Latent Merger: Latent-Space Composition of Hypernetwork-Generated LoRAs

Yanlai Yang[1], Minki Kang[2], Divyam Madaan[1], Sung Ju Hwang[2,3], and Mengye Ren[1]
[1]New York University, [2]KAIST, [3]DeepAuto.ai
yy2694@nyu.edu
agenticlearning.ai/latent-merger

Abstract

Recent work proposes hypernetworks that convert a document into a low-rank parameter update (LoRA) in a single forward pass, enabling fast injection of new knowledge into a frozen LLM. However, storing separate LoRA adapters for each document can be unsustainable in streaming settings. In this paper, we study the problem of composing hypernetwork-generated LoRAs using two representative systems, Doc-to-LoRA and SHINE. We show that hypernetwork-generated LoRAs have a geometry sharply different from directly fine-tuned LoRAs. They exhibit greater spectral concentration and stronger cross-document alignment than next-token prediction and QA fine-tuned adapters, which are diffuse and nearly orthogonal across documents. Guided by this observation, we propose latent-space LoRA merging. We compose the intermediate hypernetwork latents and then decode the merged latent through the frozen LoRA-generation head. We show that latent-space merging techniques consistently improve over weight-space and factor-space baselines while preserving a single fixed-rank adapter. These results support latent-space composition as a promising mechanism for combining hypernetwork-generated LoRAs under a fixed output-rank budget.

1 Introduction

Large language models (LLMs) have become increasingly capable at using long contexts, but incorporating new information into an LLM remains costly. A new document can be supplied in the prompt whenever it is needed, but this consumes context length, increases inference cost, and leaves the model's parameters unchanged. Alternatively, the document can be distilled into model parameters through fine-tuning (Gururangan et al., 2020; Hu et al., 2022; Zweiger et al., 2025; Tang et al., 2026) or context distillation (Snell et al., 2022), but this often requires expensive per-document optimization. Hypernetwork-generated LoRAs (Charakorn et al., 2026; Liu et al., 2026) offer a different interface for knowledge injection: after a one-time meta-training stage, a hypernetwork maps a document or context into a compact LoRA adapter in a single forward pass, while the base LLM remains frozen. The resulting adapter can then be stored, reused, and activated without placing the original document back into the prompt.

This interface is especially appealing for settings where knowledge arrives as a stream. For example, a deployed assistant may need to internalize a sequence of news articles, reports, user documents, or database records, and later answer many questions about this accumulated information. Generating one

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

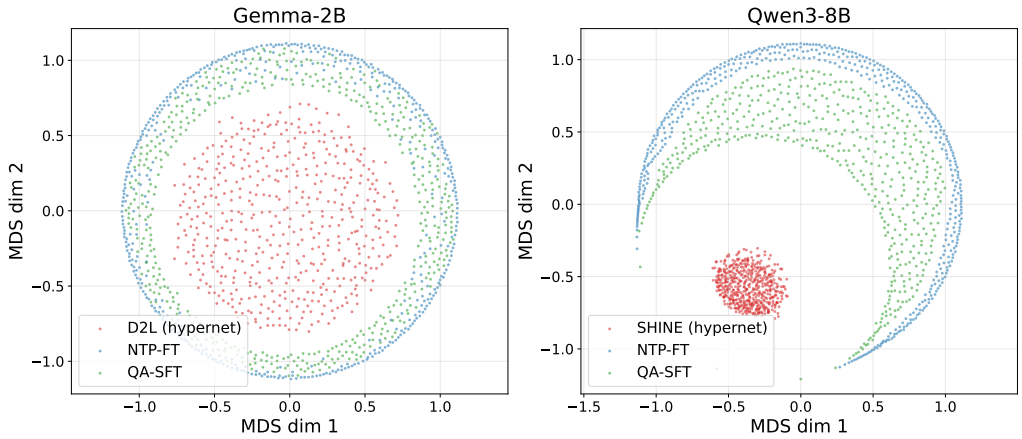

Figure 1: ***Hypernetwork-generated LoRAs occupy a qualitatively different region of adapter space.*** *We visualize pairwise distances between unit-Frobenius-normalized adapters using metric MDS, with 500 passages per method sampled from the SQuAD validation set. In the two-dimensional MDS embedding, directly fine-tuned NTP-FT and QA-SFT adapters form broad shells, while D2L and SHINE adapters concentrate in compact regions. This structural difference motivates latent merging methods that respect the hypernetwork-induced adapter geometry.*

LoRA per document is a natural first step, but it creates a new composition problem. Maintaining an ever-growing bank of document adapters is inefficient, query-time routing to a single adapter is insufficient when questions may involve multiple documents, and activating many adapters by concatenation causes the effective rank, memory footprint, and update magnitude to grow with the number of documents. A practical document-to-LoRA system therefore needs a way to compose multiple generated adapters into a bounded representation.

Most existing model-merging methods operate directly in weight space or LoRA factor space, and are usually motivated by adapters obtained through independent gradient-based fine-tuning. Hypernetwork-generated LoRAs have a different origin. They are not arbitrary endpoints of separate optimization runs; they are outputs of a shared generator, often decoded from an intermediate latent representation.

In this work, we study this question using two recent hypernetwork-based LoRA systems for knowledge injection, Doc-to-LoRA (D2L) (Charakorn et al., 2026) and SHINE (Liu et al., 2026). We compare their document-specific adapters against LoRAs obtained by direct next-token prediction fine-tuning and QA supervised fine-tuning on the same passage set. Figure 1 shows a metric MDS embedding of pairwise adapter distances. In the two-dimensional MDS embedding, directly fine-tuned adapters form broad shells, while hypernetwork-generated adapters occupy more compact regions, especially for SHINE. This qualitative picture is confirmed by spectral and angular statistics in Section 3: hypernetwork-generated adapters exhibit much stronger spectral concentration and substantially larger within-method cosine similarity than directly fine-tuned LoRAs.

Motivated by this observation, we propose Latent Merger, a method to merge hypernetwork-generated LoRAs in the latent space of the hypernetwork rather than in the decoded weight space. Given a set of documents, we extract their frozen hypernetwork latents, compose them with a learned, lightweight merging module, and then pass the merged latent through the frozen LoRA-generation head. Latent Merger is trained with the base LLM and the hypernetwork frozen. It produces a single fixed-rank LoRA adapter for the whole document set, rather than increasing rank or storing one adapter per document.

We evaluate this approach on multi-document QA tuples. Across both D2L and SHINE, Latent Merger consistently outperforms standard weight-space and factor-space merging methods. These results suggest that hypernetwork-generated LoRAs should be treated not merely as low-rank weight updates, but as decoded points on a structured adapter manifold whose latent coordinates provide a natural space for composition.

Our contributions are as follows:

- We empirically characterize the geometry of hypernetwork-generated LoRAs, showing greater spectral concentration and cross-document alignment than next-token prediction and QA fine-tuned adapters. Metric MDS provides a qualitative visualization of these differences.
- We propose Latent Merger, a learned, lightweight latent-space LoRA merging module that composes document latents before decoding them through the frozen hypernetwork head.
- We show empirically that Latent Merger outperforms standard weight-space and factor-space merging baselines for multi-document QA while producing a single fixed-rank adapter.

2 Related Work

Hypernetwork-generated LoRAs. A growing line of work studies parameter-space updates that inject new knowledge into a pre-trained LLM. Earlier approaches include supervised fine-tuning on raw documents, document-specific question-answer pairs, or context distillation (Askell et al., 2021; Mitchell et al., 2021; Snell et al., 2022; Mitchell et al., 2022; Meng et al., 2022, 2023; Qi et al., 2024; Shin et al., 2025). These methods enable the model to answer queries without re-reading the original context, but they typically require expensive per-document optimization or synthetic data generation. Hypernetwork-generated LoRAs offer a complementary route by amortizing the update procedure: after meta-training, a single forward pass maps a document (Charakorn et al., 2026; Liu et al., 2026), task description (Charakorn et al., 2025), or support set (Phang et al., 2023; Ivison et al., 2023) to a compact adapter for a frozen backbone.

Most relevant to our work, Doc-to-LoRA (D2L) instantiates this idea for document internalization by meta-learning an approximate context-distillation procedure (Charakorn et al., 2026). It encodes document-conditioned activations from a frozen LLM through a Perceiver-style bottleneck and decodes the resulting latent into LoRA factors, with a chunking mechanism that produces higher-rank adapters for long documents. SHINE (Liu et al., 2026) pursues the same context-to-LoRA goal with a different architecture: it reuses the frozen LLM to extract memory states and applies a Transformer-based memory-to-parameter module whose structure mirrors the target LoRA parameters, enabling communication across layers and memory tokens before decoding the adapter. Our work studies how the document-specific adapters emitted by such hypernetworks can be merged. We use D2L and SHINE as architecturally distinct testbeds, but exploit their shared structure: both generate LoRA weights from an intermediate latent representation that can be more effectively composed than the LoRA weights themselves.

LoRA and model merging. Model merging seeks to combine independently fine-tuned models or adapters without joint retraining. Popular approaches include averaging checkpoints (Wortsman et al., 2022), combining task vectors through task arithmetic (Ilharco et al., 2022), or using second-order (Matena and Raffel, 2022) or regression-based (Jin et al., 2022) criteria to form weighted averages. Some recent methods improve robustness by sparsifying updates, resolving sign conflicts, or dropping and rescaling delta weights (Yadav et al., 2023; Yu et al., 2024). These ideas have also been extended to the LoRA setting (Huang et al., 2023; Prabhakar et al., 2025). A complementary line of work exploits LoRA's low-rank structure more directly, for example by merging in singular-vector or singular-value space (Gargiulo et al., 2025; Stoica

Table 1: *Structural metrics for hypernetwork-generated and gradient-trained LoRAs. We report the average adapter Frobenius norm, the mean off-diagonal within-method cosine after unit-Frobenius normalization, and the fraction of uncentered spectral energy captured by the leading component.*

Backbone	Method	$\|\Delta W\|_F$	within cos.	top-1 energy
Hypernetwork-based methods				
Gemma2-2B	D2L	30.2	+0.144	15.5%
Qwen3-8B	SHINE	209.4	+0.888	88.8%
Gradient based methods				
Gemma2-2B	NTP-FT	5.6	+0.005	0.6%
Qwen3-8B	NTP-FT	8.5	+0.001	0.2%
Gemma2-2B	QA-SFT	3.5	+0.014	1.5%
Qwen3-8B	QA-SFT	6.1	+0.025	2.6%

et al., 2024), encouraging orthogonal adapter subspaces to reduce interference (Buyukakyuz, 2024; Zhang and Zhou, 2025; Xiong and Xie, 2026; Cao and Wu, 2026), searching for task-specific linear combination coefficients from a small validation set (Huang et al., 2023; Prabhakar et al., 2025). Some more recent methods also explore training a hypernetwork to predict the merge coefficients (Shenaj et al., 2025b) or the merged weights directly (Zhang et al., 2025). Also relevant to our work, KMerge (Shenaj et al., 2025a) considers budgeted sequential setting and maintains a bounded cache of adapters, merging similar LoRAs when capacity is reached; SLAO (Qiao and Mahdavi, 2025) continually folds new adapters into a single LoRA with orthogonal initialization and time-aware scaling.

In contrast to these prior methods that operate in the weight or factor space, our method exploits the fact that each adapter is decoded by a frozen hypernetwork from a structured intermediate representation, and merges in this latent representation space directly. The merged latent code is then decoded through the frozen hypernetwork into the final LoRA. This latent-space formulation supports both training-free operators, such as averaging, and learned operators, such as a lightweight Transformer trained on document mixtures.

Composition in generative models. Related work in image generation studies operations on intermediate representations: StyleGAN mixes styles derived from learned latent codes (Karras et al., 2019). Composable diffusion models combine conditional score estimates to compose visual concepts (Liu et al., 2022). In this work, we present a lightweight learned composition rule for hypernetwork-generated LLM adapters under an output-rank budget.

3 Structure of Hypernetwork-generated LoRA Adapters

Before studying how to merge multiple document adapters, we first examine whether hypernetwork-generated LoRAs have the same geometry as LoRAs obtained by direct fine-tuning. For each passage p and method m, we concatenate the effective LoRA updates ΔW from all target layers l and modules a into a single vector

$$x_p^{(m)} = \mathrm{vec}\left(\{\Delta W_{p,\ell,a}^{(m)}\}_{\ell,a}\right).$$

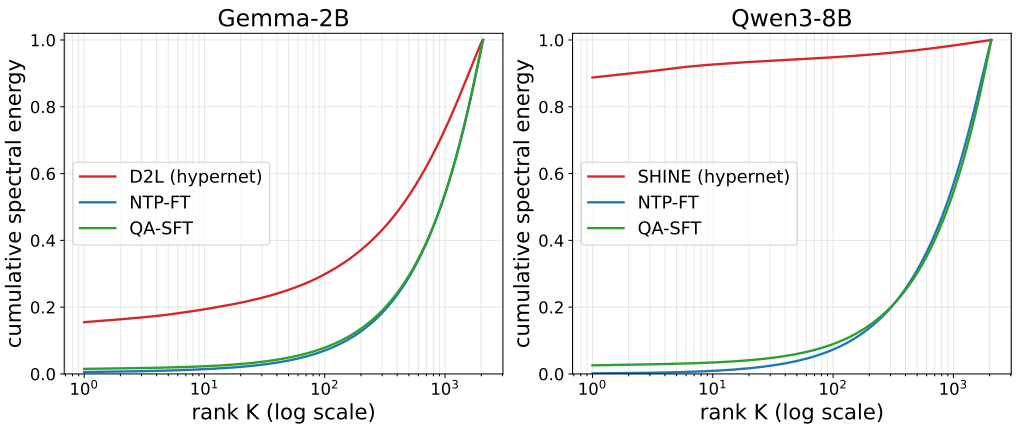

Figure 2: *Cumulative spectral energy of stacked adapter matrices. Hypernetwork-generated adapters concentrate energy in fewer directions than directly fine-tuned NTP-FT and QA-SFT adapters.*

We analyze the resulting adapter bank through its Gram matrix $G_{ij}^{(m)} = \left\langle x_i^{(m)}, x_j^{(m)} \right\rangle$, which we compute directly from the low-rank factors without materializing the dense updates. From this Gram matrix, we measure both the uncentered spectral energy,

$$E_m(K) = \frac{\sum_{k=1}^{K} \lambda_k(G^{(m)})}{\sum_k \lambda_k(G^{(m)})},$$

and the angular geometry of unit-Frobenius-normalized adapters,

$$c_{ij}^{(m)} = \frac{G_{ij}^{(m)}}{\sqrt{G_{ii}^{(m)} G_{jj}^{(m)}}}, \qquad d_{ij}^{(m)} = \left\| u_i^{(m)} - u_j^{(m)} \right\|_2 = \sqrt{2(1 - c_{ij}^{(m)})}.$$

We compare D2L-generated adapters on Gemma2-2B (Team et al., 2024) and SHINE-generated adapters on Qwen3-8B (Yang et al., 2025) against two direct fine-tuning controls on the same passage set: next-token prediction fine-tuning (NTP-FT) and QA supervised fine-tuning (QA-SFT).

Figure 2 shows that hypernetwork-generated adapters are substantially more spectrally concentrated than directly fine-tuned adapters. For both backbones, the spectral energy curve of the hypernetwork method lies far above the corresponding NTP-FT and QA-SFT curves, indicating that a small number of global directions explain a much larger fraction of the uncentered spectral energy. The effect is especially pronounced for SHINE on Qwen3-8B, whose spectrum is dominated by the leading component.

The angular statistics reveal the same qualitative distinction. As shown in Table 1, directly fine-tuned adapters have near-zero within-method cosine similarity, meaning that different passage-specific updates are almost orthogonal after unit-Frobenius normalization. By contrast, D2L adapters are more aligned with one another, and SHINE adapters are highly aligned. This shows that the spectral concentration above is not only a consequence of scale: even after normalizing each adapter to unit Frobenius norm, hypernetwork-generated adapters retain a much stronger shared direction than adapters obtained by direct fine-tuning. The Frobenius norms in Table 1 further show that the generated adapters also differ in magnitude from the fine-tuned controls, motivating our use of normalized cosine and distance metrics when comparing geometry.

Figure 1 provides a qualitative visualization of this angular geometry using metric MDS on pairwise unit-Frobenius distances. In the two-dimensional MDS embedding, the directly fine-tuned NTP-FT and QA-SFT adapters form broad shells, consistent with the near-zero cosine values in Table 1. In contrast, the hypernetwork-generated adapters occupy more compact regions, with the SHINE adapters forming a particularly tight cluster.

This structural difference has direct implications for adapter merging. If per-document LoRAs are nearly orthogonal, weight-space averaging attenuates each document-specific contribution, while concatenation retains the individual factors at a larger representation size. Hypernetwork-generated adapters exhibit stronger cross-document alignment and are decoded from latent representations by a shared generator. This shared parameterization motivates composing document representations before decoding them through the frozen hypernetwork head.

4 Merging Hypernetwork-generated LoRAs

We now study whether multiple document-specific hypernetwork LoRAs can be composed into a single adapter. Each evaluation example consists of a tuple of N passages and their associated questions. A merger receives the adapters, or the corresponding hypernetwork latents, for all N passages and must produce one LoRA dictionary shared across the whole tuple. At inference time, the model answers each question using this single merged adapter, without access to the original document in the context or the document-specific oracle adapter.

4.1 Proposed Method: Learned Latent Merger

Our hypothesis is that hypernetwork-generated LoRAs are easier to compose in the latent space than directly in the LoRA weight space. We therefore train Latent Merger, a lightweight cross-document merging module that composes multiple document-specific hypernetwork latents before the frozen LoRA-generation head.

Given a tuple of N documents, we first extract the frozen hypernetwork latent for each document. We denote the latent at document slot i and latent position p by $z_{i,p} \in \mathbb{R}^D$, where D is the latent feature dimension. The position index p ranges over the non-feature axes of the corresponding hypernetwork latent: for D2L, one position corresponds to a single (b, ℓ, m, r) entry, while for SHINE, one position corresponds to a single (ℓ, t) entry. The merger is applied independently at each such latent position.

Our merger predicts a residual correction to the mean latent $m_p = \frac{1}{N}\sum_{i=1}^{N} z_{i,p}$. For each latent position p, we treat the N document slots as a set. We first project each $z_{i,p}$ into a low-dimensional bottleneck, apply a small Transformer encoder with no positional embeddings along the document axis, and mean-pool the resulting document representations to obtain a residual $\Delta_p \in \mathbb{R}^D$. The final merged latent is

$$z_{\text{merge},p} = m_p + g \cdot \text{Scale}(\Delta_p, m_p),$$

where g is a learned scalar gate. The residual scaling function matches the RMS of the residual to the RMS of the mean latent at each position:

$$\text{RMS}_D(x_p) = \sqrt{\frac{1}{D}\sum_{j=1}^{D} x_{p,j}^2}, \qquad \text{Scale}(\Delta_p, m_p) = \Delta_p \cdot \frac{\text{RMS}_D(m_p)}{\text{RMS}_D(\Delta_p) + \epsilon}.$$

The RMS is computed only along the feature dimension D: we do not center the activations, use global statistics, or pool statistics across layers or latent positions. The output projection is zero-initialized and

Table 2: *Latent merger architecture hyperparameters for D2L and SHINE.*

Hyperparameter	D2L	SHINE
Bottleneck dimension d_{lat}	128	256
Transformer layers	3	3
Attention heads	8	8
FFN hidden dimension	512	1024
Activation	GELU	GELU

the scalar gate is initialized to 0.1, so the merger starts from latent averaging while allowing the learned residual to modify the average during training. The merged latent is then passed through the frozen downstream hypernetwork head to generate one fixed-rank LoRA dictionary shared across all questions in the tuple.

This architecture is permutation-invariant over the input documents and does not increase the LoRA rank. It can therefore be viewed as learning a set function over document-specific hypernetwork latents while preserving the original adapter interface used by the base model.

4.2 Experimental Setup and Baselines

Training data and tuple construction. We train Latent Merger on the SQuAD training split (Rajpurkar et al., 2016). Both D2L and SHINE use the same merger training data. At each training step, we sample the tuple size uniformly as $N \sim \text{Uniform}\{2, 3, 4, 5\}$. We then sample N distinct passages uniformly from the SQuAD training split. For each selected passage, we sample one question uniformly from that passage's associated question list. Thus, each training tuple contains N passages and one question-answer pair per passage.

Models and latent extraction. For the main experiments, we use pretrained generator checkpoints for Gemma-2-2B-it (D2L) and Qwen3-8B (SHINE). For D2L, we merge the post-ResMLP hypernetwork latents and pass the merged latent through the frozen normalization and LoRA-generation head. For SHINE, we apply the same Latent Merger architecture to the Qwen3-8B memory-state latents and pass the merged memory state through the frozen metanetwork. In both cases, the base LM, hypernetwork trunk, and LoRA-generation head remain frozen; only Latent Merger parameters are optimized. Each merger is trained for the latent representation and decoding head of its corresponding generator checkpoint.

Training objective. For SQuAD tuples, Latent Merger is trained using knowledge distillation (Hinton et al., 2015) with a KL-divergence objective from the corresponding per-document oracle adapters.

For each tuple, the merged adapter is shared across all N questions. For question q_i associated with passage i, the teacher is the oracle model using the document-specific LoRA generated from passage i. We do not mix teacher distributions for the same question. We compute teacher-forced oracle logits on the gold answer sequence. The student logits are produced by the same base model using the single merged adapter, also under teacher forcing on the gold answer sequence. The KL loss is computed only over answer-token positions:

$$\mathcal{L}_{\text{KL}} = \frac{1}{|\mathcal{M}|} \sum_{(i,t)\in\mathcal{M}} \text{KL}\left(p_{i,t}^{\text{oracle}} \,\|\, p_{i,t}^{\text{merge}}\right),$$

where $\mathcal{M}$ is the set of answer-token positions across the tuple. The KL temperature is $T = 1.0$. For 2Wiki examples, the teacher is instead the frozen base model with both source documents in context.

Optimization and compute. For D2L, we train Latent Merger for 4000 steps. For SHINE, we train for 2000 steps. Both use AdamW (Loshchilov and Hutter, 2017) with default betas, learning rate 5×10^{-5}, weight decay 0.01, linear warmup for 100 steps, cosine learning-rate decay, gradient clipping at norm 1.0, and batch size 2. Each batch contains two independently sampled tuples, and each tuple independently samples $N \in \{2, 3, 4, 5\}$. The main training and evaluation experiments can be run on a single NVIDIA A100 GPU.

Evaluation protocol. We evaluate on tuples sampled from the held-out SQuAD validation set (Rajpurkar et al., 2016) and the out-of-distribution MS MARCO validation set (Bajaj et al., 2016) with $N \in \{2, 5, 10\}$. For each tuple, a method must produce a single merged LoRA dictionary that is used to answer every question in the tuple. The exception is the oracle, which does not merge adapters and instead answers each question using the LoRA generated from its own evidence document. We report token-overlap F1, multiplied by 100, with mean ± 95% confidence intervals. Confidence intervals are based on 100 seeds for $N = 2$, 50 seeds for $N = 5$, and 20 seeds for $N = 10$ on SQuAD, and 50 seeds for all N values on MS MARCO.

Baselines and training-free LoRA merging methods. We compare against a suite of training-free methods that combine the document-specific adapters in a tuple without optimizing additional parameters. Let the LoRA update produced for document i at a given target module be $\Delta W_i = A_i B_i^\top$, where each adapter has rank r. All non-oracle methods produce a single LoRA dictionary shared across all questions in the tuple.

- **Oracle:** We do not merge adapters. Instead, each question is answered using the LoRA generated from its own evidence document. This provides an empirical reference for methods that must compress all tuple information into one adapter.
- **Unscaled rank concatenation:** We concatenate the low-rank factors along the rank dimension, $A_{\text{cat}} = [A_1, \dots, A_N]$, $B_{\text{cat}} = [B_1, \dots, B_N]$, yielding a rank-$Nr$ adapter with update $\Delta W_{\text{cat}} = \sum_{i=1}^{N} \Delta W_i$. This corresponds to the natural rank-expansion behavior of the original D2L construction and is an optimistic baseline, since it preserves all rank components rather than compressing the tuple into a fixed-rank adapter.
- **Mean-scaled rank concatenation:** We use the same rank-Nr concatenation but divide one factor by N, equivalently producing $\Delta W_{\text{merge}} = \frac{1}{N} \sum_{i=1}^{N} \Delta W_i$. This isolates the effect of rank expansion from the magnitude growth induced by unscaled concatenation.
- **Norm-preserving rank concatenation:** We again concatenate all rank components, but rescale the resulting update to match the average Frobenius norm of the individual document adapters:

$$\Delta W_{\text{merge}} = \gamma \sum_{i=1}^{N} \Delta W_i, \qquad \gamma = \frac{\frac{1}{N} \sum_{i=1}^{N} \|\Delta W_i\|_F}{\left\| \sum_{i=1}^{N} \Delta W_i \right\|_F + \epsilon}.$$

  This baseline preserves the full rank-expanded subspace while controlling for the scale mismatch introduced by summing multiple LoRA updates.
- **Factor averaging:** We average the LoRA factors directly, $A_{\text{avg}} = \frac{1}{N} \sum_i A_i$, $B_{\text{avg}} = \frac{1}{N} \sum_i B_i$. This keeps the adapter rank fixed at r, but averages the factorization rather than the induced weight update, and is therefore not equivalent to averaging ΔW_i in weight space.

- **SVD-compressed weight averaging:** We first compute the mean dense update $\Delta\bar{W} = \frac{1}{N}\sum_i \Delta W_i$, then take its best rank-r truncated SVD approximation and refactorize it into LoRA form. This provides a strong fixed-rank weight-space baseline for compressing the average update back to the original LoRA rank.
- **TIES:** We adapt TIES (Yadav et al., 2023) to the LoRA setting by applying magnitude trimming, majority-sign election, and sign-consistent merging to the aligned adapter tensors, followed by mean-scaled rank concatenation to materialize the merged LoRA dictionary. We use $d = 0.5$ in the main experiments.
- **DARE:** We adapt DARE (Yu et al., 2024) by randomly dropping LoRA delta entries, rescaling the retained entries by the inverse keep probability, and then applying mean-scaled rank concatenation. This evaluates whether sparsifying redundant adapter deltas before merging improves robustness to cross-document interference. We experiment with $p = 0.2, 0.5, 0.8$ and pick the best performance.
- **Latent averaging:** As a simple permutation-invariant baseline analogous to our proposed Latent Merger, we average the hypernetwork latents before LoRA generation. For D2L, we average the post-ResMLP latents and pass the result through the frozen normalization and LoRA-generation head; for SHINE, we average the memory states and pass the result through the frozen metanetwork. This is the direct training-free counterpart of our proposed Latent Merger and isolates the benefit of learning a nontrivial set function over document latents.

4.3 Results

We report the composition results for D2L and SHINE in Tables 3 and 4, respectively, comparing Latent Merger against rank-concatenation, weight-space, factor-space, and latent-space baselines across different composition sizes.

Overall performance. Latent Merger achieves the strongest overall composition performance across both hypernetwork settings. For D2L, it obtains the best or tied-best mean performance for every evaluated value of N on both SQuAD and MS MARCO. For SHINE, it similarly leads on all SQuAD settings and on MS MARCO for $N = 2$ and $N = 5$, while remaining essentially competitive with the best method in the most challenging $N = 10$ OOD setting. These results indicate that the learned latent-space composition rule provides a robust advantage over direct weight-, factor-, and rank-space composition strategies.

Generalization beyond the training regime. In the main experiments, Latent Merger is trained only on SQuAD tuples with N sampled from $\{2, 3, 4, 5\}$, yet the evaluation probes two forms of generalization: extrapolation to larger compositions and transfer to an out-of-domain dataset. The method remains the strongest approach when extrapolated to $N = 10$ on SQuAD for both D2L and SHINE, suggesting that the learned merger does not merely memorize the training cardinalities. It also transfers well to MS MARCO, consistently matching or exceeding competing approaches except in the most OOD SHINE setting, where its performance is nearly indistinguishable from the best baseline. This suggests that latent composition captures structure that is not specific to the SQuAD training distribution.

Limitations of rank concatenation. Rank concatenation provides a larger effective LoRA rank than the other composition methods, and might therefore be expected to preserve more task-specific information. Surprisingly, this additional capacity does not translate into reliable performance. This is especially notable for D2L, where the training procedure already encourages concatenation-like composition by chunking documents and composing the resulting LoRAs. Across both D2L and SHINE, however, none of the unscaled, $1/N$-scaled, or norm-preserving concatenation variants is consistently competitive across all values of N. The results suggest that simply increasing rank does not solve the composition problem: unscaled concatenation becomes unstable as more LoRAs are added, whereas aggressive scaling weakens the

Table 3: *D2L composition results on held-out SQuAD-val and OOD MS MARCO tuples across different values of N. We report token-overlap F1, multiplied by 100, and mean ± 95% CI. Concatenation-based methods increase the effective LoRA rank from $r = 8$ to Nr, while other methods produce a single rank-r LoRA.*

	SQuAD			MS MARCO		
Method	$N = 2$	$N = 5$	$N = 10$	$N = 2$	$N = 5$	$N = 10$
Oracle D2L		66.8			42.2	
Rank Concatenation Methods						
1/N Scaled Concat	24.1 ± 2.5	19.3 ± 2.1	18.6 ± 1.4	20.3 ± 3.0	20.5 ± 1.9	20.7 ± 1.7
Unscaled Concat	30.9 ± 2.7	22.9 ± 2.7	0.6 ± 0.2	24.3 ± 2.8	6.7 ± 1.5	2.2 ± 0.6
Norm-preserving	26.6 ± 2.5	21.8 ± 2.2	20.0 ± 1.5	21.1 ± 2.8	22.1 ± 1.9	**22.2 ± 1.8**
Weight and factor-space Methods						
Factor Average	41.2 ± 2.7	19.8 ± 2.0	14.8 ± 1.6	29.5 ± 3.5	22.4 ± 2.1	19.7 ± 1.5
SVD Compress	48.5 ± 2.8	22.0 ± 2.1	15.5 ± 1.7	31.0 ± 3.5	23.2 ± 2.1	20.1 ± 1.5
TIES ($d = 0.5$)	43.7 ± 2.7	21.9 ± 2.4	16.4 ± 1.5	29.0 ± 3.6	21.6 ± 1.8	15.5 ± 1.5
DARE	18.7 ± 2.3	17.0 ± 1.9	17.1 ± 1.4	19.2 ± 2.9	19.2 ± 1.9	19.5 ± 1.7
Latent-space Methods						
Latent Average	47.6 ± 2.8	25.2 ± 2.5	18.6 ± 1.7	31.3 ± 3.5	**23.5 ± 2.2**	20.7 ± 1.5
Latent Merger (Ours)	**50.7 ± 2.8**	**29.8 ± 2.5**	**21.0 ± 1.3**	**32.0 ± 3.7**	**23.5 ± 2.0**	**22.2 ± 1.7**

contribution of each individual LoRA.

Effectiveness of latent-space composition. The comparison between latent-space methods and weight- or factor-space baselines supports the central hypothesis that hypernetwork-generated LoRAs are easier to compose before they are decoded into LoRA factors. Even the training-free Latent Average baseline is highly competitive and often outperforms direct factor averaging, SVD compression, TIES, and DARE. Latent Merger further improves upon this by learning a composition function in the hypernetwork latent space, yielding the most consistent gains across datasets, composition sizes, and hypernetwork backbones. This pattern indicates that the latent representation provides a smoother and more semantically aligned space for merging than the induced LoRA weight space itself.

Multi-hop composition. Table 6 reports D2L results on 2WikiMultihopQA (Ho et al., 2020) compositional questions. Training the merger on an XSum+2Wiki+MMLU mixture with an in-context teacher that reads both source documents for 2Wiki examples increases judge accuracy from 36.0% to 42.0%, an absolute gain of 6.0 percentage points. The SQuAD-trained merger does not improve over latent averaging in this evaluation.

KL vs. Cross-Entropy training objective for Latent Merger: In addition to the knowledge distillation KL-divergence objective that we used to train Latent Merger, using the cross-entropy loss on the ground truth answers as in standard supervised fine-tuning (SFT) is also a natural choice. We compare KL diver-

Table 4: *SHINE composition results on held-out SQuAD-val and OOD MS MARCO tuples across different values of N. We report token-overlap F1, multiplied by 100, and mean ± 95% CI.*

	SQuAD			MS MARCO		
Method	$N = 2$	$N = 5$	$N = 10$	$N = 2$	$N = 5$	$N = 10$
Oracle SHINE		56.8			38.1	
Rank Concatenation Methods						
1/N Scaled Concat	44.3 ± 3.1	29.2 ± 3.0	24.7 ± 2.9	26.6 ± 3.9	22.1 ± 2.1	20.0 ± 1.6
Unscaled Concat	12.2 ± 2.0	0.0 ± 0.0	0.0 ± 0.0	6.4 ± 1.2	0.1 ± 0.1	0.1 ± 0.1
Norm-preserving	44.9 ± 3.0	29.1 ± 2.7	21.7 ± 2.3	28.9 ± 3.9	21.7 ± 2.1	19.4 ± 1.5
Weight and factor-space Methods						
Factor Average	44.7 ± 3.1	29.5 ± 3.0	23.7 ± 2.6	26.7 ± 3.9	22.1 ± 2.0	20.1 ± 1.6
SVD Compress	44.6 ± 3.1	29.3 ± 3.0	23.6 ± 2.7	26.5 ± 3.7	22.2 ± 2.0	20.1 ± 1.6
TIES ($d = 0.5$)	43.3 ± 4.6	26.7 ± 2.5	19.4 ± 2.1	29.0 ± 7.5	21.8 ± 3.2	19.0 ± 1.5
DARE	31.3 ± 4.1	27.5 ± 3.1	23.8 ± 2.8	31.8 ± 6.7	27.0 ± 3.6	**26.0 ± 1.4**
Latent-space Methods						
Latent Average	45.4 ± 3.0	30.6 ± 2.8	23.5 ± 2.5	29.5 ± 3.6	22.9 ± 2.1	20.9 ± 1.5
Latent Merger (Ours)	**50.2 ± 4.2**	**39.8 ± 2.8**	**30.2 ± 2.5**	**33.5 ± 4.0**	**27.2 ± 2.2**	25.8 ± 1.8

gence and cross-entropy as the training objective for Latent Merger training in Table 5. We observe that the KL loss consistently outperforms the CE loss across all values of N on both D2L and SHINE. The results suggest that matching the full output distribution provides a more reliable training signal than optimizing only the ground-truth answer likelihood. Overall, these results indicate that KL loss is the preferable objective for training Latent Merger.

Additional evaluations. Appendix A reports evaluations on additional D2L backbones. Appendix B extends fixed-rank evaluation to $N = 20, 50$. Appendix D reports one-time composition latency and transient GPU memory. At $N = 20$, the complete latent path takes 0.77 ms for D2L and 44.5 ms for SHINE on an NVIDIA H200. The latency and peak-memory trade-offs depend on the architecture and baseline.

5 Discussion and Conclusion

In this work, we study the geometry and composition of hypernetwork-generated LoRA adapters. Across two representative systems, D2L and SHINE, we find that generated document adapters differ sharply from LoRAs obtained by direct fine-tuning. While NTP-FT and QA-SFT adapters are diffuse and nearly orthogonal across documents, hypernetwork-generated adapters are much more spectrally concentrated and occupy compact regions of adapter space. These results suggest that hypernetwork-generated LoRAs should not be treated simply as ordinary low-rank weight updates, but as decoded points on a structured adapter manifold induced by the shared generator.

Table 5: *Comparison of KL and CE training objectives for Latent Merger on SQuAD validation set. We report token-overlap F1, multiplied by 100, and mean ± 95% CI.*

	D2L			SHINE		
Loss	$N = 2$	$N = 5$	$N = 10$	$N = 2$	$N = 5$	$N = 10$
KL Divergence	**51.5 ± 4.1**	**29.8 ± 2.5**	**21.0 ± 1.3**	**50.2 ± 4.2**	**40.6 ± 2.8**	**32.3 ± 2.6**
Cross-Entropy	48.3 ± 4.1	24.7 ± 2.4	17.9 ± 1.2	49.8 ± 4.6	39.8 ± 2.8	30.2 ± 2.5

Table 6: *D2L composition on 2WikiMultihopQA compositional questions. Judge accuracy is reported as a percentage. The mixed-task-trained merger uses the frozen base language model with both source documents in context as its teacher for 2Wiki examples; the SQuAD-trained merger uses per-document teachers.*

Method	Merger training data	Judge accuracy (%)
Latent averaging	None	36.0
Latent Merger	SQuAD	36.0
Latent Merger	XSum+2Wiki+MMLU	42.0

This structural difference directly motivates our merging method. Instead of composing adapters after they have been decoded into LoRA weights or factors, we merge their intermediate hypernetwork latents and then decode the merged representation through the frozen LoRA-generation head. Both latent averaging and Latent Merger outperform standard weight-space and factor-space baselines across D2L and SHINE, while producing a single fixed-rank adapter. The learned merger further improves over latent averaging, showing that a lightweight permutation-invariant set function can capture useful cross-document corrections without modifying the base LLM or the original hypernetwork. These findings suggest that latent-space composition can be a promising mechanism for bounded accumulation of hypernetwork-generated document adapters.

Limitations and future work. Our reported composition evaluations cover QA, while non-QA generalization and resolution of conflicting sources remain untested. Our experiments primarily evaluate composing multiple LoRAs into a single fixed-rank LoRA adapter, which can become a capacity bottleneck as more documents are merged. A natural next step is to combine latent merging with adaptive rank allocation, retrieval over merged adapter banks, and online memory-management policies that decide when to merge, retain, or discard document adapters in a streaming setting.

Acknowledgments

We thank Kyunghyun Cho for helpful discussion. This research was supported by Visko Platform, Pionex, a Google TPU Award, the NYU-KAIST Award A25-0081-002, and the Institute of Information & Communications Technology Planning & Evaluation (IITP) with a grant funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research Agreement (No. RS-2024-00469482 & RS-2024-00509279). This work was also supported by the Institute for Information & communications Technology Planning & Evaluation (IITP) grant

funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)). The compute is supported by the NYU High Performance Computing resources, services, and staff expertise.

Funding Transparency Statement

Funding in direct support of this work: Visko Platform, Pionex, a Google TPU Award, the NYU-KAIST Award A25-0081-002, and grants from IITP and MSIT of the Republic of Korea through the Global AI Frontier Lab (RS-2024-00469482, RS-2024-00509279) and the Artificial Intelligence Graduate School Program (KAIST) (RS-2019-II190075), and NYU HPC.

Additional revenues: Student support from the Meta AI Mentorship Program, Azure credits from the Microsoft Accelerating Foundation Models Research program, Google Cloud credits, Anthropic API credits, and compute from the Center for AI Safety.

References

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. (2021). A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Buyukakyuz, K. (2024). Olora: Orthonormal low-rank adaptation of large language models.

Cao, K. and Wu, S. (2026). Orthogonal low-rank adaptation in lie groups for continual learning of large language models.

Charakorn, R., Cetin, E., Tang, Y., and Lange, R. T. (2025). Text-to-lora: Instant transformer adaption. *arXiv preprint arXiv:2506.06105*.

Charakorn, R., Cetin, E., Uesaka, S., and Lange, R. T. (2026). Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*.

Gargiulo, A. A., Crisostomi, D., Bucarelli, M. S., Scardapane, S., Silvestri, F., and Rodola, E. (2025). Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18695–18705.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8342–8360.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Ho, X., Duong Nguyen, A.-K., Sugawara, S., and Aizawa, A. (2020). Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference*

on Computational Linguistics, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

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

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2022). Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Ivison, H., Bhagia, A., Wang, Y., Hajishirzi, H., and Peters, M. E. (2023). Hint: hypernetwork instruction tuning for efficient zero-and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11272–11288.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. (2022). Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, N., Li, S., Du, Y., Torralba, A., and Tenenbaum, J. B. (2022). Compositional visual generation with composable diffusion models. In *Computer Vision – ECCV 2022*, pages 423–439, Cham. Springer Nature Switzerland.

Liu, Y., Wang, X., Mao, Y., Gelbery, Y., Maron, H., and Zhang, M. (2026). Shine: A scalable in-context hypernetwork for mapping context to lora in a single pass. *arXiv preprint arXiv:2602.06358*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Matena, M. S. and Raffel, C. A. (2022). Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. (2022). Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Meng, K., Sharma, A. S., Andonian, A., Belinkov, Y., and Bau, D. (2023). Mass-editing memory in a transformer.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. (2021). Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Mitchell, E., Lin, C., Bosselut, A., Manning, C. D., and Finn, C. (2022). Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Phang, J., Mao, Y., He, P., and Chen, W. (2023). Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*, pages 27854–27875. pmlr.

Prabhakar, A., Li, Y., Narasimhan, K., Kakade, S., Malach, E., and Jelassi, S. (2025). Lora soups: Merging loras for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 644–655.

Qi, S., Yang, B., Jiang, K., Wang, X., Li, J., Zhong, Y., Yang, Y., and Zheng, Z. (2024). In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

Qiao, F. and Mahdavi, M. (2025). Merge before forget: A single lora continual learning via continual merging. *arXiv preprint arXiv:2512.23017*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392.

Shenaj, D., Bohdal, O., Ceritli, T., Ozay, M., Zanuttigh, P., and Michieli, U. (2025a). K-merge: Online continual merging of adapters for on-device large language models. *arXiv preprint arXiv:2510.13537*.

Shenaj, D., Bohdal, O., Ozay, M., Zanuttigh, P., and Michieli, U. (2025b). Lora. rar: Learning to merge loras via hypernetworks for subject-style conditioned image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16132–16142.

Shin, H., Ji, L., Gong, Y., Kim, S., Choi, E., and Seo, M. (2025). Generative prompt internalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7338–7363.

Snell, C., Klein, D., and Zhong, R. (2022). Learning by distilling context. *arXiv preprint arXiv:2209.15189*.

Stoica, G., Ramesh, P., Ecsedi, B., Choshen, L., and Hoffman, J. (2024). Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*.

Tang, P., Wang, Y., and Zhang, M. (2026). Knowledge is not enough: Injecting rl skills for continual adaptation.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. (2024). Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. (2022). Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Xiong, Y. and Xie, X. (2026). Oplora: Orthogonal projection lora prevents catastrophic forgetting during parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34088–34096.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. (2023). Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. (2025). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024). Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Zhang, H., Li, Z., Bao, R., Gao, Y., Xiao, X., Zhang, H., Zhang, S., Huang, B., Wu, Y., Wang, T., et al. (2025). Hyperadalora: Accelerating lora rank allocation during training via hypernetworks without sacrificing performance. *arXiv preprint arXiv:2510.02630*.

Zhang, H. and Zhou, J. (2025). Unraveling lora interference: Orthogonal subspaces for robust model merging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26459–26472.

Zweiger, A., Pari, J., Guo, H., Akyürek, E., Kim, Y., and Agrawal, P. (2025). Self-adapting language models. *arXiv preprint arXiv:2506.10943*.

A Additional backbone evaluation

We additionally train Latent Merger using released D2L checkpoints for Mistral-7B (Jiang et al., 2023) and Qwen3-4B (Yang et al., 2025), keeping the underlying language models and hypernetworks frozen. Table 7 reports the SQuAD-2067 composition evaluation. Latent Merger achieves higher mean scores than latent averaging at all three composition sizes on both backbones. The mergers use 2,000 training steps. Evaluation uses the 2,067-passage, 10,570-question SQuAD dataset, one random tuple per seed per N, and 64-token greedy decoding.

Table 7: *Additional D2L backbones on SQuAD-2067. We report token-overlap F1, multiplied by 100, and mean ± 95% CI.*

Method	$N = 2$	$N = 5$	$N = 10$
D2L Mistral-7B			
Per-document oracle	40.5 ± 5.8	38.7 ± 3.7	38.3 ± 2.0
Latent Merger	29.1 ± 4.4	20.6 ± 3.5	15.5 ± 1.8
Latent averaging	22.3 ± 4.9	13.8 ± 2.6	10.8 ± 1.6
SVD-compressed weight averaging	22.1 ± 5.2	13.0 ± 2.8	10.8 ± 1.5
Factor averaging	20.3 ± 4.4	11.8 ± 2.8	10.3 ± 1.5
TIES ($d = 0.5$)	16.2 ± 3.7	9.5 ± 2.5	8.3 ± 1.2
D2L Qwen3-4B			
Per-document oracle	25.4 ± 3.8	26.4 ± 2.4	26.5 ± 1.3
Latent Merger	23.0 ± 3.5	16.0 ± 3.0	12.6 ± 1.4
Latent averaging	12.9 ± 2.7	8.7 ± 1.2	7.0 ± 0.7
SVD-compressed weight averaging	15.3 ± 3.4	7.5 ± 1.4	6.4 ± 0.6
Factor averaging	10.7 ± 2.2	6.6 ± 1.2	6.1 ± 0.6
TIES ($d = 0.5$)	11.1 ± 3.0	6.6 ± 1.3	7.2 ± 0.9

B Fixed-rank composition at larger cardinalities

Table 8 reports the extended SQuAD-2067 evaluation at $N = 20$ and $N = 50$, alongside closed-book and per-document-oracle references. The flat merger remains above the closed-book score, but its score decreases with cardinality and remains substantially below the per-document oracle.

C Multi-hop QA extension

The 2WikiMultihopQA extension uses compositional questions with evidence from two source documents. Instead of using a per-document oracle teacher, we train the merger against the frozen base language model with both documents in context for 2Wiki examples. The student answers using the merged adapter without access to those source documents or their titles.Table 6 compares this mixed-task-trained merger with latent averaging and the SQuAD-trained merger. Evaluation uses 300 dev-split questions filtered to

Table 8: *Extended fixed-rank composition on SQuAD-2067, in token-overlap F1 multiplied by 100.*

Setting	D2L Gemma-2B	SHINE Qwen3-8B
Closed-book	10.6	5.0
Per-document oracle	68.0	54.8
Flat Latent Merger, $N = 20$	15.1	24.2
Flat Latent Merger, $N = 50$	13.0	21.5

exactly two gold supporting paragraphs, with 32-token greedy decoding.

D Latency Measurements

We profile the latency of different LoRA merging methods on an NVIDIA H200 GPU. Results are presented in Tables 9 and 10. At $N = 20$, the complete latent path is faster than the measured TIES, SVD-compression, and DARE implementations for both architectures.

Table 9: *D2L Gemma-2B composition latency in milliseconds on an NVIDIA H200.*

Operation	$N = 2$	$N = 5$	$N = 10$	$N = 20$
Composition block	0.54	0.54	0.57	0.50
Generator head	0.17	0.17	0.16	0.17
Complete latent path	0.83	0.83	0.86	0.77
Elementwise mean	0.03	0.03	0.04	0.05
Rank concatenation	0.04	0.06	0.09	0.16
TIES	12.5	13.2	14.2	16.3
SVD compression	17.4	50.6	96.8	209.1
DARE	17.6	49.1	88.3	180.6

Table 10: *SHINE Qwen3-8B composition latency in milliseconds on an NVIDIA H200.*

Operation	$N = 2$	$N = 5$	$N = 10$	$N = 20$
Composition block	4.5	5.6	7.2	10.4
Generator head	34.0	33.8	33.9	33.8
Complete latent path	39.0	39.9	41.1	44.5
Elementwise mean	7.3	13.4	23.4	43.1
Rank concatenation	4.3	4.8	5.5	7.1
TIES	175.7	437.4	873.1	1745.3
SVD compression (rank r)	163.3	463.8	840.3	1726.4
DARE ($p = 0.5$, rescaled)	28.9	66.1	128.0	249.3