PhaseLoRA: Role-Specialised Low-Rank Adaptation through Prompt-Completion Phase Splitting

Anonymous ACL submission

Abstract

In Low-Rank Adaptation (LoRA) fine-tuning of instruction-tuned LLMs, a single set of adapters is applied uniformly across all token positions, treating prompt and completion tokens identically. We question whether this is appropriate. The two phases serve semantically different roles: prompt tokens constitute the natural language input to be understood, while completion tokens constitute the task-specific output to be generated. Conflating the two under a single adapter set ignores a meaningful distinction that is intrinsic to the instruction-tuning paradigm. The phases are also computationally different: prompt tokens are conditioned on but never predicted, whereas completion tokens are generated autoregressively based on the preceding context. We introduce PhaseLoRA, an extension of LoRA that maintains independent adapter sets for the prompt and completion phases. Experiments across multiple benchmarks and a range of model scales and LoRA ranks show that PhaseLoRA consistently outperforms standard LoRA in low-capacity and low-rank settings. Beyond benchmark comparisons, we conduct mechanistic analysis of the learned adapter representations, structured ablations on warm-up initialisation and asymmetric rank allocation, and a quantitative error analysis characterising the inputs on which phase specialisation helps most.

1 Introduction

Low-Rank Adaptation (LoRA) (Hu et al., 2021) has become the standard approach for parameter-efficient fine-tuning of instruction-tuned LLMs. In practice, a single set of LoRA adapters is applied uniformly across all token positions during both training and inference, without distinguishing between input prompt and generated completion tokens.

The two phases, however, are computationally asymmetric by nature: prompt tokens are conditioned on but never predicted, while completion tokens are generated autoregressively based on the preceding context. This asymmetry is further reflected during training under standard loss masking, where only completion tokens are directly supervised. More importantly, the two phases are semantically distinct in a way intrinsic to the instruction-tuning paradigm: the prompt is a natural language instruction or query to be read and understood, while the completion is a task-specific response to be generated. The two phases are therefore not just different positions in a sequence but play fundamentally different roles.

This raises a natural question: does enforcing a single shared adapter set across these functionally distinct phases limit representational capacity, and can explicitly separating them yield measurable benefits?

We introduce PhaseLoRA, which maintains two independent adapter sets: one applied during the prompt phase and one during the completion phase. The assignment is deterministic based on token phase, requiring no learned routing and no additional parameters beyond the two adapter sets. PhaseLoRA thus provides a minimal operationalisation of the hypothesis that prompt and completion phases benefit from distinct adapter specialisation in instruction-tuned decoder-based LLMs.

We study this hypothesis on generative information extraction benchmarks, a setting where the prompt–completion distinction is unambiguous: the prompt is a natural language input sentence, and the completion is a structured extraction. We evaluate across three information extraction (IE) tasks, three LLMs of varying scale and three LoRA ranks.

Our results show that PhaseLoRA consistently outperforms standard LoRA under limited model capacity and low adapter rank, with gains of up to 3% F1 and 5% recall. The advantage diminishes both with increasing model scale and with increasing adapter rank, suggesting that higher-capacity

settings allow a single adapter set to sufficiently capture the behaviour of both phases.

Mechanistic analysis via centered kernel alignment (CKA) shows that prompt and completion adapters learn near-orthogonal representations throughout the network. Frobenius norm trajectories further indicate that prompt adapters dominate in early and middle layers, while completion adapters dominate in the final layers. Performance gains are primarily driven by improved recall on annotation-dense inputs, where standard LoRA systematically omits relevant spans. Ablations on warm-up initialisation and asymmetric rank allocation further characterise PhaseLoRA across training and capacity settings.

Our contributions are as follows: (1) we investigate whether prompt and completion tokens benefit from independent adapter specialisation in LoRA fine-tuning; (2) we introduce PhaseLoRA, a minimal, routing-free method that operationalises this idea; (3) we provide mechanistic evidence that phase-specific adapters learn orthogonal representations throughout the network; and (4) we characterise when and why phase specialisation is beneficial through analysis across model scales and LoRA ranks, ablations on warm-up initialisation and rank allocation, and a quantitative error analysis[1].

2 Related Work

Parameter-Efficient Fine-Tuning and LoRA. Parameter-efficient fine-tuning (PEFT) methods adapt pre-trained language models to downstream tasks while updating only a small fraction of parameters (Hu et al., 2021; Houlsby et al., 2019; Lester et al., 2021; Li and Liang, 2021). Among these, Low-Rank Adaptation (Hu et al., 2021) has become the dominant approach, injecting trainable low-rank matrices into frozen transformer weights. Subsequent work has explored adaptive rank allocation (Zhang et al., 2023), quantisation-aware variants (Dettmers et al., 2023) and optimisation improvements (Hayou et al., 2024). In supervised fine-tuning of instruction-tuned models, it is standard practice to mask the loss on prompt tokens and train only on completion tokens (Huerta-Enochian and Ko, 2024), a convention we follow throughout.

Mixture-of-Experts LoRA and Token-Level Routing. A parallel line of work integrates the mixture-of-experts (MoE) paradigm (Shazeer et al., 2017; Fedus et al., 2022) with LoRA, maintaining multiple adapter sets and routing tokens to different experts during the forward pass. MoLoRA (Zadouri et al., 2024) introduces token-level soft routing over multiple LoRA experts for instruction tuning. MoLE (Huang and Wei, 2024) applies learned gating functions per token to compose multiple trained LoRA adapters hierarchically. MoELoRA (Luo et al., 2024) introduces contrastive learning among experts to encourage specialisation and mitigate routing collapse. MixLoRA (Li et al., 2024) constructs sparse MoE architectures from LoRA modules with auxiliary load balancing, while LoRAMoE (Dou et al., 2024) uses MoE-style plugins to mitigate world knowledge forgetting during alignment fine-tuning. Such methods share a common design principle: routing is learned from hidden state representations, introducing additional parameters and training complexity.

Phase-Aware and Position-Sensitive Adaptation. The most closely related work to PhaseLoRA is Activated LoRA (aLoRA) (Greenewald et al., 2026), which activates LoRA adapters only after a designated invocation sequence, leaving preceding tokens to be processed entirely by the base model. aLoRA is motivated primarily by inference efficiency, enabling KV cache reuse across adapter invocations in multi-turn and agentic settings. PhaseLoRA differs in both motivation and design. Rather than disabling adaptation on prompt tokens, it maintains separate adapters for the prompt and completion phases. The goal is not inference efficiency, but to test whether the two phases benefit from distinct adapter specialisation during fine-tuning.

3 Methodology

Figure 1 provides an overview of both methods described in this section.

3.1 Standard LoRA

Standard LoRA adapts a pre-trained language model by injecting trainable low-rank decompositions into selected weight matrices while keeping the base model frozen. For a weight matrix $\mathbf{W} \in R^{d\times k}$, LoRA parametrises the update as $\Delta\mathbf{W} = \mathbf{BA}$, where $\mathbf{A} \in R^{r\times k}$ and $\mathbf{B} \in R^{d\times r}$ with rank $r \ll \min(d, k)$. The adapted forward

[1] Our code will be made available upon paper acceptance.

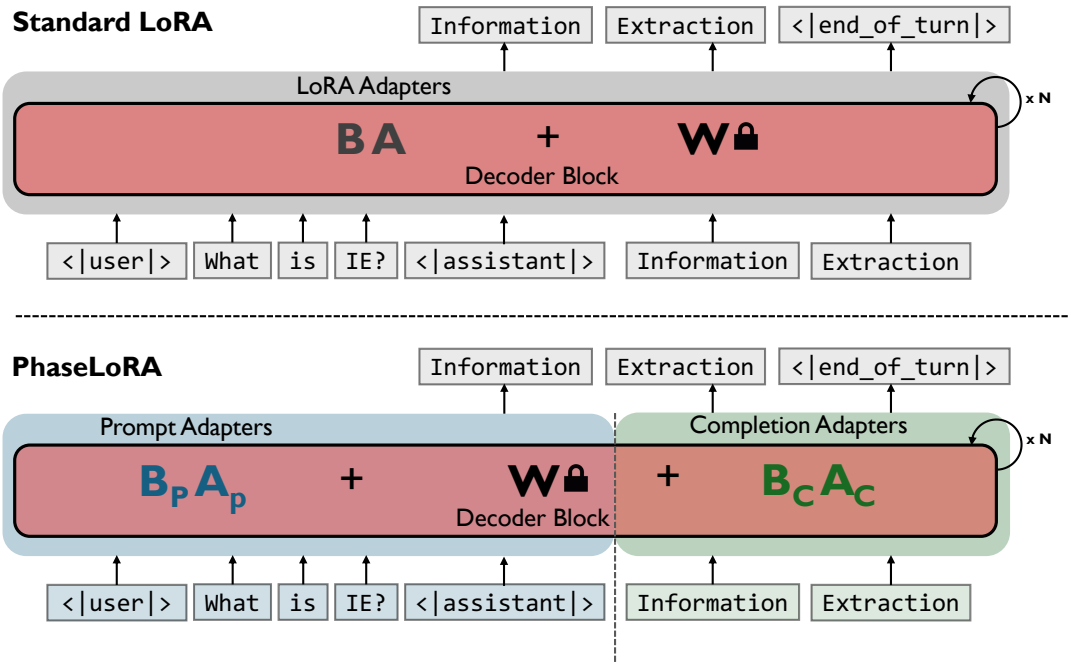

Figure 1: Standard LoRA (top) applies a single adapter set uniformly across all token positions. PhaseLoRA (bottom) maintains separate prompt and completion adapters, with the boundary determined by the assistant start-of-turn marker in the chat template. Base model weights $\mathbf{W}$ are frozen in both cases.

pass for a given input $\mathbf{x}$ is:

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \frac{\alpha}{r}\mathbf{B}\mathbf{A}\mathbf{x} \tag{1}$$

where α is a scaling hyperparameter (Hu et al., 2021).

3.2 PhaseLoRA

PhaseLoRA maintains two independent LoRA adapter sets: a *prompt adapter* $\{\mathbf{A}_p, \mathbf{B}_p\}$ and a *completion adapter* $\{\mathbf{A}_c, \mathbf{B}_c\}$, each of rank r. During the forward pass, the prompt adapter is active for all prompt token positions and the completion adapter for all completion token positions. Formally, for token position t, the adapted output is:

$$\mathbf{h}_t = \mathbf{W}\mathbf{x}_t + \frac{\alpha}{r}\begin{cases}\mathbf{B}_p\mathbf{A}_p\mathbf{x}_t & \text{if } t \in \mathcal{P} \\ \mathbf{B}_c\mathbf{A}_c\mathbf{x}_t & \text{if } t \in \mathcal{C}\end{cases} \tag{2}$$

where $\mathcal{P}$ and $\mathcal{C}$ denote the sets of prompt and completion token positions respectively. The boundary between $\mathcal{P}$ and $\mathcal{C}$ is determined by the chat template of each model: all tokens up to and including the assistant start-of-turn special token (such as `<|assistant|>` from the example in Figure 1) are treated as prompt, and all subsequently generated tokens are treated as completion. This boundary is fully deterministic and requires no additional computation at training or inference time.

Both adapter sets are initialised independently following the standard LoRA initialisation scheme and are trained jointly via a cross-entropy objective with prompt masking. The completion adapter receives gradient signal directly from completion token losses, while the prompt adapter is updated via gradients that backpropagate through the attention mechanism from completion positions into prompt representations. At inference time, the prompt adapter is applied during the prefill stage and the completion adapter is applied during autoregressive decoding. The split point is consistent with the training.

PhaseLoRA at rank r has twice the trainable parameters of Standard LoRA at rank r, but identical per-token adapter capacity, since only one adapter set is active at any position.

4 Experimental Setup

4.1 Tasks and Benchmarks

All benchmarks are formatted as prompt-completion pairs. The prompt is a natural language sentence, and the completion is a JSON array of objects, each containing a text span from the sentence and its associated label. We preprocess each benchmark by removing samples without annotated spans, yielding approximately 3,000 training samples. We evaluate on three IE tasks:

named entity recognition using OntoNotes 5.0 (Weischedel et al., 2013), event detection using ACE2005 (Doddington et al., 2004) and semantic role labelling using NounAtlas (Navigli et al., 2024).

4.2 Models

We use three open-weight, instruction-tuned LLMs: Qwen2.5-3B-Instruct (Qwen et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Mistral-Nemo-Instruct-2407 (Mistral AI, 2024), spanning 3B to 12B parameters, providing diversity across both model family and scale.

4.3 Training Details

LoRA adapters are applied to all attention projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and feed-forward projections (`gate_proj`, `up_proj`, `down_proj`). We evaluate three ranks $r \in \{4, 8, 16\}$ with $\alpha = 2r$ throughout, following recommended practice (Shuttleworth et al., 2026). All hyperparameters are held identical between Standard LoRA and PhaseLoRA. Models are fine-tuned for 3 epochs. Main experiments are run on three random seeds, and we report the mean scores. Further details on hyperparameters and compute are provided in Appendix A.1.

4.4 Evaluation

A prediction is counted as a true positive only if both the extracted span and its associated label exactly match the gold annotation. We report micro-averaged F1 score as the evaluation metric.

5 Results & Discussion

5.1 Overall Comparison: PhaseLoRA vs Standard LoRA

Table 1 reports mean F1 scores (averaged over three seeds) for Standard LoRA and PhaseLoRA across all model, benchmark and rank configurations. PhaseLoRA outperforms Standard LoRA in the large majority of settings. The advantage is most pronounced for Qwen2.5-3B, where PhaseLoRA is consistently better across all benchmarks and ranks, and diminishes with increasing model size, with Mistral-Nemo-12B showing small average improvements and results more often favouring Standard LoRA.

5.2 Effect of Model Scale and LoRA Rank

The benefit of phase specialisation decreases monotonically with both model scale and LoRA rank.

Effect of rank. Averaged across models and benchmarks, ΔF1 decreases monotonically with rank (1.37% at $r = 4$, 0.60% at $r = 8$, 0.37% at $r = 16$). This trend is consistent across all models, with higher ranks systematically reducing the advantage of PhaseLoRA.

Effect of model scale. Averaged across ranks and benchmarks, ΔF1 also decreases monotonically with model scale (1.82% for Qwen2.5-3B, 0.45% for Llama-3.1-8B, 0.07% for Mistral-Nemo-12B). This pattern holds across all benchmarks.

The interaction between scale and rank is also informative: for Qwen2.5-3B, PhaseLoRA remains consistently beneficial even at rank 16, whereas for Mistral-Nemo-12B the advantage disappears already at rank 8, where Standard LoRA outperforms PhaseLoRA across all benchmarks. This suggests that the benefit of explicit phase specialisation diminishes at lower ranks as model capacity increases.

Taken together, these results indicate that phase specialisation provides substantial benefit under limited model capacity and low adapter rank. As both increase, the model appears to develop sufficient representational capacity to handle the functional distinction between prompt and completion phases implicitly, reducing the benefit of making this distinction explicit in the adapter structure.

5.3 Precision and Recall Decomposition

Full precision and recall breakdowns are provided in Appendix A.2. Across all 27 model–benchmark–rank configurations, the sign of the F1 difference between PhaseLoRA and Standard LoRA matches the sign of the recall difference in 26 of 27 cases and matches the sign of the precision difference in 24 of 27 cases. Thus, improvements in F1 are almost always accompanied by improvements in both precision and recall.

The magnitude of these effects, however, is markedly asymmetric. Precision differences remain small across all models, with maximum $|\Delta|$ of 1.24%, 1.24% and 0.70% for Qwen2.5-3B, Llama-3.1-8B and Mistral-Nemo-12B respectively. Recall differences are substantially larger: maximum absolute $|\Delta|$ reaches 4.86%, 1.37% and 1.12% for Qwen2.5-3B, Llama-3.1-8B and Mistral-Nemo-12B respectively. The gains of PhaseLoRA are therefore driven primarily by improvements in recall, despite precision and recall typically moving in the same direction.

ACE2005

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	70.81	**73.51**	2.70 ± 0.71	75.19	**76.49**	1.30 ± 0.74	78.25	**78.77**	0.53 ± 0.12
8	73.42	**75.10**	1.68 ± 0.54	76.06	**76.64**	0.58 ± 0.14	**78.47**	78.02	-0.45 ± 0.28
16	75.26	**75.92**	0.66 ± 0.05	75.94	**76.77**	0.83 ± 0.37	**78.19**	77.57	-0.62 ± 0.25

OntoNotes

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	84.73	**86.89**	2.16 ± 0.21	87.58	**88.03**	0.44 ± 0.43	87.78	**88.32**	0.54 ± 0.38
8	85.75	**87.90**	2.15 ± 0.45	87.98	**88.83**	0.85 ± 0.06	**88.58**	88.46	-0.12 ± 0.07
16	87.19	**88.50**	1.31 ± 0.23	**89.00**	88.91	-0.09 ± 0.32	88.89	**89.29**	0.40 ± 0.15

NounAtlas

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	50.18	**53.24**	3.06 ± 0.05	56.63	**57.48**	0.84 ± 0.06	58.25	**58.98**	0.73 ± 0.51
8	52.68	**54.03**	1.35 ± 0.31	**57.93**	57.41	-0.52 ± 0.78	**58.97**	58.90	-0.07 ± 0.98
16	54.36	**55.72**	1.36 ± 0.84	**58.82**	58.59	-0.23 ± 0.46	**59.92**	59.62	-0.30 ± 0.10

Table 1: Mean F1 scores across three seeds for Standard LoRA and PhaseLoRA at ranks 4, 8 and 16, across three benchmarks and three models. Δ denotes the mean difference (PhaseLoRA $-$ Standard LoRA) $\pm$ standard error across seeds. Bold indicates the higher mean F1 between the two methods for each configuration. Precision and recall breakdowns are provided in Appendix A.2.

5.4 Error Analysis

To investigate the source of PhaseLoRA's recall advantage, we analyse spans correctly predicted by PhaseLoRA but missed by Standard LoRA and examine the annotation density of the examples from which these spans originate. Annotation density is defined as the total number of gold spans in an example. Across a selection of model–benchmark–rank configurations, we compare the mean density of examples contributing PhaseLoRA-exclusive true positives against the remaining test examples (see Table 2).

Across all examined configurations, PhaseLoRA-exclusive true positives originate disproportionately from annotation-dense examples. For Qwen2.5-3B on ACE2005, the mean density of contributing examples is 2.02 versus 1.36 for the remaining test set at rank 4, and 2.21 versus 1.36 at rank 16. In both settings, over 70% of contributing examples contain multiple annotated spans, compared to 28% in the remaining test set. Similar trends persist across ranks and benchmarks. For Mistral-Nemo-12B on OntoNotes, the density gap remains substantial at both rank 4 (3.74 vs 2.49) and rank 16 (4.14 vs 2.49), even though the overall F1 difference in the latter setting is minimal.

Qualitative inspection shows that Standard LoRA's failures on these dense examples are predominantly omissions rather than misclassifications: relevant spans are frequently not predicted at all, as opposed to being assigned incorrect labels or boundaries. This pattern is consistent across configurations. Taken together, these findings suggest that phase-specific adaptation improves the model's ability to exhaustively recover relevant spans in annotation-dense inputs, which contributes to the recall gains observed in Section 5.3.

Model (rank)	Benchmark	Mean Density		%>1 Span	
		Excl.	Rest	Excl.	Rest
Qwen2.5-3B (rank 4)	ACE2005	**2.02**	1.36	71	28
Qwen2.5-3B (rank 16)	ACE2005	**2.21**	1.36	78	28
Mistral-Nemo-12B (rank 4)	OntoNotes	**3.74**	2.49	74	60
Mistral-Nemo-12B (rank 16)	OntoNotes	**4.14**	2.49	82	60

Table 2: Annotation density of examples containing at least one span that PhaseLoRA predicts correctly and Standard LoRA misses (Excl.), versus all remaining test examples (Rest) across selected configurations. Mean Density denotes the mean number of annotated spans per example. %>1 Span denotes the percentage of examples containing more than one annotated span. The higher value in each pair is shown in bold for Mean Density and underlined for %>1 Span.

5.5 Parameter and Inference Efficiency

The primary comparisons in Table 1 hold LoRA rank constant across PhaseLoRA and Standard LoRA. This is the appropriate baseline for assessing the effect of phase specialisation under matched per-token adapter capacity: both methods apply rank-r adapters to each token, and the per-token capacity is therefore identical. PhaseLoRA at rank r does, however, require twice the parameters of Standard LoRA at rank r, since two adapter sets are maintained. For practitioners with a fixed parameter budget, a more relevant comparison is PhaseLoRA at rank r against Standard LoRA at rank $2r$.

We assess this across all model–benchmark com-

binations, yielding 18 comparisons in total (3 models $\times$ 3 benchmarks $\times$ 2 rank pairs: PhaseLoRA r4 vs Standard LoRA r8, and PhaseLoRA r8 vs Standard LoRA r16). Taken together, PhaseLoRA wins in 9 of 18 cases, with neither method consistently dominating, suggesting broad parity under parameter-invariant comparison. Breaking this down by rank pair reveals more nuanced patterns. For the lower rank pair (PhaseLoRA r4 vs Standard LoRA r8), PhaseLoRA wins in 7 of 9 cases. For the higher rank pair (PhaseLoRA r8 vs Standard LoRA r16), Standard LoRA wins in 7 of 9 cases. This pattern is consistent with the capacity story established in Section 5.2: at low rank, explicit phase specialisation provides an advantage that even double the token-level adapter capacity cannot replicate, whereas at higher rank, Standard LoRA accumulates sufficient capacity to accommodate both phases within a single adapter set.

The practical implication is that the appropriate comparison depends on the deployment constraint. Under a matched-rank comparison, PhaseLoRA offers a consistent advantage at smaller model scales and lower ranks. Under a fixed parameter budget, the two methods are broadly comparable, though PhaseLoRA retains an advantage at lower ranks where capacity is most constrained.

5.6 Ablation Studies

5.6.1 Adapter Swap Experiments

A key question is whether the two adapter sets learn distinct representations or converge to functionally equivalent behaviour, which would render phase splitting redundant. To investigate this, we compare four **inference-time** configurations across models and benchmarks: the standard PhaseLoRA configuration, a configuration in which the two adapters are swapped across phases (i.e., the prompt adapter is applied during the completion phase and vice versa) and two configurations in which either adapter is applied uniformly throughout the full sequence.

Swapping the adapters yields entirely incoherent output, characterised by cascading bracket sequences and text bearing no relation to the input (e.g., "`I'm [[[[[[[[[[[[[[...`"). Applying the prompt adapter throughout the sequence produces similarly incoherent output.

The completion-only configuration is divergent in that rather than producing incoherent output, the model generates fluent but task-unaware text resembling the output of an *unadapted* model (e.g., *"I'm sorry, but I don't have enough context to verify the details of this statement..."*). One plausible explanation is that due to the joint training, the completion adapter expects a context already processed by the prompt adapter and in its absence lacks the grounded input representation required to produce task-appropriate output.

Taken together, these results confirm that the two adapter sets are not interchangeable and have developed qualitatively distinct functional roles during training, providing motivation for the representation analyses in Section 6.

5.6.2 Warm-up Initialisation

Qwen2.5 3B Instruct			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
No warmup (0+3)	**75.10**	87.90	54.03
1 epoch warmup (1+2)	74.45	**88.12**	**54.82**
2 epoch warmup (2+1)	71.42	87.77	53.67

Table 3: F1 scores for warm-up initialisation variants across three benchmarks (Qwen2.5-3B, rank 8, single seed). Results for Llama-3.1-8B and Mistral-Nemo-12B are provided in Appendix A.3.

We investigate whether PhaseLoRA benefits from a curriculum-style training procedure in which a single shared adapter is first trained before being used to initialise both phase-specific adapters. All experiments use a fixed training budget of 3 epochs. We compare three configurations: no warm-up (3 epochs of phase-specialised training), 1 epoch of warm-up followed by 2 epochs of phase-specialised training, and 2 epochs of warm-up followed by 1 epoch of phase-specialised training. During warm-up, standard LoRA training is used, after which the adapter weights are copied into both the prompt and completion adapters and phase-specialised training proceeds for the remaining epochs.

Table 3 reports F1 scores for the Qwen2.5-3B model with rank 8 adapters across benchmarks, with corresponding results for Llama-3.1-8B and Mistral-Nemo-12B provided in Appendix A.3. Two clear patterns emerge.

First, extended warm-up is consistently harmful. The 2-epoch warm-up configuration underperforms the no-warm-up baseline in 8 of 9 combinations, with drops of up to 3.68% F1. The effect is strongest for Qwen2.5-3B, while Llama-3.1-8B and Mistral-Nemo-12B exhibit smaller but still consistent declines.

Second, minimal warm-up is largely neutral to mildly beneficial. The 1-epoch warm-up config-

uration outperforms the no-warm-up baseline in 7 of 9 combinations, although the gains are generally small, with the largest improvement being 1.91% F1 on Mistral-Nemo-12B for NounAtlas. Importantly, the 1-epoch warm-up consistently outperforms the 2-epoch warm-up across all configurations, indicating that performance deteriorates progressively as warm-up duration increases.

These results suggest that prolonged shared training is not a useful inductive bias for PhaseLoRA. Specialising the adapters from the outset, or after only minimal shared initialisation, consistently matches or outperforms longer warm-up schedules. One possible explanation is that extended shared training encourages the two adapters to converge towards similar representations early in optimisation, making subsequent phase-specific divergence more difficult. This interpretation is consistent with the near-zero CKA similarity between prompt and completion adapters reported in Section 6.1, which indicates that phase specialisation is associated with strongly divergent representations.

5.6.3 Asymmetric Rank Allocation

Qwen2.5 3B Instruct			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
r8+r8	75.10	87.90	54.03
r4+r12	74.04	87.64	**55.35**
r12+r4	**75.14**	**88.76**	54.85

Table 4: F1 scores for asymmetric rank allocation across three benchmarks (Qwen2.5-3B, single seed). Results for Llama-3.1-8B and Mistral-Nemo-12B are provided in Appendix A.3.

Because PhaseLoRA maintains independent adapters for each phase, it affords a degree of freedom not available in standard LoRA: the ranks of the prompt and completion adapters can be set independently. Rather than constraining $r_p = r_c$, rank can be allocated asymmetrically across phases while holding the total parameter budget fixed, i.e. keeping $r_p + r_c$ constant. This isolates the effect of capacity allocation from the effect of total capacity.

Table 4 reports F1 scores for symmetric and asymmetric rank allocations under a fixed total rank budget of $r_p + r_c = 16$, with results for Qwen2.5-3B shown in the main paper and results for Llama-3.1-8B and Mistral-Nemo-12B provided in Appendix A.3. Across all three models, the prompt-heavy allocation ($r_p = 12, r_c = 4$) outperforms the symmetric baseline ($r_p = 8, r_c = 8$) in 6 of 9 model–benchmark combinations and outperforms the completion-heavy allocation ($r_p = 4, r_c = 12$) in 6 of 9 combinations. By contrast, the completion-heavy allocation outperforms the symmetric baseline in only 4 of 9 combinations. Overall, performance follows a general ordering in which prompt-heavy allocations perform best, followed by symmetric allocation, with completion-heavy allocations performing worst.

This pattern is consistent with the Frobenius norm analysis in Section 6.2, where the prompt adapter exhibits larger norms throughout most of the network depth, suggesting that the prompt phase benefits from greater representational capacity. Allocating additional rank to the prompt adapter appears to improve performance.

6 Representation Analysis

The swap experiments in Section 5.6.1 establish that the prompt and completion adapters develop functionally distinct roles during training. We now examine the learned adapter representations directly to characterise the nature of this specialisation. All analyses are conducted on three model–benchmark–rank configurations: Qwen2.5-3B on ACE2005 at rank 4, Llama-3.1-8B on OntoNotes at rank 8 and Mistral-Nemo-12B on NounAtlas at rank 16.

6.1 Centered Kernel Alignment Between Phase Adapters

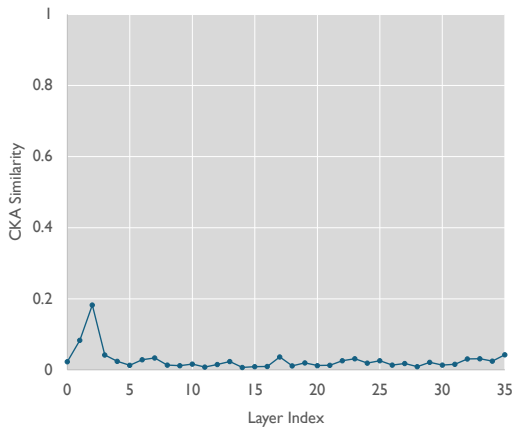

Figure 2: CKA similarity between prompt and completion adapters per layer, averaged across all adapted modules (Qwen2.5-3B, ACE2005, rank 4). Results are consistent across all three configurations (Appendix A.4).

To assess whether the prompt and completion adapters learn distinct representations, we compute the Centered Kernel Alignment (CKA) (Kornblith et al., 2019) between the prompt and completion adapter matrices $\mathbf{B}_p\mathbf{A}_p$ and $\mathbf{B}_c\mathbf{A}_c$ at each layer,

averaged across all adapted modules. CKA is a normalised similarity measure invariant to orthogonal transformations and isotropic scaling, making it well-suited for comparing learned representations across adapter sets.

Figure 2 shows the layer-wise CKA similarity for Qwen2.5-3B on ACE2005 at rank 4. The similarity remains below 0.1 throughout most of the network, with no discernible trend across layers (one layer reaches 0.18, though this is low in absolute terms and not seen in other models; see Appendix A.4). Results are consistent across all three model–benchmark–rank configurations (Appendix A.4). The near-zero CKA throughout the full depth of the network indicates that the two adapter sets learn fundamentally orthogonal transformations at every layer, not just in the deeper layers where generation is most active. This provides strong empirical support for the phase specialisation hypothesis: even under a shared training objective and identical initialisation scheme, the prompt and completion adapters diverge into distinct representations.

6.2 Frobenius Norm Trajectories

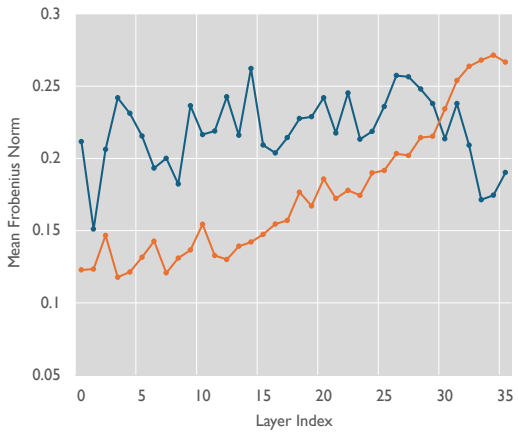

Figure 3: ● Prompt and ● completion adapter Frobenius norms per layer, averaged across all adapted modules (Qwen2.5-3B, ACE2005, rank 4). Results are consistent across all three model–benchmark–rank configurations (Appendix A.4).

While CKA establishes representational divergence between the two adapter sets, it does not indicate how adaptation strength is distributed across the network depth. To examine this, we compute the Frobenius norm $\|\mathbf{BA}\|_F$ at each layer, averaged across all adapted modules, separately for the prompt and completion adapters.

Figure 3 shows the layer-wise norm trajectories for Qwen2.5-3B on ACE2005 at rank 4, with the same pattern observed across all configurations (Appendix A.4). The prompt adapter exhibits substantially larger norms throughout the majority of the network, indicating that prompt-phase adaptation dominates most layers of the transformer stack. The completion adapter norm, by contrast, increases progressively with depth, eventually overtaking the prompt adapter between 71–87% of network depth across configurations. Beyond this intersection point, the completion adapter norm consistently exceeds that of the prompt adapter in the deepest layers. This suggests a transition in the dominant role of adaptation across the network: earlier and middle layers are shaped more strongly by prompt-phase processing, whereas the deepest layers are increasingly specialised towards token generation during the completion phase.

Analysis of per-module norms (averaged across layers) further shows that `gate_proj` and `up_proj` consistently exhibit substantially larger Frobenius norms than all other adapted modules across configurations and phases (Appendix A.4), suggesting that the largest task-specific adaptations occur within the feed-forward sublayers rather than the attention projections, though we leave deeper investigation of this to future work.

7 Conclusion

We investigated whether explicitly decoupling LoRA adapter weights across prompt and completion phases yields measurable benefit over uniform adaptation. PhaseLoRA, our minimal instantiation of this idea, consistently outperforms standard LoRA under small-model and low-rank conditions, with the benefit diminishing with increasing model scale and adapter rank. Performance gains are driven primarily by improved recall on annotation-dense inputs, where standard LoRA tends to omit spans entirely. Under parameter-controlled comparison, PhaseLoRA at rank r outperforms Standard LoRA at rank $2r$ at lower ranks, suggesting that phase specialisation yields better performance per parameter than standard LoRA in low-rank regimes. Mechanistic analysis confirms that the two adapter sets learn near-orthogonal representations throughout the network despite sharing a training objective. Ablation studies show that extended warm-up initialisation is detrimental to phase specialisation, and that allocating greater rank to the prompt adapter yields a marginal but consistent advantage over symmetric allocation.

Limitations

This work evaluates exclusively on generative information extraction benchmarks, where the prompt-completion distinction is unambiguous. Whether phase-specialised adaptation yields similar benefits in more open-ended settings such as dialogue, summarisation or instruction following remains an open question. Our experiments are limited to models up to 12B parameters and relatively low LoRA ranks. Although the observed trends suggest that the benefit of phase separation diminishes with increasing capacity, we do not evaluate whether this continues at larger scales. All experiments use a single-turn prompt-completion formulation. Although the phase boundary remains definable in multi-turn settings via the chat template, we do not investigate whether phase-specific adaptation is beneficial across extended conversational contexts.

References

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

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. 2024. 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 (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kristjan Greenewald, Luis Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Cox. 2026. Activated lora: Fine-tuned llms for intrinsics. *Advances in Neural Information Processing Systems*, 38:53022–53049.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17783–17806. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Shaohan Huang and Furu Wei. 2024. Mixture of lora experts. In *ICLR 2024*.

Mathew Huerta-Enochian and Seung Yong Ko. 2024. Instruction fine-tuning: Does prompt loss matter? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22771–22795, Miami, Florida, USA. Association for Computational Linguistics.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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

Mistral AI. 2024. Mistral nemo. Accessed: 2025-05-05.

Roberto Navigli, Marco Lo Pinto, Pasquale Silvestri, Dennis Rotondi, Simone Ciciliano, and Alessandro Scirè. 2024. NounAtlas: Filling the gap in nominal semantic role labeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16245–16258, Bangkok, Thailand. Association for Computational Linguistics.

Qwen, :, 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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2026. Lora vs full fine-tuning: An illusion of equivalence. *Advances in Neural Information Processing Systems*, 38:174627–174662.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0. LDC2013T19, Linguistic Data Consortium.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. 2024. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In *International Conference on Learning Representations*, volume 2024, pages 25401–25420.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *Preprint*, arXiv:2303.10512.

A Appendix

A.1 Hyperparameters and Compute

Table 5 reports the hyperparameter values used across all experiments. All hyperparameters are held identical between Standard LoRA and PhaseLoRA. Experiments were conducted on NVIDIA RTX A5000, A40 and A100 GPUs, with total compute estimated at approximately 300–400 GPU hours.

Hyperparameter	Value
Sampling Parameters	Greedy Sampling
Number of Epochs	3
Precision	bfloat16
Optimizer	AdamW
Learning Rate	2E-04
Learning Rate Scheduler	cosine with warm-up
Warm-up Ratio	0.05
Batch Size	4
LoRA Dropout	0.05
Gradient Accumulation Steps	4
Weight Decay	0.01
Gradient Clipping	1.0
Maximum Seq. Length	512

Table 5: Hyperparameter values used in all experiments.

A.2 Precision and Recall Breakdowns

Tables 6 and 7 report mean precision and recall scores across three seeds corresponding to the F1 results in Table 1. As discussed in Section 5.3, the F1 gains of PhaseLoRA over Standard LoRA are driven predominantly by improvements in recall, with precision differences remaining comparatively marginal across models, benchmarks and ranks.

A.3 Warm-up Initialisation and Asymmetric Rank Allocation: Additional Models

Table 8 reports warm-up initialisation results for Llama-3.1-8B and Mistral-Nemo-12B with rank 8 adapters. The overall pattern mirrors that observed for Qwen2.5-3B in the main paper.

Table 9 reports asymmetric rank allocation results for the same two models with rank 8 adapters across all three benchmarks, complementing the Qwen2.5-3B results.

A.4 CKA Similarity and Frobenius Norm: Additional Models

Figure 4 shows the CKA similarity between prompt and completion adapters for Llama-3.1-8B on OntoNotes and Mistral-Nemo-12B on NounAtlas. Figure 5 shows the corresponding layer-wise Frobenius norms of the prompt and completion adapters. Table 10 reports the mean Frobenius norm per adapted module averaged across all layers for all three configurations.

ACE2005

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	74.91	**75.34**	0.43 ± 0.68	76.85	**78.08**	1.24 ± 1.21	79.30	**80.00**	0.70 ± 0.37
8	76.36	**76.65**	0.29 ± 0.77	77.45	**77.82**	0.37 ± 0.29	**79.70**	79.34	-0.36 ± 0.16
16	**77.99**	76.97	-1.02 ± 0.03	77.44	**78.22**	0.77 ± 0.81	**79.44**	78.98	-0.46 ± 0.48

OntoNotes

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	85.86	**87.07**	1.21 ± 0.37	87.42	**88.14**	0.72 ± 0.41	87.69	**88.13**	0.44 ± 0.33
8	86.42	**87.64**	1.22 ± 0.53	87.93	**88.65**	0.72 ± 0.13	88.28	**88.44**	0.17 ± 0.04
16	87.70	**88.28**	0.58 ± 0.32	**88.90**	88.75	-0.15 ± 0.35	88.60	**88.96**	0.37 ± 0.12

NounAtlas

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	51.17	**52.41**	1.24 ± 0.17	56.57	**57.31**	0.74 ± 0.07	57.67	**58.02**	0.35 ± 0.41
8	**53.53**	53.35	-0.18 ± 0.53	**57.57**	57.04	-0.52 ± 0.82	**58.11**	58.01	-0.10 ± 1.05
16	54.77	**54.85**	0.08 ± 1.01	**58.56**	58.12	-0.44 ± 0.58	**58.74**	58.70	-0.04 ± 0.01

Table 6: Mean precision scores across three seeds for Standard LoRA and PhaseLoRA at ranks 4, 8 and 16, across three benchmarks and three models. Δ denotes the mean difference (PhaseLoRA $-$ Standard LoRA) $\pm$ standard error across seeds. Bold indicates the higher mean precision between the two methods for each configuration.

ACE2005

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	67.14	**71.77**	4.63 ± 0.71	73.60	**74.97**	1.37 ± 0.58	77.22	**77.58**	0.36 ± 0.53
8	70.70	**73.61**	2.91 ± 0.33	74.73	**75.50**	0.77 ± 0.06	**77.28**	76.75	-0.53 ± 0.67
16	72.72	**74.91**	2.19 ± 0.12	74.50	**75.38**	0.89 ± 0.10	**76.98**	76.21	-0.77 ± 0.12

OntoNotes

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	83.64	**86.71**	3.07 ± 0.23	87.75	**87.92**	0.17 ± 0.45	87.86	**88.49**	0.63 ± 0.44
8	85.09	**88.16**	3.07 ± 0.49	88.03	**89.02**	0.99 ± 0.25	**88.88**	88.47	-0.41 ± 0.14
16	86.68	**88.72**	2.03 ± 0.28	**89.10**	89.07	-0.03 ± 0.28	89.18	**89.62**	0.44 ± 0.27

NounAtlas

	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
LoRA Rank	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)	Standard LoRA	PhaseLoRA	Δ (P-S)
4	49.23	**54.10**	4.86 ± 0.07	56.70	**57.65**	0.96 ± 0.06	58.85	**59.97**	1.12 ± 0.66
8	51.86	**54.73**	2.87 ± 0.08	**58.31**	57.79	-0.52 ± 0.76	**59.86**	59.81	-0.05 ± 0.93
16	53.96	**56.61**	2.65 ± 0.66	**59.07**	**59.07**	0.00 ± 0.37	**61.15**	60.57	-0.58 ± 0.19

Table 7: Mean recall scores across three seeds for Standard LoRA and PhaseLoRA at ranks 4, 8 and 16, across three benchmarks and three models. Δ denotes the mean difference (PhaseLoRA $-$ Standard LoRA) $\pm$ standard error across seeds. Bold indicates the higher mean recall between the two methods for each configuration.

Llama 3.1 8B Instruct			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
No warmup (0+3)	76.64	**88.83**	57.41
1 epoch warmup (1+2)	**76.68**	88.71	**59.45**
2 epoch warmup (2+1)	74.78	88.71	57.28

Mistral Nemo Instruct 2407			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
No warmup (0+3)	78.02	88.46	58.90
1 epoch warmup (1+2)	**78.05**	**89.74**	**60.81**
2 epoch warmup (2+1)	76.91	88.31	59.67

Table 8: F1 scores for warm-up initialisation variants across three benchmarks and two additional models (single seed per model).

Llama 3.1 8B Instruct			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
r8+r8	**76.64**	88.83	57.41
r4+r12	76.08	**88.90**	58.43
r12+r4	76.35	88.71	**58.74**

Mistral Nemo Instruct 2407			
Configuration	**ACE2005**	**OntoNotes**	**NounAtlas**
r8+r8	78.02	88.46	58.90
r4+r12	77.29	**88.52**	58.33
r12+r4	**78.59**	88.27	**59.17**

Table 9: F1 scores for asymmetric rank allocation across three benchmarks and two additional models (single seed per model).

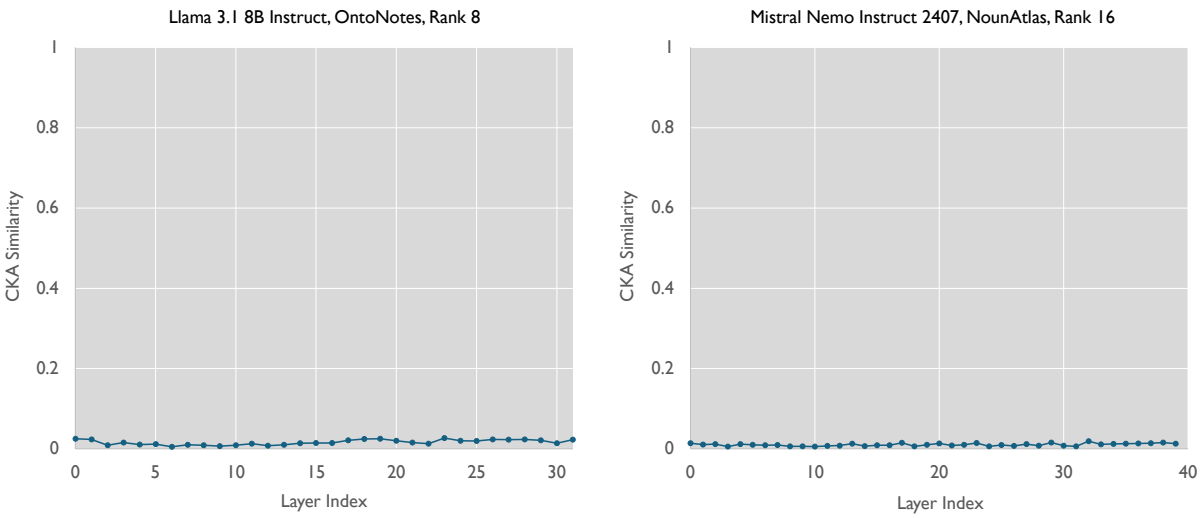

Figure 4: CKA similarity between prompt and completion adapters per layer, averaged across all adapted modules .

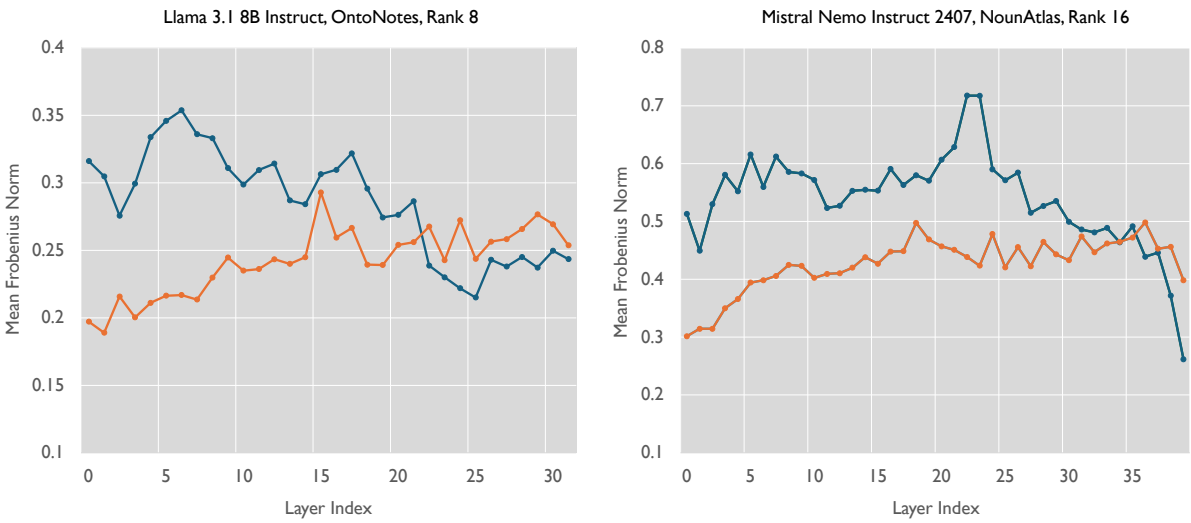

Figure 5: ● Prompt and ● completion adapter Frobenius norms per layer, averaged across all adapted modules.

Qwen2.5 3B Instruct, ACE2005, Rank 4

Module	$\|B_pA_p\|_F$	$\|B_cA_c\|_F$
q_proj	0.17	0.15
k_proj	0.06	0.04
v_proj	0.05	0.05
o_proj	0.17	0.14
gate_proj	**0.44**	**0.36**
up_proj	**0.42**	**0.33**
down_proj	0.22	0.15

Llama 3.1 8B Instruct, OntoNotes, Rank 8

Module	$\|B_pA_p\|_F$	$\|B_cA_c\|_F$
q_proj	0.26	0.25
k_proj	0.16	0.11
v_proj	0.12	0.10
o_proj	0.22	0.21
gate_proj	**0.47**	**0.42**
up_proj	**0.43**	**0.39**
down_proj	0.32	0.22

Mistral Nemo Instruct 2407, NounAtlas, Rank 16

Module	$\|B_pA_p\|_F$	$\|B_cA_c\|_F$
q_proj	0.51	0.43
k_proj	0.26	0.20
v_proj	0.19	0.16
o_proj	0.43	0.38
gate_proj	**1.01**	**0.77**
up_proj	**0.79**	**0.63**
down_proj	0.59	0.41

Table 10: Mean Frobenius norm per module averaged across all layers, for each of the three model–benchmark–rank configurations. gate_proj and up_proj consistently exhibit the largest norms across all configurations (shown in bold).