GEOFIELD-LORA: READING TASK-SPECIFIC ADAPTERS FROM A SHARED GAUSSIAN PRIMITIVE FIELD

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) provides an efficient way to adapt large language models, but multi-task adaptation must capture both the commonalities and differences across task-specific updates. Existing conditional LoRA methods typically address this through learned mixing, routing, or parameter-generation functions. In practice, such functions can concentrate most tasks on a few shared components. We instead formulate multi-task adaptation from a multi-view perspective, where different tasks obtain related but distinct updates from a shared parameter field. We introduce Geofield-LoRA, which represents shared low-rank updates as Gaussian primitives in a high-dimensional task-view space. A task condition defines a view of this shared field, and ray-based Gaussian responses determine how the primitive-specific low-rank updates are combined into a task-specific LoRA residual. This allows different tasks to reuse the same low-rank component bank while forming distinct task-specific updates through different combinations. Extensive and controlled studies show that Geofield-LoRA makes effective use of shared low-rank capacity: tasks with related adaptation requirements share primitives while retaining distinct updates, highlighting view-conditioned parameter fields as a promising direction for multi-task adaptation.

1 INTRODUCTION

Low-Rank Adaptation (LoRA) has become a standard way for parameter-efficient adaptation of large language models (Hu et al., 2022). By freezing the pretrained backbone and learning small low-rank residuals for selected projections, LoRA makes adaptation cheap and modular. However, extending this approach to multiple tasks introduces a fundamental challenge: multi-task learning studies show that tasks exhibit complex relationships—they can benefit from joint learning, but they also frequently compete (e.g., gradient interference), where different task objectives pull shared parameters in opposite optimization directions (Yu et al., 2020; Standley et al., 2020). These observations raise a central question: how a shared adaptation representation can capture both the commonalities and differences across tasks.

Existing conditional LoRA methods address this problem through adapter selection, mixing, routing, or generation (Wu et al., 2024; Li et al., 2024; Jin et al., 2024; Charakorn et al., 2025; Shao et al., 2025; Shahroz et al., 2025). These methods allow different tasks to use shared low-rank components in different ways. However, the relationships between tasks and shared low-rank components are typically represented implicitly through the outputs of a routing or parameter-prediction function. In practice, such functions can concentrate most tasks on a few shared components. This raises a more basic representation question: rather than asking a global function to predict how every task should use the shared parameters, can we represent the relationships between tasks and shared adaptation parameters more directly?

We approach this question from a multi-view perspective. Instead of treating each task as an independent request for adapter parameters, we consider different tasks as different perspectives on a shared adaptation space. We hypothesize that tasks with related adaptation requirements can reuse overlapping parts of this shared space, while still retaining task-specific differences in their resulting updates. From this perspective, the key is no longer only how to predict an adapter for each task but

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.

how to construct a shared adaptation representation from which different task-specific adapters can be obtained.

Building on this view, we instantiate this idea with Geofield-LoRA, which represents the shared adaptation space as a Gaussian primitive field. A task condition c_t is constructed from a task description, answer schema, and support examples. A frozen text encoder maps this condition to an embedding, and a trainable projection maps the embedding to a task-view coordinate v_t. Each adapted transformer layer contains a shared set of Gaussian primitives in this space, with each primitive associated with its own pair of low-rank factors. Given a task view, we construct an inward ray and evaluate the response of each Gaussian along that ray. The normalized responses determine how the primitive-specific low-rank updates are combined into the task-specific LoRA residual. The task-view mapping, Gaussian geometry, and low-rank factors are learned jointly across tasks. In this way, the low-rank factors determine *what* update each primitive provides, while the learned geometry determines *how* that update is shared across task views.

We evaluate Geofield-LoRA from both broad and controlled perspectives. First, we compare it across multiple language-model backbones and against shared LoRA, dense conditional composition, sparse routing, and published multi-task LoRA methods. We then fix Qwen2.5-1.5B as a controlled analysis platform to study how different parameter-sharing structures behave under matched low-rank budgets. Finally, we analyze the task-specific LoRA updates produced by the shared field and isolate the contribution of its geometric design through component ablations.

Our contributions are threefold.

- We introduce a multi-view formulation of multi-task LoRA, in which task-specific adapters are treated as different views of a shared adaptation field, allowing task commonality and task-specific variation to be represented within the same parameter space.
- We propose Geofield-LoRA, which represents shared low-rank updates with Gaussian primitives in a learned high-dimensional task-view space and constructs task-specific LoRA residuals from their geometric responses to each task view.
- Extensive and controlled studies validate the multi-view formulation and further characterize how the shared field supports task-dependent adaptation.

2 Related Work

Parameter-efficient multi-task adaptation. Parameter-efficient tuning adapts frozen backbones through small trainable modules, including adapters and low-rank residuals (Houlsby et al., 2019; Hu et al., 2022). Multi-task PEFT further asks how such modules should be reused: AdapterFusion composes pretrained task adapters (Pfeiffer et al., 2021), HyperFormer generates adapters with shared hypernetworks (Karimi Mahabadi et al., 2021), ATTEMPT mixes source prompts (Asai et al., 2022), AdaMix learns mixtures of adaptation modules (Wang et al., 2022), and PEMT transfers knowledge through a task-correlation-guided PEFT mixture (Lin et al., 2024). More recent multi-task LoRA work has also revisited how a shared low-rank space should be organized, for example through explicit task-adaptive parameters, alignment-based sharing, or heterogeneity-aware subspace allocation (Yang et al., 2024b; Liu et al., 2025; Liang et al., 2025). These works establish parameter sharing as a central problem in multi-task PEFT.

Conditional LoRA composition and routing. Recent LoRA-specific work composes, routes, or generates low-rank updates. LoraHub searches weighted combinations of existing LoRA modules for new tasks (Huang et al., 2023), while MoLE, MixLoRA, AdaMoLE, and HMoRA use LoRA experts with gating, sparse routing, adaptive thresholds, or hierarchical dispatch (Wu et al., 2024; Li et al., 2024; Liu & Luo, 2024; Liao et al., 2025). Related multi-task variants also mix universal and domain-specific experts or share part of the LoRA factorization across experts, as in MoDULA and MoSLD (Ma et al., 2024; Zhao et al., 2025). Polytropon and MHR learn task-indexed allocations over a shared inventory of adapters (Ponti et al., 2023; Page-Caccia et al., 2023), whereas Geofield-LoRA derives the combination from task conditions. A complementary line generates LoRA parameters from task conditions, including conditional Diffusion, online LoRA generation, text-to-LoRA generation, CVAE-based meta LoRA generation, and recurrent Diffusion (Jin et al., 2024; Xiao et al., 2025; Charakorn et al., 2025; Shao et al., 2025; Shahroz et al., 2025). Geofield-LoRA addresses the

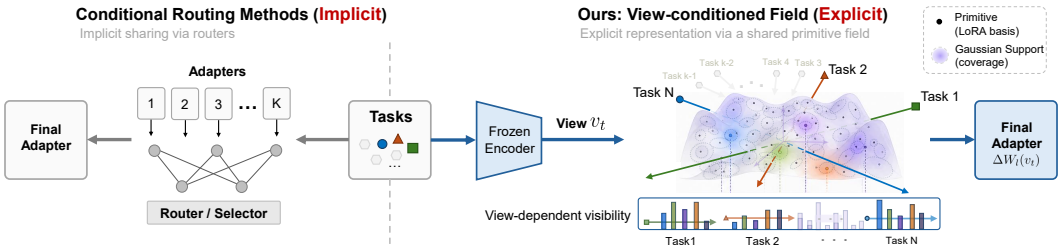

Figure 1: Overview of Geofield-LoRA. A task condition defines a view in the shared task-view space. Each adapted layer contains Gaussian primitives associated with low-rank factors. The task view determines primitive responses, which are normalized and used to combine the primitive-specific low-rank updates into the task-specific LoRA residual.

same goal of task-specific adaptation, but represents the relation between tasks and shared low-rank components through a view-conditioned parameter field.

View-dependent primitive representations. Our formulation is inspired by 3D Gaussian Splatting, which represents a scene with a shared set of optimizable Gaussian primitives whose contributions vary with viewpoint (Kerbl et al., 2023; Bao et al., 2025). Unlike 3DGS, our task-view space has no physical occlusion; we therefore borrow the shared Gaussian primitives and their view-dependent contributions, but not projection or alpha compositing (Appendix D.2).

3 METHOD

3.1 OVERVIEW

Geofield-LoRA constructs task-specific LoRA updates by querying a shared field of low-rank primitives with a task view. A task condition is mapped to a task view in a high-dimensional space. Each adapted layer contains a shared set of Gaussian primitives, each associated with a low-rank update. The geometric relation between the task view and the primitives determines their contributions. These contributions are normalized and used to combine the primitive-specific low-rank updates into the task-specific LoRA residual. The task-view mapping, primitive geometry, and low-rank factors are optimized jointly across tasks. Figure 1 summarizes the overall framework.

3.2 GAUSSIAN LOW-RANK PRIMITIVE FIELD

Each adapted transformer layer maintains a field of K Gaussian low-rank primitives in the task-view space:

$$\Phi = \Big\{ \big(\underbrace{\boldsymbol{\mu}_k,\ \boldsymbol{\Lambda}_k,\ s_k}_{\text{Geometry}},\ \underbrace{\mathbf{A}_k,\ \mathbf{B}_k}_{\text{LoRA factors}} \big) \Big\}_{k=1}^{K}. \tag{1}$$

The geometric parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k, s_k)$ define the support of primitive k in the task-view space: $\boldsymbol{\mu}_k$ is its center, $\boldsymbol{\Lambda}_k$ is its precision matrix, and $s_k > 0$ controls its strength. The LoRA factors $(\mathbf{A}_k, \mathbf{B}_k)$ define the rank-r update associated with the primitive:

$$\Delta \mathbf{W}_k = \mathbf{B}_k \mathbf{A}_k, \tag{2}$$

where $\mathbf{A}_k \in \mathbb{R}^{r \times n_{\text{in}}}$ and $\mathbf{B}_k \in \mathbb{R}^{n_{\text{out}} \times r}$.

The two groups of parameters play complementary roles: the low-rank factors determine *what* update each primitive provides, while the Gaussian geometry determines *how strongly* that update contributes to different task views.

3.3 TASK VIEWS

Each task is represented by a task-level condition c, constructed from its task description, answer schema, and a small set of support examples. We map this condition to a d_v-dimensional task view $\mathbf{v}$ using a frozen text encoder E followed by a trainable linear projection:

$$\mathbf{v} = \mathbf{W}_{\text{view}} E(c). \tag{3}$$

The text encoder remains fixed, while $\mathbf{W}_{\text{view}}$ is optimized jointly with the Gaussian primitives and low-rank factors. All prompt templates belonging to the same task share the same condition c and therefore use the same task view $\mathbf{v}$. The same task view is supplied to every adapted layer, while each layer maintains its own Gaussian primitive field.

3.4 View-Conditioned Primitive Response

A task view determines both where and from which direction the shared primitive field is queried. Given a task view $\mathbf{v}$, we define the inward unit direction:

$$\mathbf{d} = -\frac{\mathbf{v}}{\|\mathbf{v}\|_2}, \tag{4}$$

and the corresponding task ray:

$$\mathbf{r}(u) = \mathbf{v} + u\mathbf{d}, \qquad u \geq 0. \tag{5}$$

Different task views therefore query the same primitive field along different inward rays.

Each Gaussian primitive defines a localized, anisotropic region in the task-view space. As described above, primitive k has center $\boldsymbol{\mu}_k$ and precision matrix $\boldsymbol{\Lambda}_k$. The response of this Gaussian at any point $\mathbf{x}$ in the task-view space is:

$$G_k(\mathbf{x}) = s_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Lambda}_k (\mathbf{x} - \boldsymbol{\mu}_k)\right]. \tag{6}$$

The precision matrix $\boldsymbol{\Lambda}_k$ determines the scale and orientation of the Gaussian support, and therefore how quickly the primitive response decreases in different directions.

Given the task ray $\mathbf{r}(u)$, we first locate the point that maximizes the Gaussian response along the unconstrained ray direction. Equivalently, we minimize its Mahalanobis distance to the primitive:

$$u_k^{\text{raw}} = \arg\min_{u \in \mathbb{R}} \big(\mathbf{r}(u) - \boldsymbol{\mu}_k\big)^\top \boldsymbol{\Lambda}_k \big(\mathbf{r}(u) - \boldsymbol{\mu}_k\big). \tag{7}$$

which has the closed form:

$$u_k^{raw} = \frac{\mathbf{d}^\top \boldsymbol{\Lambda}_k (\boldsymbol{\mu}_k - \mathbf{v})}{\mathbf{d}^\top \boldsymbol{\Lambda}_k \mathbf{d}}, \tag{8}$$

where $\mathbf{d} = -\mathbf{v}/\|\mathbf{v}\|_2$ is the inward ray unit direction. For the forward half-ray, the exact constrained solution is $\max(0, u_k^{\text{raw}})$. During optimization, we use a SoftPlus approximation to this constraint to maintain smooth gradients:

$$u_k^\star = \text{SoftPlus}(u_k^{raw}). \tag{9}$$

Substituting this point into Eq. 6 gives the view-conditioned response of primitive k:

$$q_k = s_k \exp\left[-\frac{1}{2}\big(\mathbf{r}(u_k^\star) - \boldsymbol{\mu}_k\big)^\top \boldsymbol{\Lambda}_k \big(\mathbf{r}(u_k^\star) - \boldsymbol{\mu}_k\big)\right]. \tag{10}$$

Finally, the responses are normalized across primitives:

$$w_k = \frac{q_k}{\sum_{j=1}^{K} q_j + \epsilon}. \tag{11}$$

The resulting weight w_k determines how strongly primitive k contributes to the adapter for the current task view.

3.5 Task-Specific LoRA Update

The normalized response weights in Eq. 11 determine how the shared low-rank primitives are combined for the current task view. As defined in Eq. 1, primitive k carries the LoRA factors $\mathbf{A}_k$ and $\mathbf{B}_k$. The resulting task-specific weight update is:

$$\Delta\mathbf{W} = \sum_{k=1}^{K} w_k \, \mathbf{B}_k \mathbf{A}_k. \tag{12}$$

The adapted projection is then:

$$\mathbf{W}^{\text{adapted}}(\mathbf{v}) = \mathbf{W} + \frac{\lambda_{\text{LoRA}}}{r}\Delta\mathbf{W}(\mathbf{v}), \tag{13}$$

where $\mathbf{W}$ is the frozen pretrained weight and λ_{LoRA}/r follows the standard LoRA scaling.

Overall, the Gaussian geometry determines how strongly each primitive contributes to a task view, while its low-rank factors determine the corresponding parameter update. In this way, different task views obtain different adaptations from the same shared primitive field.

3.6 Training and Inference

During training, the pretrained backbone and the task encoder E are frozen. The task-view projection $\mathbf{W}_{\text{view}}$, Gaussian primitive parameters, and LoRA factors $\{\mathbf{A}_k, \mathbf{B}_k\}$ are jointly optimized under the task supervision. At inference time, a task condition c is encoded once to obtain its task view $\mathbf{v}$. For each adapted layer, the task view determines the view-conditioned primitive weights $\{w_k\}_{k=1}^{K}$, which are then used to construct the task specific LoRA update in Eq. 12. The same task view is reused for all inputs belonging to the task.

4 Experiments

Benchmark and Backbones. We evaluate on CLEAN56, a subset of PromptSource (Bach et al., 2022) in which every task has at least five prompt templates with a consistent target schema. For each task, three templates are used for training and two disjoint templates for testing, so evaluation measures generalization across prompt formulations; all templates of a task share one task condition. We report exact-match accuracy for classification and ROUGE-L for generation. We use the instruction-tuned versions of Qwen2.5-1.5B (Yang et al., 2024a), Gemma-2-2B (Gemma Team, 2024), and Llama-3.2-3B (Meta, 2024), and adapt the query and value projections of every attention layer while keeping the backbone frozen. Task conditions are encoded by a frozen Qwen3-Embedding-0.6B model (Zhang et al., 2025) and linearly projected to a 32-dimensional task view.

Baselines and training. We compare with the frozen backbone, Shared LoRA, LoRA-MLP (dense task-conditioned mixing), MixLoRA (sparse top-k routing), MOELoRA, and MoSLD. MoSLD keeps its original token-level routing and factor-sharing design. The other conditional baselines receive the same task view and the same component budget. All methods share the data split, target modules, optimizer, auxiliary losses, and schedule (batch size 48, three epochs). Implementation details are given in Appendices A.1 and B.

4.1 Cross-Backbone Multi-Task Adaptation

Table 1(a) compares Geofield-LoRA with shared, conditional-composition, and expert-routing baselines across three backbone families. Geofield-LoRA achieves the highest test score on all three backbones, reaching 60.62 on Qwen2.5-1.5B, 67.10 on Gemma-2-2B, and 66.20 on Llama-3.2-3B. The gains over the strongest non-Geofield baseline are 1.09, 4.05, and 2.91 points, respectively. Moreover, the improvements are accompanied by smaller train–test gaps, particularly on Gemma and Llama. Table 1(b) provides a complementary task-level view of these results. Across all three backbones, Geofield-LoRA shows positive paired differences against LoRA-MLP, MixLoRA, MOELoRA, and MoSLD, with all but one of the corresponding 95% confidence intervals remaining above zero. The W/T/L counts further show that these improvements are distributed across multiple tasks, although their range varies across methods and backbones.

We additionally examine whether the task-conditioned combination mechanism should be shared across adapted layers. For Geofield-LoRA, using a layer-specific combination mechanism improves test score on all three backbones compared to a layer-shared counterpart. We therefore adopt the layer-specific setting for the controlled comparison below.

4.2 Controlled Study of Shared Adaptation Mechanisms

To examine how shared adaptation capacity is organized, we fix Qwen2.5-1.5B and compare four configurations under matched low-rank budgets $Kr \in \{64, 128, 256, 512\}$. Shared LoRA uses a

Table 1: **Cross-backbone performance and per-task consistency.** $K = 32$, $r = 16$; Shared LoRA uses rank 512. (a) test and train scores and their gap ($\mathrm{Gap} = \mathrm{Train} - \mathrm{Test}$). Shared indicates whether the task-conditioned combination mechanism is shared across adapted layers. (b) test score, paired differences ($\Delta = \mathrm{Geo} - \mathrm{baseline}$), 95% confidence intervals, and Geofield-LoRA's task-level W/T/L. Best and second-best adaptation results are shown in **bold** and underline, respectively.

(a) Cross-backbone comparison.

Method	Shared	Qwen2.5-1.5B			Gemma-2-2B			Llama-3.2-3B		
		Test↑	Train↑	Gap↓	Test↑	Train↑	Gap↓	Test↑	Train↑	Gap↓
Frozen backbone	–	12.02	12.38	0.36	8.94	10.13	1.19	10.49	10.07	−0.42
Shared LoRA	–	58.30	74.97	16.67	58.06	75.48	17.42	61.39	76.28	14.89
LoRA-MLP	×	58.56	74.98	16.42	60.61	76.07	15.46	62.73	76.67	13.94
MixLoRA	×	59.53	75.08	15.55	63.05	76.72	13.67	63.29	77.30	14.01
MOELoRA	✓	59.05	74.63	15.58	61.92	76.74	14.82	62.88	76.93	14.05
MoSLD	×	53.34	73.72	20.38	59.58	75.83	16.25	62.91	77.01	14.10
Geofield-LoRA	✓	59.16	74.83	15.67	66.02	77.57	11.55	65.90	**78.03**	12.13
Geofield-LoRA	×	**60.62**	**75.26**	**14.64**	**67.10**	**77.63**	**10.53**	**66.20**	**78.03**	**11.83**

(b) Per-task consistency

Method	Qwen2.5-1.5B				Gemma-2-2B				Llama-3.2-3B			
	Test↑	Δ	95% CI	W/T/L	Test↑	Δ	95% CI	W/T/L	Test↑	Δ	95% CI	W/T/L
Geofield-LoRA	**60.62**	–	–	–	**67.10**	–	–	–	**66.20**	–	–	–
Shared LoRA	58.30	+2.32	[0.23, 4.74]	34/0/22	58.06	+9.04	[5.75, 12.95]	44/1/11	61.39	+4.81	[2.32,7.44]	38/0/18
LoRA-MLP	58.56	+2.06	[0.20, 4.16]	42/0/14	60.61	+6.49	[2.86, 10.42]	34/0/22	62.73	+3.47	[0.91, 6.24]	35/0/21
MixLoRA	59.53	+1.09	[−0.03, 2.91]	38/0/18	63.05	+4.05	[0.93, 7.44]	31/0/25	63.29	+2.91	[0.11, 6.75]	37/0/19
MOELoRA	59.05	+1.57	[0.59, 3.11]	37/0/19	61.92	+5.18	[2.00, 8.68]	35/0/21	62.88	+3.32	[1.04, 5.82]	38/0/18
MoSLD	53.34	+7.28	[4.65, 9.71]	43/0/13	59.58	+7.52	[3.88, 11.50]	32/0/24	62.91	+3.29	[1.17, 5.50]	37/0/19

task-agnostic adapter, LoRA-MLP performs dense task-conditioned combination, MixLoRA uses sparse top-k routing, and Geofield-LoRA uses the proposed view-conditioned geometric combination.

Table 2 shows that the methods use increasing capacity differently. Shared LoRA remains nearly unchanged as Kr grows, while LoRA-MLP improves only modestly. MixLoRA benefits more from the larger component bank, but Geofield-LoRA improves consistently from 59.02 to 60.62 test score. Its advantage over the strongest conditional baseline therefore increases from 0.29 points at $Kr = 64$ to 1.09 points at $Kr = 512$. Meanwhile, train score changes only slightly, and the train–test gap decreases from 15.57 to 14.64, indicating that the additional capacity primarily improves test performance. Additional comparisons across (K, r) configurations and seed-level results are provided in Figure A1.

4.3 Analysis of the Gaussian Primitive Field

The previous experiments compare adaptation performance. We next examine how tasks use the shared field, first through the combination weights and then through the resulting LoRA updates.

Task–primitive combination weights. Figure 2 shows the final combination weights of all tasks at a representative projection of each backbone, with every panel sorted by its own dominant components. Geofield-LoRA assigns tasks to 16, 26, and 25 distinct dominant primitives on Qwen, Gemma, and Llama, forming clear task-specific allocations with balanced component usage. Given the same task view and component budget, the conditional baselines concentrate most tasks on a few components. MixLoRA has 1, 6, and 5 dominant components, and LoRA-MLP has 2 on every backbone. We additionally measure task diversity as one minus the mean pairwise cosine similarity between task weight vectors, which is 0 when all tasks receive the same weights. Geofield-LoRA reaches 0.79–0.87, compared with 0.01–0.52 for MixLoRA and 0.01–0.31 for LoRA-MLP. These results show that Geofield-LoRA gives different tasks different primitive combinations, while MixLoRA and LoRA-MLP route most tasks to the same few components.

Figure A3 further shows that this allocation emerges during training. Most tasks initially rely on the same few primitives, and task diversity rises from 0.00–0.32 at 2% of training to 0.79–0.87 at the end. The field thus first captures what tasks have in common and then how they differ.

Task organization. Intuitively, tasks of the same type should share primitives. This is partly true. Among task pairs that share a dominant primitive, 22–36% belong to the same task type (Appendix D.4), whereas only 11% of randomly drawn task pairs do. Tasks of the same type

Table 2: **Controlled comparison on Qwen2.5-1.5B.** Results under matched low-rank budgets $Kr \in \{64, 128, 256, 512\}$. Test and train scores are reported as mean±std over five runs, with their gap. $H/\log K$ denotes the entropy of the normalized primitive weights (lower indicates more concentrated). For Params./layer, percentages denote additional trainable overhead.

Method	K	r	Test↑	Train↑	Gap↓	$H/\log K$	Params. / layer
				$Kr = 64$			
Shared LoRA	1	64	58.18 ± 0.89	**74.67** ± 0.17	16.49	—	196K
LoRA-MLP	8	8	57.59 ± 0.69	74.58 ± 0.47	16.99	0.986 ± 0.005	196K (+0.7%)
MixLoRA	8	8	$\underline{58.73} \pm 0.80$	74.55 ± 0.46	$\underline{15.82}$	0.332 ± 0.000	196K (+0.7%)
Geofield-LoRA	8	8	**59.02** ± 1.44	$\underline{74.59} \pm 0.78$	**15.57**	0.519 ± 0.038	196K (+2.3%)
				$Kr = 128$			
Shared LoRA	1	128	58.04 ± 0.92	**75.04** ± 0.45	17.00	—	393K
LoRA-MLP	16	8	58.28 ± 0.44	74.72 ± 0.25	16.44	0.981 ± 0.004	393K (+0.4%)
MixLoRA	16	8	$\underline{59.21} \pm 0.77$	74.87 ± 0.33	$\underline{15.66}$	0.498 ± 0.000	393K (+0.4%)
Geofield-LoRA	16	8	**59.61** ± 0.78	$\underline{74.89} \pm 0.44$	**15.28**	0.599 ± 0.072	393K (+2.3%)
				$Kr = 256$			
Shared LoRA	1	256	58.22 ± 1.00	74.98 ± 0.37	16.76	—	786K
LoRA-MLP	16	16	58.37 ± 0.86	75.03 ± 0.42	16.66	0.956 ± 0.007	786K (+0.2%)
MixLoRA	16	16	$\underline{59.49} \pm 0.80$	$\underline{75.05} \pm 0.63$	$\underline{15.56}$	0.494 ± 0.000	786K (+0.2%)
Geofield-LoRA	16	16	**60.21** ± 0.89	**75.07** ± 0.45	**14.86**	0.509 ± 0.032	786K (+1.1%)
				$Kr = 512$			
Shared LoRA	1	512	58.30 ± 0.75	74.97 ± 0.37	16.67	—	1573K
LoRA-MLP	32	16	58.56 ± 0.92	74.98 ± 0.55	16.42	0.955 ± 0.012	1573K (+0.1%)
MixLoRA	32	16	$\underline{59.53} \pm 0.74$	$\underline{75.08} \pm 0.60$	$\underline{15.55}$	0.594 ± 0.003	1573K (+0.1%)
Geofield-LoRA	32	16	**60.62** ± 0.60	**75.26** ± 0.36	**14.64**	0.629 ± 0.009	1573K (+1.1%)

Table 3: **Task-specific LoRA update structure.** Mean cosine reports the average pairwise cosine similarity between task-specific LoRA updates. First mode reports the percentage of task-dependent variation explained by the leading principal direction, and d_{eff} denotes the effective dimension.

Method	Qwen2.5-1.5B			Gemma-2-2B			Llama-3.2-3B		
	Mean cosine	First mode	d_{eff}	Mean cosine	First mode	d_{eff}	Mean cosine	First mode	d_{eff}
Geofield-LoRA	0.55	16.53	18.14	0.31	8.49	28.00	0.58	10.57	22.65
MixLoRA	0.94	73.07	1.84	0.71	34.45	6.13	0.63	45.25	4.19
LoRA-MLP	0.94	89.33	1.24	0.93	81.13	1.48	0.84	70.71	1.88

are therefore two to three times more likely than a random pair to share a primitive (permutation $p < 0.001$). Most sharing, however, still crosses task types. For instance, on all three backbones, WiC (word sense) shares its dominant primitive with PAWS (paraphrase identification): both judge whether two texts carry the same meaning and answer with a binary label. Conversely, Yelp star rating and Yelp polarity share a source and a task type, yet use different dominant primitives; they differ in output, as the first predicts one of five star ratings and the second only predicts positive or negative. Human-defined task types thus capture only part of how tasks are related; the field also recovers relations across types that the labels miss, consistent with our hypothesis that tasks with related adaptation requirements reuse overlapping parts of the shared field.

Distinct Task-Specific LoRA Updates. Different weights can still produce similar LoRA updates. We therefore compare the LoRA updates produced for different tasks. We conduct this analysis at $K = 32$ and $r = 16$ (Table 3). First, we use cosine similarity to compare the directions of the LoRA updates across tasks. Geofield-LoRA produces much lower task-to-task similarity between the updates of different tasks on all three backbones, while LoRA-MLP produces highly aligned updates (0.84–0.94). MixLoRA reduces this similarity through sparse top-k routing, but its task-to-task similarity remains higher than that of Geofield-LoRA.

PCA of the updates shows the same pattern. On Qwen2.5-1.5B, the first principal direction explains 16.53% of the differences between tasks for Geofield-LoRA, compared with 73.07% for MixLoRA and 89.33% for LoRA-MLP, and the effective dimension is 18.14, compared with 1.84 and 1.24. Gemma and Llama show similar trends. Figure A6 further shows that the differences between task-specific updates of Geofield-LoRA gradually spread across more principal directions, while MixLoRA and LoRA-MLP end with much lower effective dimensions.

Geometry of the Gaussian Primitive Field. Training reshapes the primitive field at every capacity (Table 4, Figure A2). Across the four budgets, primitive anisotropy increases by 44–58, the long

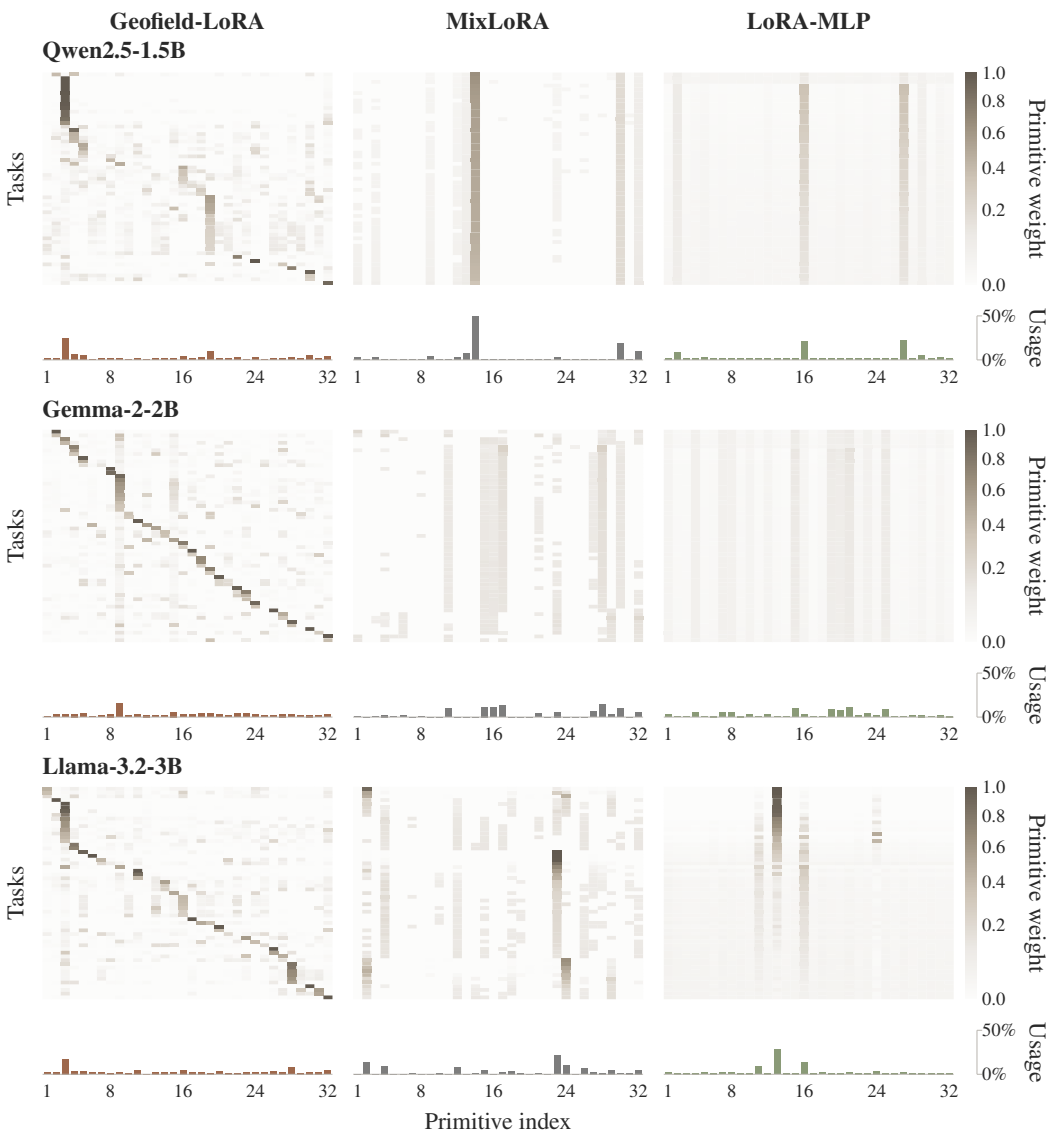

Figure 2: **Task–primitive combination weights.** Each panel shows one representative projection per backbone and is sorted by its own dominant primitives. Bars show mean usage over tasks.

axes of the primitives align more closely with the task rays (+0.026–0.043), and the primitive centers move by 0.42–0.46 from their initialization. The Mahalanobis distance from each task view to its nearest primitive also grows (+2.8–4.7), since the ray query evaluates primitives along the task ray rather than at the view itself. We then ablate the three geometric choices of Geofield-LoRA (Table 5): Point-Query evaluates each Gaussian at the task-view coordinate instead of along the inward ray, fixed geometry freezes the Gaussian parameters, and diagonal precision restricts each precision matrix to a diagonal form. All three variants lower the test score, from 60.62 to 58.61, 58.74, and 59.52, respectively, with the ray query contributing the most. The ray query also allows the primitives to become substantially more anisotropic (+44.2, vs. +13.2 under Point-Query). Performance therefore depends not only on having a Gaussian field, but also on how the field is queried and how its geometry adapts.

4.4 Training and Inference Efficiency

Table 6 compares the computational cost of the controlled methods at $Kr = 512$. Optimizing the Gaussian geometry and computing view-conditioned responses increase training time to 4.73 hours,

Table 4: **Geometry evolution across field capacities.** Layer-averaged geometric statistics before and after training. A denotes primitive anisotropy, A_l view-axis alignment, M task–primitive Mahalanobis distance, and C center displacement. Each capacity is reported as Kr with (K/r) in parentheses.

State	$Kr = 64$ (8/8)				$Kr = 128$ (16/8)				$Kr = 256$ (16/16)				$Kr = 512$ (32/16)			
	A	A_l	M	C	A	A_l	M	C	A	A_l	M	C	A	A_l	M	C
Init.	67.0	0.147	4.20	0.000	71.6	0.141	2.88	0.000	71.5	0.143	2.90	0.000	72.4	0.141	2.27	0.000
Trained	124.8	0.190	8.88	0.462	117.9	0.182	6.43	0.416	127.8	0.175	6.38	0.429	116.6	0.167	5.09	0.423
Δ	+57.8	+0.043	+4.68	+0.462	+46.3	+0.041	+3.55	+0.416	+56.3	+0.033	+3.48	+0.429	+44.2	+0.026	+2.83	+0.423

Table 5: **Component and geometric analysis.**

(a) Component ablation. $Kr = 512$

Variant	Ray	Geo.	Prec.	Test↑
Point-Query	×	✓	✓	58.61
Fixed geo.	✓	×	✓	58.74
Diag. prec.	✓	✓	×	<u>59.52</u>
Geofield-LoRA	✓	✓	✓	**60.62**

(b) Geometry change. *Trained − Init.*

Variant	ΔA	ΔA_l	ΔM	ΔC
Point-Query	+13.23	+0.0027	+0.697	+0.415
Geofield-LoRA	**+44.20**	**+0.0260**	**+2.83**	+0.423

Table 6: **Training and inference efficiency.**

Efficiency at one H100 GPU. $Kr = 512$

Metric	Shared	MLP	MixLoRA	**Ours**
Training				
Train time (h)↓	**3.26**	<u>3.37</u>	3.51	4.73
Train ex/s↑	**110.56**	<u>97.28</u>	93.04	73.31
Train peak (GiB)↓	**33.45**	<u>33.99</u>	<u>33.99</u>	34.05
Inference				
Latency (ms)↓	**501.33**	600.95	583.44	<u>577.97</u>
Throughput (tok/s)↑	**410.12**	357.62	354.05	<u>358.65</u>
Infer. peak (GiB)↓	**3.85**	**3.85**	**3.85**	<u>3.86</u>

compared with 3.26–3.51 hours for the baselines, while peak training memory remains similar. At inference, all inputs of a task share one task view, so the primitive weights are computed once and cached; latency, throughput, and peak memory are therefore on par with the conditional baselines. Moreover, because the resulting update $\Delta\mathbf{W}$ is fixed for each task, it can be merged into the frozen weights when serving a single task, as in standard LoRA, removing the remaining adapter overhead. Thus, Geofield-LoRA offers a favorable trade-off between training cost and model performance.

5 Discussion

The central question of this work is whether multi-task adaptation can represent both shared and task-specific behavior through task-dependent combinations over a shared parameter field. Geofield-LoRA addresses this by mapping each task condition to a view of the field and weighting its low-rank components by their Gaussian responses along the view ray. In our experiments, tasks that do similar things share primitives even across task types and still receive distinct LoRA updates, whereas the conditional baselines, given the same inputs, route most tasks to the same few components.

A multi-view formulation naturally raises the question of new views. In 3D Gaussian Splatting, a trained scene can be rendered from viewpoints never seen during training; in the same way, a trained field could be read from the view of a task never seen during training. Recently, several methods generate LoRA weights directly from task descriptions or prompts, such as Text-to-LoRA (Charakorn et al., 2025) and diffusion-based generators (Jin et al., 2024; Shahroz et al., 2025). Our formulation offers these methods a simpler target: instead of generating all LoRA weights for a new task, they only need to produce a view of a trained primitive field, and the new task then reuses the primitives already shared by related tasks.

6 Conclusion

We introduced Geofield-LoRA, a multi-view formulation of multi-task LoRA that represents shared low-rank updates as a Gaussian primitive field, which allows different task views to combine the same low-rank primitives differently. Across multiple backbones and matched low-rank budgets, Geofield-LoRA achieves higher test scores than shared and conditional adaptation baselines. Given the same task views and component budget, the conditional baselines route most tasks to the same few components, whereas Geofield-LoRA gives different tasks different primitive combinations. Analyses show that tasks with related adaptation requirements come to share primitives even across human-defined task types, while their resulting updates remain distinct and spread across many principal directions.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript polishing and editing, discussion of experimental analysis, design, and presentation, implementation of parts of the proposed algorithm, data processing, and modification of experimental scripts. All AI-assisted text, code, analyses, and experimental workflows were reviewed and verified by the authors. The authors take full responsibility for the final manuscript, implementation, experimental results, and conclusions.

REFERENCES

Akari Asai, Mohammadreza Salehi, Matthew E. Peters, and Hannaneh Hajishirzi. ATTEMPT: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6655–6672. Association for Computational Linguistics, 2022.

Stephen H. Bach et al. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.acl-demo.9/.

Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3D Gaussian splatting: Survey, technologies, challenges, and opportunities. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(7):6832–6852, 2025.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7485–7514. PMLR, 2025.

Gemma Team. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024. doi: 10.48550/arXiv.2408.00118. URL https://arxiv.org/abs/2408.00118.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. doi: 10.1037/h0071325.

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*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 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.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition, 2023. URL https://arxiv.org/abs/2307.13269.

Xiaolong Jin, Kai Wang, Dongwen Tang, Wangbo Zhao, Yukun Zhou, Junshu Tang, and Yang You. Conditional LoRA parameter generation, 2024. URL https://arxiv.org/abs/2408.01415.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2021.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

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, 2024. URL https://arxiv.org/abs/2404.15159.

Jian Liang, Wenke Huang, Xianda Guo, Guancheng Wan, Bo Du, and Mang Ye. ThanoRA: Task heterogeneity-aware multi-task low-rank adaptation, 2025. URL https://arxiv.org/abs/2505.18640.

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. URL https://openreview.net/forum?id=lTkHiXeuDl.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Zhisheng Lin, Han Fu, Chenghao Liu, Zhuo Li, and Jianling Sun. PEMT: Multi-task correlation guided mixture-of-experts enables parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6869–6883, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.410. URL https://aclanthology.org/2024.findings-acl.410/.

Jinda Liu, Bo Cheng, Yi Chang, and Yuan Wu. Align, don't divide: Revisiting the LoRA architecture in multi-task learning, 2025. URL https://arxiv.org/abs/2508.05078.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When MOE meets LLMs: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1104–1114. ACM, 2024. doi: 10.1145/3626772.3657722. URL https://doi.org/10.1145/3626772.3657722.

Zefang Liu and Jiahua Luo. AdaMoLE: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=ndY9qFf9Sa.

Yufei Ma, Zihan Liang, Huangyu Dai, Ben Chen, Dehong Gao, Zhuoran Ran, Zihan Wang, Linbo Jin, Wen Jiang, Guannan Zhang, Xiaoyan Cai, and Libin Yang. MoDULA: Mixture of domain-specific and universal LoRA for multi-task learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2758–2770, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.161. URL https://aclanthology.org/2024.emnlp-main.161/.

Meta. Llama 3.2 3B Instruct. Hugging Face model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct. Released September 25, 2024.

Lucas Page-Caccia, Edoardo Maria Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. Multi-head adapter routing for cross-task generalization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56916–56931. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b295b3a940706f431076c86b78907757-Abstract-Conference.html.

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*, pp. 487–503. Association for Computational Linguistics, 2021.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining parameter-efficient modules for task-level generalisation. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 687–702, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.49. URL https://aclanthology.org/2023.eacl-main.49/.

Rana Shahroz, Dongwen Tang, Pingzhi Li, Kai Wang, and Tianlong Chen. ORAL: Prompting your large-scale LoRAs via conditional recurrent diffusion. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 1357–1370, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.71. URL https://aclanthology.org/2025.findings-emnlp.71/.

Yihua Shao, Minxi Yan, Yang Liu, Siyu Chen, Wenjie Chen, Xinwei Long, Ziyang Yan, Lei Li, Chenyu Zhang, Nicu Sebe, Hao Tang, Yan Wang, Hao Zhao, Mengzhu Wang, and Jingcai Guo. In-context meta LoRA generation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 6138–6146. International Joint Conferences on Artificial Intelligence Organization, 2025.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9120–9132. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/standley20a.html.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2022.

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

Yicheng Xiao, Lin Song, Rui Yang, Cheng Cheng, Yixiao Ge, Xiu Li, and Ying Shan. LoRA-Gen: Specializing large language model via online LoRA generation. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oZM5g4IvmS.

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, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024a. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, Weizhu Chen, and Yunhai Tong. MTL-LoRA: Low-rank adaptation for multi-task learning, 2024b. URL https://arxiv.org/abs/2410.09437.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Lulu Zhao, Weihao Zeng, Xiaofeng Shi, and Hua Zhou. MoSLD: An extremely parameter-efficient mixture-of-shared LoRAs for multi-task learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 1647–1659, Abu Dhabi, UAE, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.111/.

A Implementation Details

A.1 Task Conditions and Task Views Implementation Details

To obtain a stable representation for each task, we first construct a task-level condition that is shared across all prompt templates of that task. Specifically, each condition contains the task description, answer schema, and four rendered prompt–answer examples from the training set. Test templates are never used during training. As a result, different prompt templates of the same task are always associated with the same task information.

Following the query-side input format of Qwen3-Embedding, we provide a fixed instruction describing the desired task representation and place the constructed task condition in the `Query` field. The complete encoder input is:

Listing 1: Input format for task-condition encoding.

```
Instruct: Represent this task condition by its task intent, answer schema
    , and few-shot behavioral signature so that behaviorally similar
    tasks are close together.
Query: Task: <task description>

Answer schema:
<answer schema>

Support examples:

Example 1
Prompt: <rendered input 1>
Answer: <target 1>

Example 2
Prompt: <rendered input 2>
Answer: <target 2>

Example 3
Prompt: <rendered input 3>
Answer: <target 3>

Example 4
Prompt: <rendered input 4>
Answer: <target 4>
```

We encode this input with the frozen Qwen3-Embedding-0.6B model and use the hidden state of the final valid token as the 1024-dimensional task embedding. We do not apply additional L2 normalization. The embedding is then cached, and a trainable linear projection maps it into the 32-dimensional task-view space:

$$\mathbf{v} = \mathbf{W}_{\text{view}} E(c). \tag{14}$$

To initialize $\mathbf{W}_{\text{view}}$, we use PCA (Hotelling, 1933). Specifically, we first center the cached embeddings of the training tasks and perform PCA on them. We retain the leading 32 principal components, matching the dimensionality of our task-view projection, and use the corresponding projection directions to initialize $\mathbf{W}_{\text{view}}$. We then rescale the projected coordinates so that all dimensions have the same prescribed standard deviation, providing a balanced scale for the initial task views. After initialization, $\mathbf{W}_{\text{view}}$ is optimized jointly with the Gaussian primitive fields and low-rank factors.

The Qwen3 encoder remains frozen throughout training. The same task view is supplied to every adapted layer, while each layer maintains its own Gaussian primitive field.

B Baseline Implementation Details

We provide additional implementation details for the conditional adaptation baselines used in the main experiments. For LoRA-MLP, MixLoRA, and MOELoRA, we use the same cached task

representations and the same trainable projection to the task-view space as in Geofield-LoRA, and therefore receive the same 32-dimensional task view $\mathbf{v}$, providing a controlled comparison of how each method combines the shared components. MoSLD retains its original token-conditioned routing and factor-sharing structure, which we explain below.

LoRA-MLP and MixLoRA. For an adapted layer ℓ, LoRA-MLP uses an independent two-layer MLP gate g_ℓ with architecture $32 \to 32 \to K$. A GELU (Gaussian Error Linear Unit) activation is applied between the two linear layers. Given the task view $\mathbf{v}$, the gate produces one logit for each of the K low-rank components, and softmax normalization gives the combination weights:

$$\mathbf{w}_\ell^{\mathrm{MLP}} = \mathrm{Softmax}(g_\ell(\mathbf{v}))\,, \tag{15}$$

where $\mathbf{w}_\ell^{\mathrm{MLP}} \in \mathbb{R}^K$ contains the component weights for adapted layer ℓ. All K components therefore participate in the resulting LoRA update.

MixLoRA (Li et al., 2024) uses the same task view and the same MLP gate, but sparsifies the resulting combination:

$$\mathbf{w}_\ell^{\mathrm{Mix}} = \mathrm{TopKNorm}_{k_{\mathrm{route}}}\left(\mathbf{w}_\ell^{\mathrm{MLP}}\right). \tag{16}$$

Here, k_{route} is the number of selected components, and $\mathrm{TopKNorm}$ denotes the operation that retains the k_{route} largest weights, sets the remaining weights to zero, and renormalizes the retained weights to sum to one. Thus, LoRA-MLP forms a dense combination over all components, while MixLoRA forms a sparse combination over the highest-weighted components. For $K = 8$, we follow the official MixLoRA configuration and set $k_{\mathrm{route}} = 2$. We preserve the same routing ratio, $k_{\mathrm{route}} = K/4$, for larger component banks, giving $k_{\mathrm{route}} = 4$ for $K = 16$ and $k_{\mathrm{route}} = 8$ for $K = 32$.

MOELoRA. MOELoRA (Liu et al., 2024) uses a single linear gate shared across all adapted layers. Given the same task view $\mathbf{v}$, its expert weights are computed as

$$\mathbf{w}^{\mathrm{MOE}} = \mathrm{Softmax}\left(\mathbf{W}_T\mathbf{v}\right), \tag{17}$$

where $\mathbf{W}_T \in \mathbb{R}^{K\times 32}$ is the trainable gating matrix shared by all adapted layers, and $\mathbf{w}^{\mathrm{MOE}} \in \mathbb{R}^K$ contains the weights of the K LoRA experts. Consequently, a given task uses the same expert weights across adapted layers, while the low-rank factors of the experts remain layer-specific.

MoSLD. MoSLD (Zhao et al., 2025) uses token-level routing rather than the task-level conditioning used by the methods above. At each adapted layer, a bias-free linear router operates on the token hidden state. Routing probabilities are obtained by softmax, after which the top-k experts are retained and their weights are renormalized.

Within each adapted layer, MoSLD assigns an independent input-side low-rank factor to each expert while sharing the output-side factor across experts. This factorization is designed to separate expert-specific information from features shared across experts (Zhao et al., 2025); dropout is applied to the shared factor during training. We follow this factor-sharing parameterization in our implementation.

Because the output-side factor is shared, its low-rank parameter count differs from a bank of K independent rank-r components. Ignoring the router, the two parameterizations contain

$$Kr(n_{\mathrm{in}} + n_{\mathrm{out}}) \qquad \text{and} \qquad Kr\,n_{\mathrm{in}} + r\,n_{\mathrm{out}}, \tag{18}$$

respectively. We therefore use MoSLD's backbone-specific expert configuration in the cross-backbone comparison.

Layer-shared and layer-specific Gaussian fields. The *Shared* setting in Table 1 determines whether the Gaussian geometry is shared across adapted layers. The task-view projection $\mathbf{W}_{\mathrm{view}}$ remains shared in both settings.

When the Gaussian field is shared, all adapted layers use the same primitive centers, precision matrices, and strengths:

$$\{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k, s_k\}_{k=1}^K. \tag{19}$$

For a given task view, this shared geometry produces the same primitive weights $\{w_k\}_{k=1}^K$ at every adapted layer. The low-rank factors remain layer-specific, so the update at layer ℓ is

$$\Delta\mathbf{W}_\ell = \sum_{k=1}^{K} w_k\,\mathbf{B}_{\ell,k}\mathbf{A}_{\ell,k}, \tag{20}$$

where ℓ indexes the adapted layer.

In the layer-specific setting, each adapted layer learns its own Gaussian geometry,

$$\{\boldsymbol{\mu}_{\ell,k}, \boldsymbol{\Lambda}_{\ell,k}, s_{\ell,k}\}_{k=1}^{K}. \tag{21}$$

The same task view can therefore produce different primitive weights $w_{\ell,k}$ at different layers, giving

$$\Delta \mathbf{W}_\ell = \sum_{k=1}^{K} w_{\ell,k}\, \mathbf{B}_{\ell,k} \mathbf{A}_{\ell,k}. \tag{22}$$

Thus, the shared setting uses one common task–primitive relationship across the network, while the layer-specific setting allows this relationship to adapt independently at each layer.

B.1 Train- and test-template evaluation

For each task, we evaluate on the same validation examples using two prompt-template sets: the three prompt templates used during training (*train templates*) and two disjoint prompt templates reserved for testing (*test templates*). This setup evaluates generalization across different prompt formulations of the same task while keeping the evaluation examples fixed.

The task-level metric follows the task metadata. For classification tasks, we use exact-match accuracy: the decoded prediction is lowercased and stripped of leading and trailing whitespace before comparison with the reference. For generation tasks, we use ROUGE-L F-score with stemming (Lin, 2004). Predictions are obtained with teacher forcing: we take the argmax of the next-token logits at the labeled target positions, conditioned on the reference prefix. We compute 95% confidence intervals for the mean test-score difference between Geofield-LoRA and each baseline with a paired task-level bootstrap over the T tasks, using 200,000 resamples and reporting the 2.5th and 97.5th percentiles.

Let $s_{t,v} \in [0, 1]$ denote the resulting score for task t under template set $v \in \{\text{train}, \text{test}\}$. We report the task-balanced score

$$S_v = \frac{1}{N} \sum_{t=1}^{N} s_{t,v},$$

scaled by 100 in all tables. The train–test gap is

$$\text{Gap} = 100\,(S_{\text{train}} - S_{\text{test}})\,.$$

Each task therefore contributes equally to the aggregate score regardless of its number of evaluation examples.

C Geometric Analysis Details

C.1 Geometry Metrics

To characterize how the Gaussian field changes during training, we measure four aspects of its geometry: primitive anisotropy, view-axis alignment, task–primitive Mahalanobis distance, and center displacement.

Let L denote the number of adapted layers, T the number of tasks, and K the number of primitives in each field. For task t, let $\mathbf{v}_t$ denote its task view. For primitive k in adapted layer ℓ, let $\boldsymbol{\mu}_{\ell,k}$ and $\boldsymbol{\Lambda}_{\ell,k}$ denote its center and precision matrix, respectively. The corresponding covariance matrix is

$$\boldsymbol{\Sigma}_{\ell,k} = \boldsymbol{\Lambda}_{\ell,k}^{-1}. \tag{23}$$

Primitive anisotropy (A). We measure the anisotropy of each primitive using the ratio between the largest and smallest eigenvalues of its covariance matrix. The ratios are aggregated using a geometric mean within each Gaussian field and then averaged across adapted layers:

$$A = \frac{1}{L} \sum_{\ell=1}^{L} \exp\left[\frac{1}{K} \sum_{k=1}^{K} \log \frac{\lambda_{\max}(\boldsymbol{\Sigma}_{\ell,k})}{\lambda_{\min}(\boldsymbol{\Sigma}_{\ell,k})}\right]. \tag{24}$$

An isotropic primitive has an eigenvalue ratio of one, while larger values indicate stronger anisotropy. Since the metric is defined using covariance eigenvalues, it corresponds to the squared ratio between the longest and shortest standard-deviation axes.

View-axis alignment ($\mathrm{A_l}$)**.** We measure how strongly the principal axes of the Gaussian primitives align with the inward task-view directions. Let $\mathbf{u}_{\ell,k}$ denote the unit eigenvector associated with the largest eigenvalue of $\mathbf{\Sigma}_{\ell,k}$, corresponding to the longest axis of the primitive. For task t, the inward direction is

$$\mathbf{d}_t = -\frac{\mathbf{v}_t}{\|\mathbf{v}_t\|_2}. \tag{25}$$

The alignment metric is

$$\mathrm{A_l} = \frac{1}{LTK} \sum_{\ell=1}^{L} \sum_{t=1}^{T} \sum_{k=1}^{K} \left| \mathbf{d}_t^\top \mathbf{u}_{\ell,k} \right|. \tag{26}$$

Thus, $\mathrm{A_l}$ is the mean absolute cosine similarity between the inward task-view directions and the primitive principal axes, averaged over all task–primitive pairs and adapted layers. The absolute value accounts for the sign ambiguity of an eigenvector.

Task–primitive Mahalanobis distance (M)**.** We measure the distance from each task view to its nearest primitive under the Gaussian geometry:

$$M = \frac{1}{LT} \sum_{\ell=1}^{L} \sum_{t=1}^{T} \min_{k} \sqrt{(\mathbf{v}_t - \boldsymbol{\mu}_{\ell,k})^\top \mathbf{\Lambda}_{\ell,k} (\mathbf{v}_t - \boldsymbol{\mu}_{\ell,k})}. \tag{27}$$

For each task and adapted layer, the nearest primitive is determined by its Mahalanobis distance. Because the precision matrix is included in the distance, M reflects both the relative positions of task views and primitives and the learned Gaussian shape.

Center displacement (C)**.** Finally, we measure how far the primitive centers move from their initialization:

$$C = \frac{1}{LK} \sum_{\ell=1}^{L} \sum_{k=1}^{K} \left\| \boldsymbol{\mu}_{\ell,k}^{\mathrm{trained}} - \boldsymbol{\mu}_{\ell,k}^{(0)} \right\|_2. \tag{28}$$

Primitive indices are kept fixed when computing the displacement, so each trained center is compared with its own initialization.

For each run, we compute the four statistics from its task views and Gaussian fields. We then average each statistic across runs. The changes reported in the main paper are

$$\Delta X = X_{\mathrm{trained}} - X_{\mathrm{init}}, \qquad X \in \{A, \mathrm{A_l}, M, C\}. \tag{29}$$

C.2 Geometric Component Controls

We construct three controlled variants to examine the three geometric choices in Geofield-LoRA: ray-based querying, learnable Gaussian geometry, and full anisotropic precision. Each variant changes one of these components while keeping the remaining model structure unchanged.

Point-Query. Geofield-LoRA evaluates each Gaussian at the strongest-response point along the inward ray defined by the task view. Point-Query instead evaluates the Gaussian directly at the task-view coordinate $\mathbf{v}$:

$$q_k^{\mathrm{point}} = s_k \exp\left[-\frac{1}{2} (\mathbf{v} - \boldsymbol{\mu}_k)^\top \mathbf{\Lambda}_k (\mathbf{v} - \boldsymbol{\mu}_k) \right], \tag{30}$$

where q_k^{point} denotes the resulting response of primitive k. The task views, Gaussian parameters, low-rank factors, and normalization across primitives remain unchanged. This variant therefore isolates the effect of querying the Gaussian field along the inward ray.

Fixed geometry. This variant uses the same ray-based query as Geofield-LoRA but keeps the Gaussian geometry fixed at its initialization throughout training. The low-rank factors remain trainable. It therefore measures the contribution of allowing the primitive geometry to reorganize together with the task-specific adaptation.

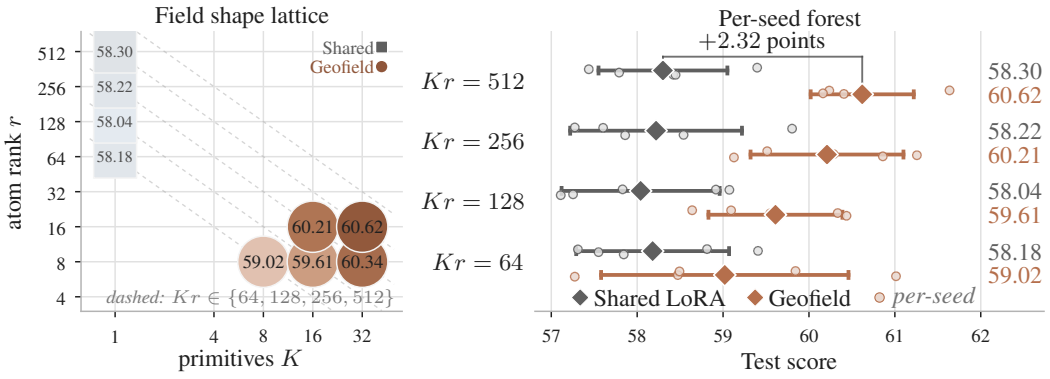

Figure A1: **Matched-budget capacity scaling.** *Left:* test score of Geofield-LoRA across different primitive counts K and ranks r; dashed diagonals indicate matched low-rank budgets $Kr \in \{64, 128, 256, 512\}$. Shared LoRA is shown at $K = 1$ for the corresponding budgets. *Right:* Per-seed test score for Shared LoRA and Geofield-LoRA at each budget, with diamonds indicating the mean and bars showing ± 1 standard deviation.

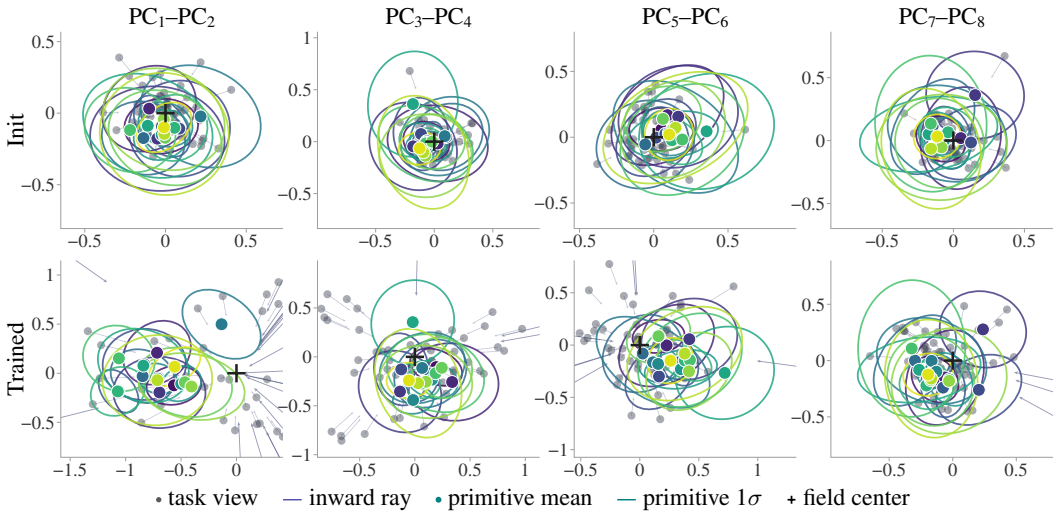

Figure A2: Gaussian primitive field before and after training. PCA projections of the same adapted layer using a shared basis. Columns show paired projections from PC1–PC2 through PC7–PC8. Rows show initialization and the trained field; ellipses denote projected 1σ Gaussian supports.

Diagonal precision. This variant keeps the Gaussian geometry learnable but restricts each precision matrix $\mathbf{\Lambda}_k$ to a diagonal form. Each task-view dimension can still learn its own Gaussian scale, but the primitive can no longer learn a rotated anisotropic support. This isolates the contribution of the full precision matrix to the learned directional structure.

D Additional Analysis

Capacity scaling. Figure A1 extends the controlled comparison in Section 4.2. At a fixed rank $r = 8$, increasing the number of primitives from $K = 8$ to 16 and 32 raises the test score from 59.02 to 59.61 and 60.34. At the same budget $Kr = 256$, 32 primitives of rank 8 (60.34) slightly outperform 16 primitives of rank 16 (60.21), so adding primitives is at least as effective as increasing their rank. The mean gain over Shared LoRA grows from 0.84 points at $Kr = 64$, where it is smaller than the seed-level standard deviation of Geofield-LoRA (± 1.44), to 2.32 points at $Kr = 512$.

Field geometry. Figure A2 shows one adapted layer before and after training in a shared PCA basis. After training, several primitives move away from the field center and their 1σ ellipses become more

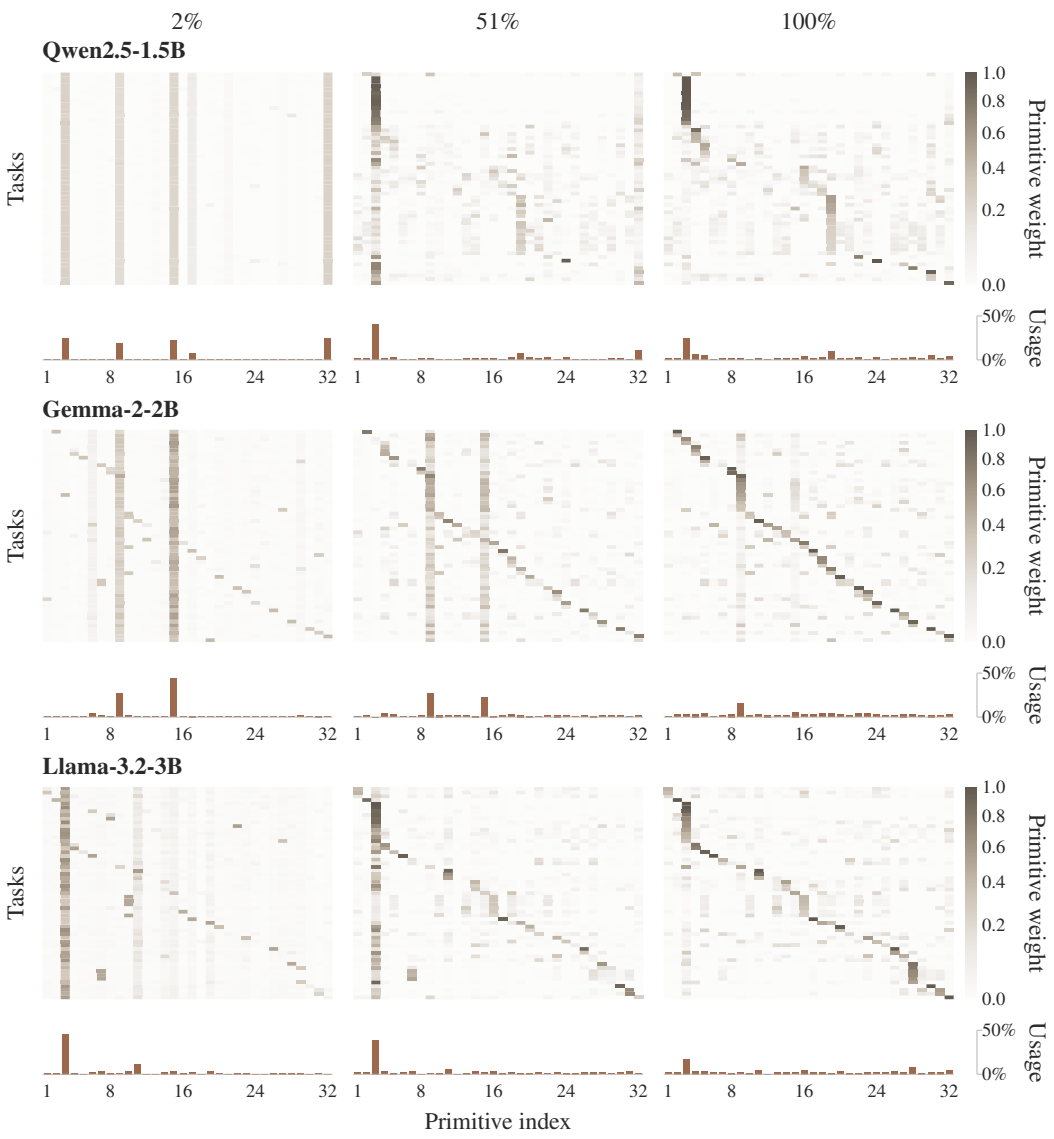

Figure A3: **Evolution of Geofield-LoRA's combination weights during training.** Columns show checkpoints at approximately 2%, 51%, and 100% of training, for the same projection as in Figure 2. Tasks keep the final ordering across columns.

elongated, consistent with the increases in center displacement and anisotropy in Table 4. The task views also spread out, since $\mathbf{W}_{\text{view}}$ is trained jointly with the field.

Combination weights during training. Figure A3 shows the combination weights of Figure 2 at approximately 2%, 51%, and 100% of training. At 2%, most tasks share the same few primitives: only 2, 12, and 15 primitives are dominant for any task on Qwen, Gemma, and Llama, and the task diversity is 0.00, 0.20, and 0.32. Both quantities increase during training, to 13, 24, and 19 primitives and a diversity of 0.52, 0.55, and 0.60 at 51%, and to 16, 26, and 25 primitives and a diversity of 0.79, 0.87, and 0.86 at the end.

D.1 Task-Specific LoRA Update Analysis

We provide additional details for the task-specific LoRA update analysis in Sec. 4.3. We compare Geofield-LoRA, MixLoRA, and LoRA-MLP at $K = 32$ and $r = 16$. The analysis covers all $T = 56$ tasks and all adapted query and value projections.

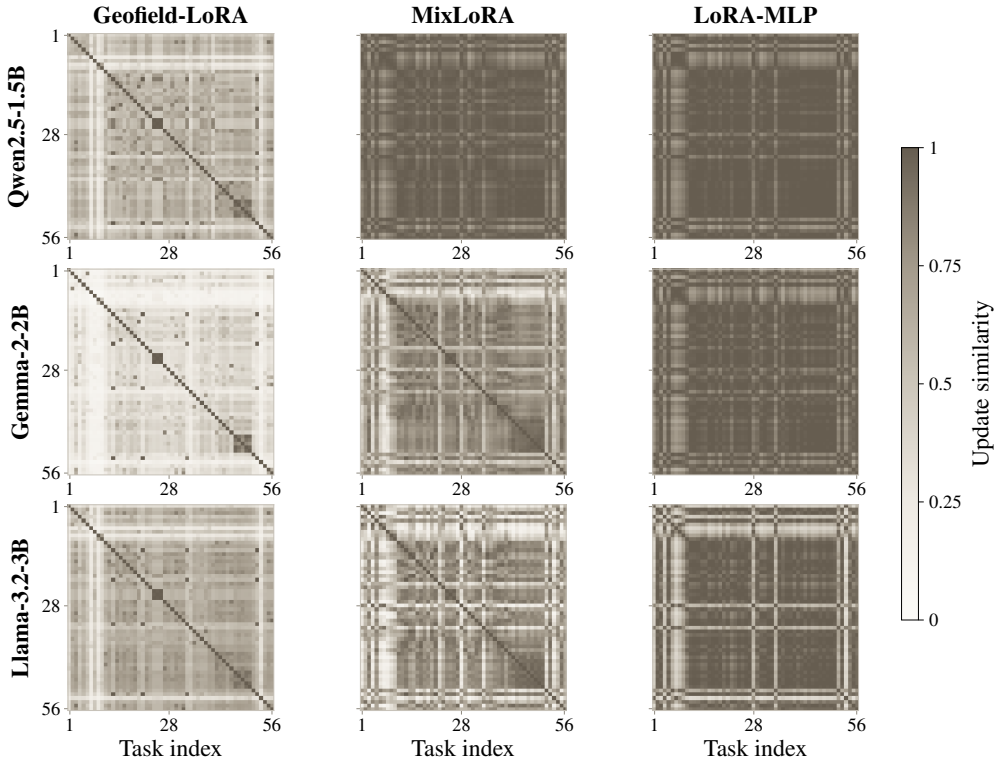

Figure A4: **Pairwise similarity of task-specific LoRA updates.** Each cell shows $R_{t,s}$ from Eq. 32 for a pair of tasks. Rows correspond to backbones and columns to adaptation methods. All panels use the same color scale.

Similarity between task-specific LoRA updates. For task t at adapted projection ℓ, the resulting LoRA update is

$$\Delta \mathbf{W}_{\ell,t} = \sum_{k=1}^{K} w_{\ell,t,k}\, \mathbf{B}_{\ell,k}\mathbf{A}_{\ell,k}. \tag{31}$$

We compare two tasks using the cosine similarity between their LoRA updates within each adapted projection and then average across the L adapted projections:

$$R_{t,s} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{\langle \Delta \mathbf{W}_{\ell,t}, \Delta \mathbf{W}_{\ell,s} \rangle_F}{\|\Delta \mathbf{W}_{\ell,t}\|_F \, \|\Delta \mathbf{W}_{\ell,s}\|_F}, \tag{32}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product. The mean cosine reported in Table 3 is the average of $R_{t,s}$ over all distinct task pairs. Because each update is normalized within its projection, this measure compares update directions independently of their magnitudes.

Figure A4 shows the complete pairwise similarity matrices underlying the mean-cosine values in Table 3. Geofield-LoRA produces less uniformly aligned task updates across all three backbones, whereas LoRA-MLP shows high similarity for most task pairs. MixLoRA produces more task-dependent variation than LoRA-MLP, but its updates remain more aligned than those of Geofield-LoRA.

PCA of task-specific LoRA updates. We next analyze how the differences between task-specific LoRA updates are distributed across principal directions. For each task, we first normalize the update within every adapted projection, flatten the normalized matrices, and concatenate them into a single vector:

$$\mathbf{z}_t = \frac{1}{\sqrt{L}} \begin{bmatrix} \mathrm{vec}(\Delta \mathbf{W}_{1,t})/\|\Delta \mathbf{W}_{1,t}\|_F \\ \vdots \\ \mathrm{vec}(\Delta \mathbf{W}_{L,t})/\|\Delta \mathbf{W}_{L,t}\|_F \end{bmatrix}. \tag{33}$$

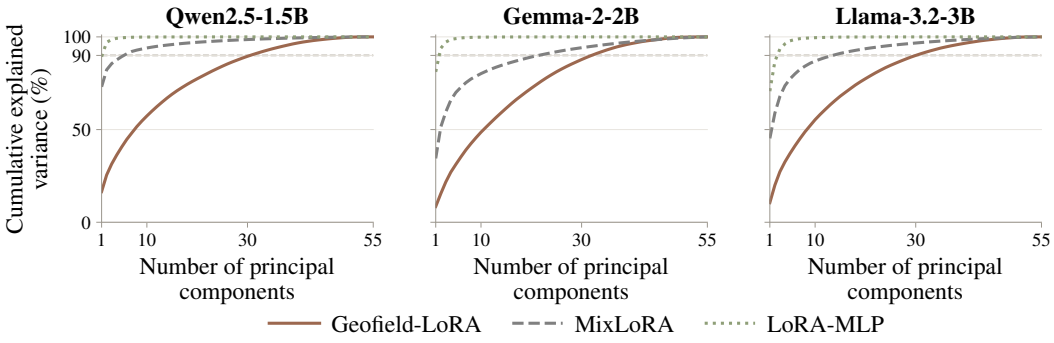

Figure A5: **Distribution of task-specific LoRA update variation.** Cumulative variation explained by the leading principal directions across the 56 centered task-update vectors. A more gradual curve indicates that task-to-task differences are distributed across more principal directions.

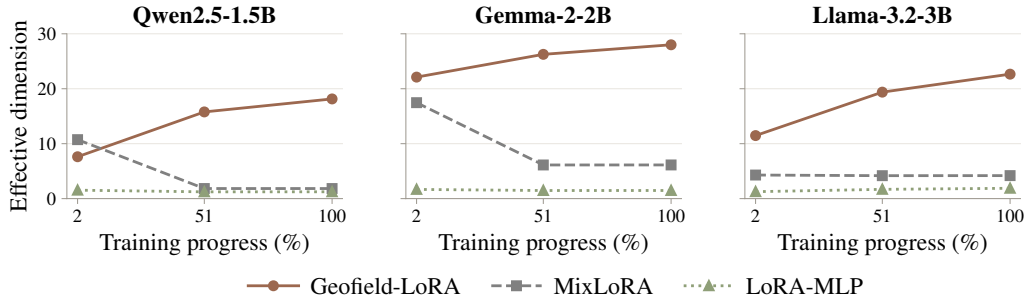

Figure A6: **Evolution of LoRA update structure.** Effective dimension at approximately 2%, 51%, and 100% of training for Geofield-LoRA, MixLoRA, and LoRA-MLP across three backbones.

We then subtract the mean across the $T = 56$ tasks,

$$\widetilde{\mathbf{z}}_t = \mathbf{z}_t - \frac{1}{T}\sum_{s=1}^{T}\mathbf{z}_s, \tag{34}$$

and perform PCA on the centered task vectors. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{T-1} \geq 0$ denote the resulting nonzero eigenvalues.

We use three quantities to describe this spectrum. The fraction explained by the leading principal direction is

$$p_1 = 100\,\frac{\lambda_1}{\sum_i \lambda_i}. \tag{35}$$

The cumulative fraction explained by the first m directions is

$$F(m) = \frac{\sum_{i=1}^{m}\lambda_i}{\sum_i \lambda_i}, \tag{36}$$

and the effective dimension is

$$d_{\text{eff}} = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2}. \tag{37}$$

A larger first-mode ratio indicates that task-dependent variation is more strongly concentrated in one direction, whereas a larger d_{eff} indicates that the variation is distributed across more principal directions.

Figure A5 shows the full PCA spectra summarized by the first-mode ratio and effective dimension in Table 3. Across all three backbones, Geofield-LoRA exhibits a more gradual cumulative spectrum, while MixLoRA and LoRA-MLP concentrate a larger fraction of their task-dependent variation in the leading directions.

Table A1: **Normalized sum vs. front-to-back alpha compositing.** Test score at $Kr = 512$.

Backbone	Normalized sum	Alpha compositing	Δ
Qwen2.5-1.5B	60.62	58.99	−1.63
Gemma-2-2B	67.10	62.09	−5.01
Llama-3.2-3B	66.20	65.14	−1.06

Table A2: **Softmax temperature sweep for LoRA-MLP.** Qwen2.5-1.5B, $Kr = 512$.

Temperature	Test ↑	$H/\log K$
1.00	58.56	0.955
0.50	58.94	0.944
0.25	58.75	0.920

Efficient computation. The analysis above can be computed directly from the low-rank factors without explicitly materializing the full update matrices. For adapted projection ℓ, let $\mathbf{M}_\ell \in \mathbb{R}^{T\times K}$ contain the task-specific combination weights, and define the primitive-update Gram matrix

$$[\mathbf{G}_\ell]_{ij} = \langle \mathbf{B}_{\ell,i}\mathbf{A}_{\ell,i}, \mathbf{B}_{\ell,j}\mathbf{A}_{\ell,j}\rangle_F \,. \tag{38}$$

The inner products between the task-specific updates are then

$$\mathbf{Q}_\ell = \mathbf{M}_\ell \mathbf{G}_\ell \mathbf{M}_\ell^\top, \tag{39}$$

where $[\mathbf{Q}_\ell]_{t,s} = \langle \Delta\mathbf{W}_{\ell,t}, \Delta\mathbf{W}_{\ell,s}\rangle_F$. Thus, the per-projection cosine similarities in Eq. 32 can be obtained from $\mathbf{Q}_\ell$ without constructing dense LoRA updates.

After averaging the normalized per-projection similarities, the resulting matrix is the Gram matrix of the task vectors in Eq. 33. Centering this Gram matrix therefore gives the same nonzero PCA spectrum as explicitly performing PCA on the concatenated task vectors. The training-time analysis in Fig. A6 applies the same computation to the checkpoints at approximately 2%, 51%, and 100% of training.

D.2 Alpha compositing

We also tested a more direct transfer of 3DGS that combines primitive responses along the inward ray by front-to-back alpha compositing instead of the normalized sum in Eq. 11. This variant lowers the test score on all three backbones, by 1.63, 5.01, and 1.06 points (Table A1). We suspect that this drop is mainly caused by the depth ordering required by alpha blending. As task views and primitive centers are updated during training, the order of primitives along a task ray changes, and each swap changes the combination discontinuously: for two primitives with opacities α_i and α_j, the weights jump from $(\alpha_i, (1-\alpha_i)\alpha_j)$ to $((1-\alpha_j)\alpha_i, \alpha_j)$. In 3DGS such changes cause popping artifacts; here they directly alter $\Delta\mathbf{W}$, and with only K primitives per layer a single swap can noticeably change the update. Since the task-view space has no physical occlusion, Geofield-LoRA uses an order-independent normalized sum instead.

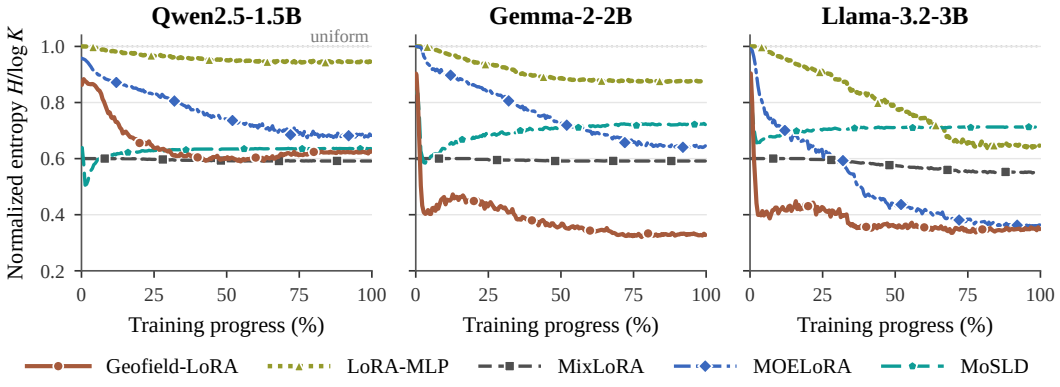

Figure A7: **Routing entropy during training** ($Kr = 512$). Normalized entropy $H/\log K$ of the combination weights, averaged over tasks and adapted layers; the dotted line marks uniform weights. For Geofield-LoRA, the weights are the normalized primitive responses in Eq. 11. Under top-k routing, MixLoRA is bounded by $\log k_{\text{route}}/\log K = 0.6$. MoSLD's entropy is averaged over tokens. Curves are averaged within bins of 0.5% of training. Entropy is reported as a diagnostic of optimization rather than a measure of adaptation quality.

D.3 Optimization of Conditional Baselines

To check that the conditional baselines are properly optimized under the shared training schedule, we track normalized entropy to characterize the evolution of the conditional combination weights during training (Figure A7). We use this quantity only as an optimization diagnostic; a lower entropy does not by itself indicate better adaptation.

The entropy of both dense gates decreases steadily during training on all three backbones. At the end of training, LoRA-MLP reaches 0.95, 0.88, and 0.64 on Qwen2.5-1.5B, Gemma-2-2B, and Llama-3.2-3B, respectively, and MOELoRA reaches 0.68, 0.64, and 0.36. MixLoRA renormalizes the weights over the top-k_{route} components, so its entropy is bounded by $\log k_{\text{route}} / \log K = 0.6$ for $K = 32$ and $k_{\text{route}} = 8$. Its routing therefore changes mainly through which components are selected rather than through the weights within the selected set. MoSLD changes rapidly early in training and then stabilizes.

We further examine whether LoRA-MLP is limited by its softmax temperature. Table A2 reports a temperature sweep on Qwen2.5-1.5B at $Kr = 512$. Lowering the temperature from 1.0 to 0.25 reduces the entropy only from 0.955 to 0.920, consistent with the fact that the final linear layer of the gate can rescale its logits and largely absorb the temperature. The test score varies by at most 0.38 points across temperatures, which is within the seed-level standard deviation of LoRA-MLP in Table 2 (± 0.92). We therefore use $T = 1$ in all experiments.

D.4 Task Types

For the analysis in Section 4.3, we assign each task to one of nine task types, listed in Table A3.

Table A3: Task types used to group the tasks in Section 4.3. Each task is assigned to one type.

Type	Tasks
Sentiment / emotion	amazon_polarity, emotion, glue__sst2, imdb, poem_sentiment, rotten_tomatoes, tweet_eval__emotion, tweet_eval__sentiment, yelp_polarity, yelp_review_full
Social-media classification	tweet_eval__hate, tweet_eval__irony, tweet_eval__offensive, tweet_eval__stance_abortion, tweet_eval__stance_atheism, tweet_eval__stance_climate, tweet_eval__stance_feminist, tweet_eval__stance_hillary
Topic, intent & other classification	ag_news, glue__cola, onestop_english, sms_spam, snips_built_in_intents, yahoo_answers_topics
Natural language inference	glue__mnli, glue__qnli, glue__wnli, multi_nli, snli, super_glue__rte, xnli__en
Paraphrase identification	glue__qqp, medical_questions_pairs, paws__labeled_final, paws__unlabeled_final, paws_x__en
QA / reading comprehension	asnq, gutenberg_time, narrativeqa, pubmed_qa__pqa_labeled, squad_v2, super_glue__boolq, super_glue__multirc, wiki_qa
Commonsense / multiple choice	cbt__P, cos_e__v1_0, cos_e__v1_11, qasc, swag__regular
Summarization / generation	aeslc, billsum, cc_news, cnn_dailymail__3_0_0, common_gen, xsum
Word sense / coreference	super_glue__wic, super_glue__wsc_fixed