), \qquad \omega_{j,u} = \delta + \beta_s\big(\bar{\omega}_{j,u} - \delta\big). \tag{5}$$

The second expression is used during training. At inference, $\omega_{j,u} = \bar{\omega}_{j,u}$. Shared mass is not interpolated with the cluster one-hot vector. The floor is $\delta = 0.02$.

The total shared mass is $\Omega_j = \min(\sum_{u=1}^{K_s} \omega_{j,u}, 1-\zeta)$, where $\zeta = 10^{-6}$ reserves capacity for task-specific experts. The unnormalized gates are $\widetilde{g}_{j,i} = (1-\Omega_j)\tau_{j,i}$ for task expert i and $\widetilde{g}_{j,K_t+u} = \omega_{j,u}$ for shared expert u. The final gate is $\mathbf{g}_j = \widetilde{\mathbf{g}}_j / \sum_{k=1}^{K} \widetilde{g}_{j,k}$. This construction combines sparse task selection with persistent shared access. Protecting shared routes outside task TOP-K_{act} selection prevents reusable branches from being starved by locally dominant task experts. Separating shared and task-specific capacity also discourages each task expert from relearning common transformations, leaving more capacity for distinct task directions.

The objective below gives these branches distinct entropy and floor roles, evaluated in Section 4.4.

The mixture adapts query, key, value, gate, and up projections. Attention output and down projections use ordinary LoRA. Parameter-sharing details appear in Appendix B.

3.4 Training Objective

All trainable components are optimized jointly with Θ frozen. The autoregressive loss is $\mathcal{L}_{\mathrm{LM}} = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \sum_{j=1}^{|\mathbf{y}|} \log p(y_j \mid \mathbf{x}, \mathbf{y}_{<j}; \Theta, \Phi)$.

Two additional losses make the intended routing roles explicit. Let $\mathcal{P}$ index all valid non-padding routing locations across examples, tokens, and routed layers. At location $\xi \in \mathcal{P}$, define the normalized task-specific gate as $\bar{g}^{\mathrm{task}}_{\xi,i} = g_{\xi,i} / \sum_{i'=1}^{K_t} g_{\xi,i'}$. Its normalized entropy is $\mathcal{U}_\xi = -(\log K_t)^{-1} \sum_{i=1}^{K_t} \bar{g}^{\mathrm{task}}_{\xi,i} \log(\bar{g}^{\mathrm{task}}_{\xi,i} + \epsilon)$, where $\epsilon > 0$ provides numerical stability. In the main two-shared-expert configuration, $g^{\mathrm{ent}}_\xi = g_{\xi,K_t+1}$ and $g^{\mathrm{floor}}_\xi = g_{\xi,K_t+2}$ denote the entropy-aligned and floor-preserved shared gates. Their regularizer is

$$\mathcal{L}_{\mathrm{shared}} = \frac{1}{|\mathcal{P}|} \sum_{\xi\in\mathcal{P}} \left[\left(g^{\mathrm{ent}}_\xi - \mathrm{sg}(\mathcal{U}_\xi) \right)^2 + \max\left(0, \delta - g^{\mathrm{floor}}_\xi\right)^2 \right]. \tag{6}$$

Entropy alignment encourages shared access under uncertain routing. Stop-gradient prevents adjustment of the target entropy. Floor preservation maintains a second path for reusable updates.

Training clusters also supervise the task-level router directly. For a sampled example, let $\boldsymbol{\chi}^{(\ell)} \in \mathbb{R}^{K_t}$ be the normalized first K_t entries of the task-level distribution at routed layer ℓ, and let Λ be the set of such layers. The auxiliary task-router loss is $\mathcal{L}_{\mathrm{aux}} = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} |\Lambda|^{-1} \sum_{\ell\in\Lambda} \log \chi^{(\ell)}_{c(\mathbf{x})}$. This term trains the soft router even while cluster guidance dominates the forward gate. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{shared}} \mathcal{L}_{\mathrm{shared}} + \lambda_{\mathrm{aux}} \mathcal{L}_{\mathrm{aux}}, \tag{7}$$

with $\lambda_{\mathrm{shared}} = 0.001$ and $\lambda_{\mathrm{aux}} = 0.01$ in the main configuration. Both auxiliary terms are used only during training and add no inference-time supervision requirement.

4 Experiments

4.1 Experimental Setup

Base model and training data. Qwen2-1.5B is used as the frozen backbone (Yang et al., 2024). Fine-tuning uses the processed FLAN-10k-flat split, a FLAN-style source–target instruction mixture derived from FLAN v2 (Chung et al., 2024; Longpre et al., 2023). The processed split contains 10,908,879 examples from 1,820 raw tasks. Six semantic clusters are constructed offline from training prompts and used only as a training-time routing scaffold. Evaluation examples never provide cluster identifiers to CAST-LoRA. Appendix B details data preprocessing and cluster construction.

Evaluation. Evaluation uses Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) and MMLU-Pro (Wang et al., 2024). It also includes AI2 Reasoning Challenge (ARC)-Challenge, ARC-Easy (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). The final two benchmarks are Physical Interaction: Question Answering (PIQA) (Bisk et al., 2020) and CommonsenseQA (Talmor et al., 2019). For MMLU and MMLU-Pro, the reported metric is subject-macro accuracy. For the remaining datasets, the reported metric is accuracy. All main scores are means over 5 seeds, and Avg is the unweighted mean across the seven datasets.

Baselines. The main comparison contains ten baselines plus CAST-LoRA. Dense and PEFT baselines are full fine-tuning, LoRA (Hu et al., 2022), LoRA+ (Hayou et al., 2024), and PiSSA (Meng et al., 2024). Mixture-based baselines are MoA (Feng et al., 2024), MixLoRA (Li et al., 2024), LoRAMoE (Dou et al., 2024), FlyLoRA (Zou et al., 2025), HotMoE (Huang et al., 2026), and HMoRA-LW (Liao et al., 2025). Appendix A gives the detailed baseline descriptions and inference assumptions. Optimization, adapter, and diagnostic-checkpoint settings are reported in Appendix B.

4.2 Main Results

Table 1 compares CAST-LoRA with ten baselines across seven benchmarks.

Table 1: Multi-seed mean accuracy on seven benchmarks and their unweighted average. Best and second-best results are bolded and underlined, respectively.

Method	MMLU	MMLU-Pro	ARC-C	ARC-E	OBQA	PIQA	CSQA	Avg
Full fine-tuning	50.99	20.86	61.11	78.19	62.75	70.87	66.52	58.75
LoRA	55.16	22.28	68.53	84.55	67.28	67.43	68.55	61.97
LoRA+	<u>55.51</u>	22.08	68.70	84.68	67.24	71.52	69.12	62.70
PiSSA	53.80	22.42	66.00	83.91	66.72	70.58	68.33	61.69
MoA	53.07	23.79	66.97	84.01	64.60	74.42	65.33	61.74
MixLoRA	55.44	**26.26**	<u>69.45</u>	84.55	**70.00**	73.88	71.00	<u>64.36</u>
LoRAMoE	54.19	24.52	68.34	84.63	67.40	72.85	67.10	62.72
FlyLoRA	54.78	23.82	68.97	<u>85.47</u>	65.20	<u>74.98</u>	69.63	63.26
HotMoE	55.38	23.80	66.97	85.18	68.88	72.88	69.71	63.25
HMoRA-LW	54.94	24.15	66.53	83.08	63.92	74.10	<u>71.23</u>	62.56
CAST-LoRA (Ours)	**56.15**	<u>24.98</u>	**70.56**	**86.20**	<u>69.60</u>	**75.30**	**71.89**	**64.95**

Table 2: Projected-gradient diagnostics on 276 task pairs. Negative pairs count cosines below zero, and Conflict gives their percentage. Cluster gap is within-cluster minus between-cluster mean cosine. Arrows indicate preferred directions. Best and second-best results are bolded and underlined.

Method	Negative pairs ↓	Conflict (%) ↓	Mean cos. ↑	Cluster gap ↑
LoRA	18/276	6.52	0.176	0.057
LoRA+	22/276	7.97	0.186	0.062
PiSSA	34/276	12.32	0.122	0.054
MixLoRA	<u>14/276</u>	<u>5.07</u>	<u>0.264</u>	0.066
FlyLoRA	<u>14/276</u>	<u>5.07</u>	0.191	<u>0.070</u>
CAST-LoRA (Ours)	**3/276**	**1.09**	**0.302**	**0.084**

Table 3: Accuracy and final raw-task routing consistency for three routing variants. Consistency is the task-averaged fraction of 96 probe samples assigned to their raw task's modal TOP-1 expert. Higher values indicate more stable task-to-expert specialization. The best and second-best entries in each column are bolded and underlined, respectively.

Variant	MMLU	MMLU-Pro	ARC-C	ARC-E	OBQA	CSQA	Avg	Consistency
CAST-LoRA (Ours)	**56.15**	**24.98**	**70.56**	**86.20**	**69.60**	**71.89**	**63.23**	**79.2**
w/o annealing	55.56	24.32	69.97	<u>86.03</u>	68.80	<u>70.78</u>	<u>62.58</u>	<u>68.8</u>
w/o cluster scaffold	<u>55.72</u>	<u>24.87</u>	<u>70.14</u>	85.19	<u>69.20</u>	69.78	62.48	64.6

CAST-LoRA achieves the highest average accuracy of 64.95, exceeding MixLoRA and HMoRA-LW by 0.59 and 2.39 points, respectively. It ranks first on five benchmarks and second on two, showing consistent gains across knowledge and reasoning tasks. This broad performance is consistent with task-specific experts separating adaptation directions while shared experts retain reusable information. The following diagnostics examine how routing and role separation support these gains.

4.3 Task Interference and Annealed Routing

Gradient conflict and routing ablations. To test whether CAST-LoRA reduces task interference, pairwise gradient cosine is measured for 24 training tasks, using four examples per task. Each token-normalized loss gradient over trainable parameters is compressed into an 8,192-dimensional CountSketch vector. Negative cosine indicates opposing updates. In Table 2, CAST-LoRA achieves the lowest conflict rate, 1.09%, a 78.5% reduction from the best baseline value of 5.07%. Its mean cosine and cluster gap improve by 14.4% and 20.0%, respectively, indicating stronger update alignment and clearer cluster separation. Conflict also decreases from 3.99% to 1.09% during training (Appendix E), showing that specialization becomes more compatible as routing matures.

Table 3 tests the contributions of annealing and cluster guidance. The full model leads all six reported benchmarks, with 63.23 average accuracy and 79.2% routing consistency. Removing annealing

Table 4: Task-expert pair consistency on MMLU subjects. *Standard* prompts include the subject name, whereas *Label-free* prompts omit it. Each column reports the percentage of subjects whose modal TOP-2 expert pair covers at least 70%, 80%, or 90% of 100 prompts. Higher percentages indicate more stable subject-level routing and less reliance on explicit task names.

Prompt form	$\tau \geq 0.7$	$\tau \geq 0.8$	$\tau \geq 0.9$
Standard	77.2%	59.6%	42.1%
Label-free	**80.7%**	**63.2%**	**45.6%**

Table 5: Shared-expert regularization ablation. Entropy Reg. and Floor Reg. mark the two role-specific losses. Changes in Avg indicate how each loss helps maintain effective shared capacity. The best and second-best entries in each column are bolded and underlined, respectively.

Variant	Entropy Reg.	Floor Reg.	MMLU	ARC-E	OBQA	CSQA	Avg	Δ
CAST-LoRA (Ours)	✓	✓	**56.15**	**86.20**	**69.60**	**71.89**	**70.96**	–
w/o entropy alignment	×	✓	55.51	85.69	68.60	70.44	70.06	−0.90
w/o floor preservation	✓	×	<u>55.88</u>	<u>86.07</u>	68.40	<u>71.56</u>	70.48	−0.48
w/o both shared losses	×	×	55.84	85.82	<u>69.20</u>	71.33	<u>70.55</u>	−0.41

Table 6: Shared-expert removal and role interventions. Masking removes one route, $X \to Y$ transfers its mass, and gate/B swaps exchange routing weights/parameters. Mass records shared-mass preservation, and Δ is the Avg change relative to CAST-LoRA. Performance drops indicate branch contribution, while mass-preserving changes test role interchangeability. Best scores are bolded.

Condition	Mass	MMLU	ARC-E	OBQA	PIQA	CSQA	Avg	Δ
CAST-LoRA (Ours)	–	56.15	**86.20**	**69.60**	**75.30**	**71.89**	**71.83**	–
w/o shared experts	N/A	**56.55**	85.01	68.60	75.19	70.66	71.20	−0.63
Mask entropy-aligned	No	56.19	85.77	68.20	75.08	70.89	71.23	−0.60
Mask floor-preserved	No	55.05	84.34	66.40	69.91	69.44	69.03	−2.80
Entropy → floor	Yes	56.30	85.82	68.00	**75.30**	71.67	71.42	−0.41
Floor → entropy	Yes	54.08	83.46	67.80	69.75	68.22	68.66	−3.17
Gate swap	Yes	54.01	83.88	67.20	70.08	68.78	68.79	−3.04
B swap	Yes	53.93	83.88	67.80	69.70	68.33	68.73	−3.10

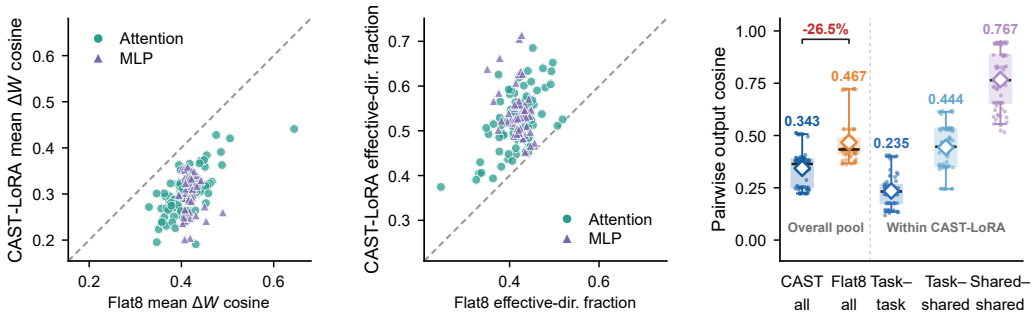
reduces accuracy by 0.65 points and consistency by 10.4 percentage points. Removing cluster guidance costs 0.75 accuracy points and 14.6 percentage points of consistency. Cluster guidance gives related tasks coherent initial experts, while annealing transfers control gradually to learned gates. These complementary steps stabilize task–expert assignments, supporting the best routing-variant accuracy and the lowest measured gradient conflict.

Final expert-pair consistency. Removing subject names tests whether stable expert selection depends on explicit task labels (Table 4). Without these names, 63.2% of MMLU subjects use one expert pair for at least 80% of prompts, compared with 59.6% under standard prompts. Consistency remains higher at all three thresholds, supporting stable routing from prompt content alone.

4.4 Role-Separated Shared Experts

Loss ablations, branch interventions, and a matched Flat8-TOP4 comparison test whether shared experts learn useful, distinct functions. Flat8-TOP4 matches the eight experts, trainable-parameter budget, and four active branches per token to isolate expert organization.

Shared-loss ablation. Table 5 ablates the two terms in Eq. (6). Removing entropy alignment or floor preservation lowers the four-benchmark average by 0.90 or 0.48 points, respectively, from 70.96. Entropy alignment links shared access to routing uncertainty, while floor preservation maintains a persistent path for common updates. The full objective achieves the best accuracy on all four benchmarks, supporting both forms of shared-capacity allocation.



(a) **(b)** **(c)**

Figure 2: Expert redundancy under matched budgets. **(a)** Mean pairwise update cosine and **(b)** effective-direction fraction for each routed module. Points below/above the respective equality lines favor CAST-LoRA. **(c)** Expert-output cosine before gate weighting, comparing overall pools and task/shared pair types within CAST-LoRA.

Shared-expert ablation and interventions. Table 6 tests branch utility by masking shared routes and tests interchangeability by transferring routing mass or swapping gates and expert parameters. A separately trained model without shared experts loses 0.63 average-accuracy points across five benchmarks. Masking the floor-preserved branch lowers every benchmark and costs 2.80 points, while transferring its mass to the entropy-aligned branch costs 3.17 points (4.4%). This broad causal utility is consistent with a persistent shared path accumulating reusable updates across tasks. Gate and B swaps preserve shared mass but reduce accuracy by 3.04 and 3.10 points, showing that routing must match expert content. The full model retains the highest average accuracy, with complementary shared functions that cannot simply replace one another. The only exception is MMLU, where the separately trained model without shared experts is 0.40 points higher. A mechanism consistent with this result is that MMLU spans many sharply separated knowledge subjects: removing shared branches concentrates all active routed updates on task-specific experts, which can sharpen subject-specific memorization, whereas shared transformations introduce a small amount of cross-task smoothing. The same change removes broadly reusable features and lowers every other score in Table 6, so the isolated MMLU gain reflects a specialization–sharing trade-off rather than stronger overall routing.

Structured expert organization and redundancy. Figure 2 compares redundancy with Flat8-TOP4 across 140 matched modules. Pairwise Frobenius cosine between normalized updates $\Delta\mathbf{W} = \mathbf{BA}$ measures repeated parameter directions. For Gram eigenvalues λ_i, the effective-direction fraction is $(\sum_i \lambda_i)^2/(8\sum_i \lambda_i^2)$. Larger values indicate more distinct directions. Pairwise cosine between expert outputs before gate weighting measures functional overlap on actual inputs.

CAST-LoRA lowers update cosine in all 140 modules, reducing its mean by 26.8% (0.414 to 0.303) and increasing the effective-direction fraction by 26.8%. Output overlap also falls by 26.5% (0.467 to 0.343), confirming that the more distinct parameter directions yield less similar transformations. Within CAST-LoRA, output overlap is lowest for task–task pairs (0.235), intermediate for task–shared pairs (0.444), and highest for shared–shared pairs (0.767). This ordering supports distinct task-specific directions with common transformations concentrated in shared paths. Average accuracy remains comparable and slightly higher on the eight-dataset comparison suite (64.59 versus 64.33). Thus, explicit role separation reduces redundancy and clarifies expert functions while preserving predictive performance. Additional geometry statistics appear in Appendix D.

4.5 Task Encoder Efficiency

Scaling experiments select six task-specific experts, 1,000 annealing steps, and one task encoder as the best-performing configuration (Appendix C).

To evaluate routing efficiency, Figure 3 measures isolated encoder forward passes with a matched parameter count of 18.89M. The single-query encoder avoids quadratic token–token interactions, yielding lower latency and peak memory than the one-layer Transformer, especially for larger batches and longer inputs. At $B = 16$ and $L = 4096$, latency falls by 86.9% (38.28 to 5.01 ms) and peak

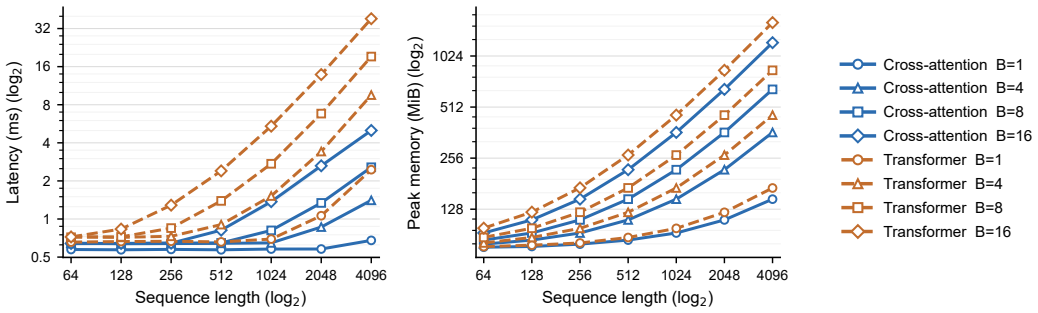


Figure 3: Matched-parameter task-encoder efficiency. **(a)** Median CUDA-event per-forward latency. **(b)** Median peak CUDA memory allocated. Both panels use $\log_2$ axes and evaluate one-layer, $d = 1536$ encoders across $B \in \{1, 4, 8, 16\}$ and $L \in \{64, 128, 256, 512, 1024, 2048, 4096\}$.

memory by 23.8% (1,611.7 to 1,227.4 MiB). Average accuracy also increases by 0.52 points to 64.95, giving the cross-attentive encoder the best accuracy, latency, and memory usage of the two designs.

5 Related Work

Conditional experts and low-rank adaptation. Adaptive Mixtures of Local Experts (Jacobs et al., 1991) gates inputs to expert networks for improved prediction. Sparsely-Gated MoE (Shazeer et al., 2017) selects few experts to scale efficiently, while Switch Transformer (Fedus et al., 2022) uses TOP-1 routing to simplify sparse computation. LoRA (Hu et al., 2022) trains low-rank updates to reduce adaptation cost, while AdaLoRA (Zhang et al., 2023) reallocates rank to improve low-budget tuning. LoRA+ (Hayou et al., 2024) uses factor-specific rates to accelerate optimization at unchanged computational cost. FAMO (Liu et al., 2023) uses loss history for constant-cost balancing.

Mixtures of LoRA experts. MoLoRA (Zadouri et al., 2024) sparsely routes adapters, matching full tuning while updating under 1% of parameters. MoA (Feng et al., 2024) routes domain LoRAs to reduce interference, while MixLoRA (Li et al., 2024) balances TOP-K experts to improve accuracy. LoRAMoE (Dou et al., 2024) separates adaptation and knowledge experts to reduce forgetting, while MoELoRA (Luo et al., 2024) contrasts experts to improve routing and reasoning. HydraLoRA (Tian et al., 2024) shares a down-projection across routed up-projections to separate common and task-specific knowledge. HMoRA (Liao et al., 2025) combines task- and token-level routing with an auxiliary loss to improve specialization and unseen-task transfer. FlyLoRA (Zou et al., 2025) uses rank-wise experts and frozen random projections to reduce interference without routers. HotMoE (Huang et al., 2026) combines lower-layer expert attention with upper-layer token routing to improve collaboration and specialization. I^2MoE (Xin et al., 2025) reweights weakly supervised interaction experts to improve fusion and interpretability.

6 Conclusion

CAST-LoRA is a parameter-efficient multi-task MoE-LoRA framework that unifies three complementary designs. Its single-query cross-attentive task encoder compresses global evidence with $\mathcal{O}(Ld)$ attention interactions, where L is sequence length and d is hidden dimension. Annealed task-to-soft routing uses semantic clusters as temporary training guidance before transferring control to label-free, input-dependent routing. The role-separated expert bank reserves distinct capacity for task specialization, routing uncertainty, and reusable cross-task knowledge. Across 10 baselines and 7 benchmarks, CAST-LoRA achieves the optimal average accuracy, reduces gradient conflict to an optimal level of 1.09%, cuts task-encoder latency by 86.9%, cuts task-encoder peak memory by 23.8%, lowers expert-parameter redundancy by 26.8%, and increases effective expert directions by 26.8%. Together, these results show that structured routing and explicit shared capacity can jointly preserve accuracy, reduce interference, and avoid redundant expert specialization.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript writing and language polishing.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432–7439, 2020.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 794–803, 2018.

Hyung Won Chung, Le Hou, Shayne Longpre, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25:1–53, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, pp. 10088–10115, 2023.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 1932–1945, 2024.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 5547–5569, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:1–39, 2022.

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*, pp. 11371–11380, 2024.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 27503–27516, 2021.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 17783–17806, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

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

Yitong Huang, Ziqi Yang, Zihui Wang, Jianzhong Qi, Rongshan Yu, Xiaoliang Fan, and Cheng Wang. Hybrid routing for a mixture of LoRA experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 31211–31219, 2026.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6265–6274, 2021.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.

Mengqi Liao, Wei Chen, Junfeng Shen, Shengnan Guo, and Huaiyu Wan. HMoRA: Making LLMs more effective with hierarchical mixture of LoRA experts. In *International Conference on Learning Representations*, 2025.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 18878–18890, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems*, pp. 57226–57243, 2023.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, pp. 1950–1965, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32100–32121, 2024.

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 *Proceedings of the 40th International Conference on Machine Learning*, pp. 22631–22648, 2023.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. MoELoRA: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*, 2024.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, pp. 121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 525–536, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9120–9132, 2020.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4149–4158, 2019.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. HydraLoRA: An asymmetric LoRA architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*, pp. 9565–9584, 2024.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, pp. 95266–95290, 2024.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

Jiayi Xin, Sukwon Yun, Jie Peng, Inyoung Choi, Jenna L. Ballard, Tianlong Chen, and Qi Long. I^2MoE: Interpretable multimodal interaction-aware mixture-of-experts. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 68870–68888, 2025.

An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 5824–5836, 2020.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient MoE for instruction tuning. In *International Conference on Learning Representations*, 2024.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34:5586–5609, 2022.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. FlyLoRA: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. In *Advances in Neural Information Processing Systems*, pp. 11540–11573, 2025.

SUPPLEMENTARY MATERIAL

This supplementary material provides the implementation details, reproducibility specifications, and additional experimental evidence omitted from the main paper because of space constraints. It complements the main results as follows:

- **Appendix A** describes the training mixture, seven evaluation benchmarks, exact multiple-choice scoring protocol, comparison methods, and verified baseline configurations.
- **Appendix B** specifies clustering, optimization, adapter placement, hyperparameters, the training and inference algorithm, tensor shapes, complexity, and parameter counts.
- **Appendices C–H** report scaling, expert geometry, gradient-conflict trajectories, annealing sensitivity, routing-transition diagnostics, and shared-expert interventions.
- **Appendix I** provides an expanded discussion of parameter-efficient adaptation, mixtures of LoRA experts, and multi-task sharing.

A ADDITIONAL EXPERIMENTAL DETAILS

A.1 TRAINING MIXTURE AND EVALUATION BENCHMARKS

Training mixture. The processed FLAN-10k-flat source–target mixture derives from FLAN v2 (Chung et al., 2024; Longpre et al., 2023) and contains 10,908,879 records with 1,820 raw task names. Each record supplies an instruction-formatted source and target. Appendix B describes how six cluster identifiers are computed from training prompts. They supervise training but are not evaluation labels. The seven benchmarks are evaluated separately from this mixture.

Knowledge and reasoning benchmarks. MMLU (Hendrycks et al., 2021) tests multiple-choice knowledge across 57 subjects. MMLU-Pro (Wang et al., 2024) uses more difficult questions and a larger option set to test reasoning and knowledge under a less guessable choice format. ARC-Challenge and ARC-Easy (Clark et al., 2018) are the difficult and easier subsets of grade-school science questions, respectively. OpenBookQA (Mihaylov et al., 2018) asks science questions that require combining a small collection of core facts with broader knowledge.

Commonsense benchmarks and aggregation. PIQA (Bisk et al., 2020) measures physical commonsense by asking which of two solutions better achieves a goal. CommonsenseQA (Talmor et al., 2019) measures concept-level commonsense through questions with plausible competing choices. All seven evaluations use the same zero-shot multiple-choice scoring protocol and Qwen chat prompt format across methods, using the stored test splits and final 10,000-step checkpoints. MMLU and MMLU-Pro report the mean of subject accuracies. ARC-Challenge, ARC-Easy, OpenBookQA, PIQA, and CommonsenseQA report example-level accuracy. The headline average is the unweighted mean of these seven dataset-level scores. SWAG and other auxiliary suites are excluded from it.

Exact multiple-choice scoring. For example n, let $\mathcal{V}_n$ be its valid answer-label tokens and let $z_{n,v}$ be the final-position logit for label v after applying the Qwen chat template. Invalid labels are masked and the prediction is $\widehat{y}_n = \arg\max_{v \in \mathcal{V}_n} z_{n,v}$. Thus, evaluation scores the single answer-label token rather than generating an answer string or length-normalizing multi-token candidates. MMLU and MMLU-Pro first compute accuracy within each subject and then average over subjects. For every other benchmark, accuracy is computed over examples. If a_d denotes the resulting score for dataset d, the reported aggregate is $\mathrm{Avg} = 7^{-1} \sum_{d=1}^{7} a_d$.

A.2 COMPARISON METHODS

Dense and single-adapter methods. Full fine-tuning updates all Qwen2-1.5B backbone parameters. LoRA (Hu et al., 2022) freezes the backbone and trains one rank-8 low-rank update per adapted projection. LoRA+ (Hayou et al., 2024) keeps the LoRA architecture but applies a higher learning-rate multiplier to its B factor. PiSSA (Meng et al., 2024) initializes the low-rank factors from principal singular components of the pretrained weight. The local comparison uses the iterative PiSSA initialization. These three adapter baselines do not use a mixture router.

Table 7: Verified comparison configurations under the common Qwen2-1.5B training and evaluation interface. "All" denotes the seven linear projections $q, k, v, o, \mathrm{gate}, \mathrm{up}, \mathrm{down}$. "Routed 5" denotes $q, k, v, \mathrm{gate}, \mathrm{up}$ with ordinary LoRA on o and down.

Method	Rank	Experts	TOP-K	Route	Placement	Method-specific setting
Full fine-tuning	–	–	–	None	All parameters	Dense backbone update
LoRA	8	1	–	None	All	Single adapter
LoRA+	8	1	–	None	All	Larger learning rate for B
PiSSA	8	1	–	None	All	`pissa_niter_4` initialization
MoA	8	6	1	Sequence	All	Cluster-associated experts and selector
MixLoRA	8	8	2	Token	FFN MoE + attention LoRA	Sparse FFN expert routing
LoRAMoE	8	8	2	Token	All	Load-balancing objective
FlyLoRA	8	–	2 ranks	Token	All	Rank-wise routing
HotMoE	8	8	2	Layer/token	All	Lower/upper layer split $= 0.5$
HMoRA-LW	8	8	2	Task+token	Routed 5	Lightweight hierarchical router
CAST-LoRA (Ours)	8	$6+2$	2 task	Task+token	Routed 5	1,000-step transition, protected shared routes

Task-selected and token-routed experts. MoA (Feng et al., 2024) trains one cluster-associated LoRA expert per training cluster and learns a sequence-level selector, whose prediction replaces cluster labels at inference. MixLoRA (Li et al., 2024) uses token-level TOP-2 routing over eight LoRA experts. LoRAMoE (Dou et al., 2024) also routes tokens over a LoRA expert bank and adds a load-balancing objective. The local Qwen2-compatible MixLoRA and LoRAMoE comparisons implement these method families under the common data and evaluation interface. They are not copies of the original repositories.

Other conditional adapters. FlyLoRA (Zou et al., 2025) routes over low-rank components rather than selecting whole LoRA experts, and its local comparison uses TOP-2 rank routing. HotMoE (Huang et al., 2026) varies expert interaction with layer depth, using global expert mixing in lower layers and token-level expert routing in upper layers in the local implementation. HMoRA-LW (Liao et al., 2025) is the lightweight HMoRA comparison, retaining hierarchical task- and token-level routing without the proposed annealed cluster-to-soft transition or role-separated shared regularizer. Unless a method inherently uses training clusters, comparisons use the same source–target data without task labels. No method receives true benchmark labels or cluster identifiers at inference.

B Implementation Details

Data preprocessing and semantic clustering. The processed FLAN-10k-flat record contains 10,908,879 examples and 1,820 raw task names. The six training clusters are constructed from the training split only. For each raw task, up to 64 source prompts are reservoir-sampled and embedded with `sentence-transformers/all-MiniLM-L6-v2` using attention-mask mean pooling and a maximum length of 256. Prompt embeddings are ℓ_2-normalized before averaging, and each resulting raw-task centroid is normalized again. Spherical k-means with $k = 6$ uses multiple seeds, initializes centroids from six distinct raw tasks, and runs for at most 100 iterations or until assignments stop changing. If a cluster becomes empty, the raw task least similar to its nearest current centroid is reassigned to it before centroids are recomputed. Cluster labels are canonicalized by descending example count, with the lexicographically first member used to break ties, and then written back as `task=cluster_0`, ..., `cluster_5`, while the original task name is retained as `raw_task`. Cluster identifiers supervise training-time routing only.

Optimization. The fixed-step experiment configuration trains for 10,000 optimizer steps using AdamW with a learning rate of 5×10^{-5} and 500 warm-up steps, followed by a constant learning rate. Weight decay, dropout, and label smoothing are set to 10^{-2}, 0.1, and 0.1, respectively. The per-device batch size is 1 with eight gradient-accumulation steps, yielding an effective batch size of 8 on one device. The input and output length limits are 1024 and 512, and training uses bfloat16.

Adapter configuration. The routed mixture adapts query, key, value, gate, and up projections. Attention output and down projections use ordinary LoRA adapters. This placement concentrates conditional capacity in feature selection and expansion while limiting routing overhead in output projections. Router sharing and Hydra-style low-rank parameter sharing reduce trainable and routing costs (Liao et al., 2025; Tian et al., 2024). Unless specified otherwise, all LoRA experts use rank

Table 8: Main training hyperparameters.

Hyperparameter	Value
Backbone / training data	Qwen2-1.5B / FLAN-10k-flat processed split
Data size / structure	10,908,879 source–target pairs; 1,820 raw tasks; 6 clusters
Cluster construction	All-MiniLM-L6-v2 raw-task centroids, spherical k-means
Training schedule	10,000 optimizer steps, final-checkpoint selection
Optimizer / precision	AdamW / bfloat16
Learning rate / warm-up / decay	5×10^{-5} / 500 steps / 10^{-2}
Dropout / label smoothing	0.1 / 0.1
Training batch	1 per device; 8 accumulation steps; effective batch size 8
Evaluation batch / max lengths	4 / 1024 input tokens / 512 output tokens
LoRA rank / alpha	8 / 8
Experts / TOP-K / roles	8 / 2 / 6 task-specific + 2 shared
Annealing steps / shared gate floor	1000 / 0.02
λ_{shared} / λ_{aux}	0.001 / 0.01 (task-router CE)

Table 9: Principal symbols and tensor shapes. $K = K_t + K_s = 8$ in the main model.

Symbol	Shape	Main value	Description
$\mathbf{T}^{\text{in}}$	$B \times L \times d$	–	Source-token embeddings supplied to the task encoder
$\mathbf{t}$	$B \times d$	–	Single-query task state
$\boldsymbol{\pi}_j, \mathbf{g}_j$	$B \times L \times K$	$K = 8$	Dense learned distribution and final expert gate
$\boldsymbol{\rho}_j, \boldsymbol{\tau}_j$	$B \times L \times K_t$	$K_t = 6$	Dense and TOP-K task-expert gates
$\mathbf{A}$	$r \times d_{\text{in}}$	$r = 8$	Shared Hydra-style down projection
$\mathbf{B}_i$	$d_{\text{out}} \times r$	$i = 1, \ldots, 8$	Expert-specific up projection
K_{act}	scalar	2	Active task-specific experts per token

$r = 8$ and scaling $\alpha = 8$. Mixture methods use eight experts and TOP-2 routing when their original design supports sparse routing. CAST-LoRA uses six task-specific experts and two shared experts. The routing transition lasts 1,000 optimizer steps, and the shared gate floor is 0.02. The auxiliary task-router cross-entropy weight is 0.01 during training. Mechanism diagnostics use the final 10,000-step checkpoint for controlled within-run analysis.

B.1 Training and Inference Algorithm

For a routed layer, let $\mathcal{I}_j = \text{TOPK}(\widehat{\boldsymbol{\rho}}_{j,s}, K_{\text{act}})$ during training and $\mathcal{I}_j = \text{TOPK}(\boldsymbol{\rho}_j, K_{\text{act}})$ during inference. Writing $r_{j,i}$ for the corresponding pre-sparsification task probability, the normalized sparse task gate is

$$\tau_{j,i} = \frac{\mathbb{I}[i \in \mathcal{I}_j] r_{j,i}}{\sum_{i'=1}^{K_t} \mathbb{I}[i' \in \mathcal{I}_j] r_{j,i'} + \epsilon}. \tag{8}$$

The shared gates follow Eq. (5), with $\Omega_j = \min(\sum_u \omega_{j,u}, 1 - 10^{-6})$. Before the final numerical normalization, task and shared gates are $(1 - \Omega_j)\tau_{j,i}$ and $\omega_{j,u}$, respectively. This guarantees nonzero task capacity and $\sum_i g_{j,i} = 1$ up to floating-point precision.

The detailed process of the proposed algorithm (CAST-LoRA) is summarized in Algorithm 1.

B.2 Tensor Shapes, Parameterization, and Complexity

For one routed projection, independent expert LoRAs require $P_{\text{ind}} = Kr(d_{\text{in}} + d_{\text{out}})$ parameters. Sharing $\mathbf{A}$ gives $P_{\text{Hydra}} = rd_{\text{in}} + Krd_{\text{out}}$, saving $(K-1)rd_{\text{in}}$ parameters while retaining an expert-specific $\mathbf{B}_i$. An ordinary LoRA projection requires $r(d_{\text{in}} + d_{\text{out}})$ parameters. The single-query attention readout has $\mathcal{O}(Ld)$ attention interactions, compared with $\mathcal{O}(L^2 d)$ for source self-attention. Both statements exclude their linear-projection costs.

C Scaling Ablations

Figure 4 summarizes three configuration sweeps using the unweighted average over the seven reported benchmarks. Six task-specific experts, 1,000 annealing steps, and one task encoder achieve

Algorithm 1 One CAST-LoRA training or inference pass. Semantic clusters enter only the training branch. Inference is label-free and input-dependent.

Require: Source embeddings $\mathbf{T}^{\text{in}}$, routed states $\{\mathbf{h}_j\}$, step s, and cluster $c(\mathbf{x})$ during training
Ensure: Routed updates $\{\Delta\mathbf{h}_j\}$ and training loss $\mathcal{L}$ when applicable

$\mathbf{t} \leftarrow \text{TaskEncoder}(\mathbf{T}^{\text{in}})$
$\boldsymbol{\pi}^{\text{task}} \leftarrow \text{softmax}(f_{\text{task}}(\mathbf{t}))$
for each routed token j **do**
  $\boldsymbol{\pi}_j^{\text{token}} \leftarrow \text{softmax}(f_{\text{token}}(\mathbf{h}_j))$
  $\boldsymbol{\pi}_j \leftarrow \eta\boldsymbol{\pi}^{\text{task}} + (1-\eta)\boldsymbol{\pi}_j^{\text{token}}$, then normalize its first K_t entries as $\boldsymbol{\rho}_j$
  if training **then**
    $\beta_s \leftarrow \min(1, s/S_{\text{ann}})$ and $\widehat{\boldsymbol{\rho}}_{j,s} \leftarrow (1-\beta_s)\mathbf{e}_{c(\mathbf{x})} + \beta_s\boldsymbol{\rho}_j$ using Eq. (3)
  else
    $\widehat{\boldsymbol{\rho}}_{j,s} \leftarrow \boldsymbol{\rho}_j$
  end if
  $\boldsymbol{\tau}_j \leftarrow \text{TOPKNormalize}(\widehat{\boldsymbol{\rho}}_{j,s}, K_{\text{act}})$ using Eq. (8)
  Compute shared gates $\omega_{j,u}$ using Eq. (5) and set $\Omega_j \leftarrow \min(\sum_u \omega_{j,u}, 1-10^{-6})$
  $\mathbf{g}_j \leftarrow \text{Normalize}([(1-\Omega_j)\boldsymbol{\tau}_j; \boldsymbol{\omega}_j])$
  $\Delta\mathbf{h}_j \leftarrow (\alpha/r)\sum_i g_{j,i}\mathbf{B}_i\mathbf{A}\mathbf{h}_j$
end for
if training **then**
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{LM}} + \lambda_{\text{shared}}\mathcal{L}_{\text{shared}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}$, with the entropy target detached
else
  Discard cluster identifiers and auxiliary losses
end if
return $\{\Delta\mathbf{h}_j\}$ and, during training, $\mathcal{L}$

Table 10: Trainable-parameter composition of the audited main checkpoint. Symbolic counts are per adapted projection or router. The total is measured from the checkpoint.

Component	Trainable count	Role
Qwen2-1.5B backbone	0	Frozen pretrained weights
Cross-attentive encoder	18,892,800	One shared source-sequence readout
Task/token routers	$K(d+1)$ per router	Task-state and token-state logits. Router instances are shared across grouped projections
Shared $\mathbf{A}$ factors	rd_{in} per routed projection	Common low-rank input subspace
Expert $\mathbf{B}_i$ factors	Krd_{out} per routed projection	Six task-specific and two shared outputs
Ordinary LoRA	$r(d_{\text{in}}+d_{\text{out}})$ per projection	Attention-output and down projections
Total	60,344,930	3.76% of 1,604,059,234 parameters

the highest average within their respective tested grids. Increasing the number of task experts or task encoders beyond the selected setting does not improve average performance, confirming that the final configuration is selected by measured effectiveness rather than capacity alone.

D Additional Expert-Geometry Statistics

Table 11 complements Figure 2 with a within-CAST-LoRA decomposition of expert-pair geometry. Because the Hydra-style parameterization shares $\mathbf{A}$ across experts, the parameter comparison uses the realized update $\Delta\mathbf{W} = \mathbf{BA}$ and the principal angle between expert-specific $\mathbf{B}$ column spaces rather than comparing raw factors in isolation. Functional statistics use the original seven-benchmark protocol, differing slightly from the eight-dataset aggregation in Figure 2.

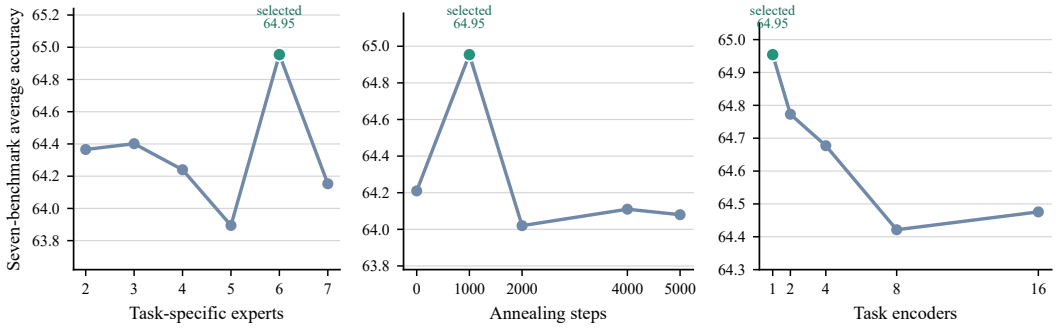


Figure 4: Seven-benchmark scaling ablations over **(a)** task-specific experts, **(b)** annealing steps, and **(c)** task encoders. Panel **(a)** retains two shared experts. Green markers show the selected settings.

Table 11: Within-CAST-LoRA expert-pair geometry at 10,000 steps over 140 routed modules and seven benchmarks. Lower cosine and larger B-subspace angles indicate less overlap.

Pair type	ΔW cos.	B angle	Output cos.
Task–task	0.191	76.0°	0.236
Task–shared	0.409	64.9°	0.444
Shared–shared	0.714	43.0°	0.768

Table 12: Projected-gradient conflict of CAST-LoRA over training on the same 276 task pairs. Decreasing conflict and gradient-norm CV indicate increasingly compatible and balanced task updates, while a higher mean cosine indicates stronger alignment.

Step	Negative pairs	Conflict (%)	Mean cos.	Norm CV
2,500	11	3.99	0.257	0.498
5,000	11	3.99	0.295	0.389
7,500	6	2.17	**0.304**	0.458
10,000	**3**	**1.09**	0.302	**0.381**

Table 13: Sensitivity to the number of annealing steps at the final checkpoint. Higher Avg indicates stronger multi-task accuracy, while lower step time indicates lower training overhead. Δ Avg is measured relative to the 1,000-step setting.

Steps	Avg	Δ Avg	Step (s)
0	64.21	−0.74	**3.71**
1000	**64.95**	**0.00**	4.92
2000	64.02	−0.93	5.14
4000	64.11	−0.84	4.06
5000	64.08	−0.87	4.11

E Training Dynamics of Task Interference

Table 12 traces the projected-gradient diagnostic used in Section 4.3 across four training checkpoints.

F Annealing Schedule Sweep

The current sweep favors a short transition. A 1000-step annealing schedule gives the best average among the tested settings, while removing annealing or extending it to 2000–5000 steps reduces the performance. This suggests that the cluster scaffold is useful early in training, but keeping it influential for too long can slow the move toward fully input-dependent routing.

Table 14: Routing diagnostics for CAST-LoRA and two transition ablations. Avg is the seven-benchmark mean. Lower JSD and higher consistency indicate smoother, more coherent routing.

Variant	Avg	Flip (%)	Adj. JSD	Last step	Consistency (%)
CAST-LoRA (Ours)	**64.89**	14.88	**0.0080**	851.5	**79.17**
w/o annealing	64.42	11.88	0.0101	706.5	68.75
w/o cluster scaffold	64.27	**11.35**	0.0080	**707.0**	64.58

Table 15: Per-benchmark accuracy under shared-expert interventions. Δ is the seven-benchmark average change from the unmodified checkpoint. The entropy-aligned and floor-preserved experts are the first and second shared branches, respectively.

Condition	MMLU	MMLU-P	ARC-C	ARC-E	OBQA	CSQA	PIQA	Avg	Δ
CAST-LoRA (Ours)	56.15	24.98	70.56	86.20	69.60	71.89	75.30	64.95	0.00
Mask entropy-aligned	56.19	25.17	70.56	85.77	68.20	70.89	75.08	64.55	-0.40
Mask floor-preserved	55.05	23.81	67.92	84.34	66.40	69.44	69.91	62.41	-2.54
Entropy mass $\rightarrow$ floor	56.30	25.22	70.90	85.82	68.00	71.67	75.30	64.74	-0.21
Floor mass $\rightarrow$ entropy	54.08	23.49	66.81	83.46	67.80	68.22	69.75	61.94	-3.01

G Routing Transition Diagnostics

The schedule sweep measures final accuracy, whereas Table 14 examines how the learned task routes evolve. The diagnostic follows 96 fixed probe examples across adjacent checkpoints. TOP-1 flip rate is the fraction whose most probable task expert changes between adjacent observations; adjacent JSD is the mean Jensen–Shannon divergence between consecutive task-gate distributions; last fluctuation is the mean final step at which the TOP-1 route changes; and raw-task consistency is the fraction of raw tasks whose probe examples share the same dominant task expert.

The complete transition gives the highest benchmark average and raw-task consistency, while its adjacent JSD is lower than that of direct soft routing without annealing. Its higher TOP-1 flip rate and later final fluctuation show that the scaffold does not freeze assignments early. Routes continue to change as control transfers to the learned router. Thus, the evidence supports smoother routing and more coherent final task groups, rather than consistent gains across all stability measures.

H Shared-Expert Causal Interventions

The interventions in Table 15 are applied after routing at the fixed final checkpoint. For masking, the selected gate is set to zero and all remaining gates are globally renormalized, so active capacity is reduced. For redistribution, the removed gate mass is assigned to the other expert of the same shared pool, preserving total shared mass and all task gates. If u is the removed shared expert and v the other shared expert, the intervention sets $g'_{j,u} = 0$ and $g'_{j,v} = g_{j,u} + g_{j,v}$. It leaves every task gate unchanged.

The two shared routes have asymmetric causal effects. Removing the floor-preserved branch lowers the average by 2.54 points, compared with 0.40 points for the entropy-aligned branch. The mass-preserving interventions retain the same ordering: moving the floor-preserved mass to the entropy-aligned expert lowers the average by 3.01 points, whereas the reverse transfer lowers it by 0.21 points. These fixed-checkpoint interventions measure conditional branch utility and non-interchangeability, rather than additive contributions to model accuracy.

I Extended Related Work

Parameter-efficient LoRA fine-tuning. PEFT methods adapt a frozen model by training a small parameter subset. Adapters (Houlsby et al., 2019), soft prompts (Lester et al., 2021), activation scaling (Liu et al., 2022), and quantized adaptation (Dettmers et al., 2023) provide complementary ways to reduce trainable parameters or memory. LoRA (Hu et al., 2022) represents weight updates in a low-rank factorization and can merge the update into the backbone for inference. AdaLoRA

(Zhang et al., 2023) improves rank allocation; LoRA+ (Hayou et al., 2024) adjusts factor learning rates; PiSSA (Meng et al., 2024) changes initialization; and DoRA (Liu et al., 2024) decomposes weight magnitude and direction. These methods mainly strengthen a single adapter. CAST-LoRA instead organizes multiple low-rank experts for mixed-task adaptation without test-time task labels.

Mixtures of LoRA experts. Combining MoE with lightweight adapters expands conditional adaptation capacity without updating the full backbone. Early MoE architectures (Shazeer et al., 2017; Du et al., 2022) established conditional specialization and sparse routing, while sparse expert adaptation (Zadouri et al., 2024) transferred these ideas to compact adapter banks. Existing MoE-LoRA methods broadly follow two directions. Task-bound methods such as MoA (Feng et al., 2024) use supervised selectors to choose domain-specific adapters. Input-dependent methods learn token-, layer-, or rank-wise mixtures. MixLoRA (Li et al., 2024), LoRAMoE (Dou et al., 2024), and MoELoRA (Luo et al., 2024) construct specialized expert banks. HydraLoRA (Tian et al., 2024) shares low-rank factors; FlyLoRA (Zou et al., 2025) routes over ranks; HMoRA (Liao et al., 2025) combines task- and token-level routing; and HotMoE (Huang et al., 2026) varies expert interactions across layers. CAST-LoRA takes inspiration from HMoRA's hierarchical task- and token-level routing but targets three distinct design problems. Specifically, CAST-LoRA uses cluster structure only as temporary training guidance, replaces the auxiliary task-token Transformer with a single-query readout, and reserves explicit shared capacity alongside task-specific experts.

Multi-task sharing and negative transfer. Multi-task learning can improve sample efficiency through shared representations, but unrelated or conflicting tasks may harm one another (Caruana, 1997; Zhang & Yang, 2022). GradNorm (Chen et al., 2018) and FAMO (Liu et al., 2023) balance task progress, while MGDA (Sener & Koltun, 2018), PCGrad (Yu et al., 2020), and CAGrad (Liu et al., 2021) modify multi-task update directions. Task grouping (Standley et al., 2020) and efficient grouping estimates (Fifty et al., 2021) instead identify objectives that benefit from shared training, while MoA (Feng et al., 2024), LoRAMoE (Dou et al., 2024), and FlyLoRA (Zou et al., 2025) allocate low-rank capacity across inputs. Shared/private architectures similarly allocate separate capacity for common and task-specific factors. CAST-LoRA follows the latter principle in a low-rank conditional-computation setting, while avoiding a fixed task-to-private-branch mapping at inference.