HYPERBRIDGE: EFFECTIVELY ADAPTING HYPERNETWORKS ACROSS LLM ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Hypernetworks have recently been explored to internalize long contexts into Large Language Model (LLM) parameters, where a hypernetwork maps documents to parameter updates that enable the adapted LLM to answer relevant queries without repeatedly processing the original context. However, obtaining such a hypernetwork requires substantial data and training compute. Beyond that, the resulting hypernetwork is tightly coupled to its corresponding LLM: its internal modules must match the backbone's hidden dimensions and number of layers. This limits its practical usage under constrained resources since repeating the same training process for each newly introduced LLM is prohibitively expensive. Thus, we ask: *Under limited data and compute, can we transfer a trained source hypernetwork across backbones and enable more effective target hypernetwork training?* To answer this, we conduct a systematic cross-architecture analysis using representational similarity analysis. We reveal a clear separation in transferability across the hypernetwork: its intermediate representations exhibit substantial shared structure across backbones and can be aligned via a linear mapping. In contrast, its parameter generator remains strongly architecture-specific. Guided by these findings, we propose HYPERBRIDGE, a cross-architecture hypernetwork transfer framework that aligns input representations, preserves transferable weights, and re-learns backbone-specific heads. We evaluate HYPERBRIDGE across multiple LLM architectures and QA datasets, showing up to 28.5% higher performance than training from scratch and remaining effective across data budgets.

1 INTRODUCTION

Hypernetworks (Ha et al., 2017) generate another model's parameters conditioned on an input, providing a mechanism to adapt neural networks without directly updating the backbone. In Large Language Models (LLMs), this paradigm has recently been explored for *context internalization* (Shin et al., 2025; Qi et al., 2025; Kujanpää et al., 2025; Caccia et al., 2025; Charakorn et al., 2025; 2026): instead of repeatedly placing a long document in the prompt, a hypernetwork can encode the document into lightweight parameter updates, such as LoRA adapters (Hu et al., 2022), that can be plugged and reused across subsequent queries (Charakorn et al., 2025; 2026)[1].

However, this benefit comes with limitations: (i) **Training Expense**: training requires millions of document-query pairs and substantial GPU training; and (ii) **Backbone Specificity**: the input interface depends on backbone-specific hidden representations, while the output modules must match the target model's layer count and dimensions. As a result, when the underlying LLM architecture changes, the learned hypernetwork cannot be directly reused, and a new target hypernetwork requires another costly training process.

These limitations motivate our central research question:

> *Under limited training data and compute, how can the knowledge learned by a context-internalization hypernetwork be reused and transferred across LLM backbones to train the target hypernetwork more effectively?*

[1]The inference procedure of hypernetworks is briefly summarized in Figure 1(i).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

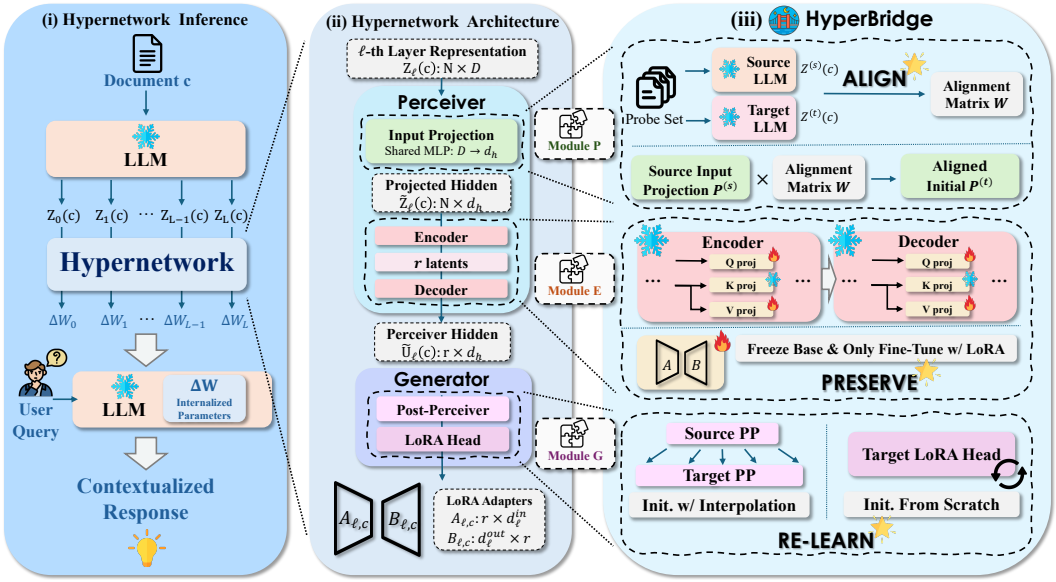


Figure 1: Hypernetwork Inference & Architecture and HYPERBRIDGE. (i) **Inference**: Hypernetwork generates LoRA parameters to condition the LLM for query answering. (ii) **Architecture**: Perceiver (including input projection P and encoder/decoder E) and LoRA Generator G . (iii) **Our Framework**: HYPERBRIDGE aligns input projections (ALIGN), preserves Perceiver with lightweight low-rank trainable parameters (PRESERVE), and re-initializes the LoRA generator (RE-LEARN).

To answer this question, we first conduct a representational similarity analysis (Kriegeskorte et al., 2008; Kornblith et al., 2019), which reveals a clear separation in transferability across the hypernetwork architecture (detailed in Figure 1(ii)). Intermediate representations align strongly across backbones, indicating that internal knowledge can be shared across architectures. In contrast, the final generation module remains strongly tied to the target architecture and must be re-learned.

Motivated by these observations, we propose HYPERBRIDGE, an effective cross-architecture hypernetwork transfer framework. As shown in Figure 1(iii), HYPERBRIDGE effectively adapts the transferable knowledge from a trained source hypernetwork: it aligns the backbone-specific input projection (ALIGN), preserves the Perceiver encoder/decoder (PRESERVE), and re-initializes the LoRA generator (RE-LEARN).

We evaluate HYPERBRIDGE across various LLM backbones and multiple question-answering datasets. Across settings, HYPERBRIDGE outperforms training a target hypernetwork from scratch across varying data budgets. We also show that HYPERBRIDGE is robust under different training data compositions. These results show that HYPERBRIDGE can effectively adapt across LLM architectures with limited resources, making hypernetworks more practical and accessible. In this paper, our main contributions are as follows:

- **Problem.** We introduce a novel problem: *cross-architecture hypernetwork transfer*, which studies whether a hypernetwork trained for one LLM backbone can be adapted to another.
- **Analysis.** We provide a systematic cross-architecture analysis of context-internalization hypernetworks, identifying which learned representations and components contain reusable knowledge and which remain backbone-specific.
- **Method.** We introduce HYPERBRIDGE, a cross-architecture transfer framework that leverages knowledge from a trained source hypernetwork to improve target hypernetwork training with limited data and compute.
- **Experiments.** We evaluate HYPERBRIDGE across various LLM architectures, multiple QA datasets, and varying training budgets, demonstrating superior performance over hypernetworks that are trained from scratch.

2 PRELIMINARIES

2.1 HYPERNETWORKS FOR CONTEXT INTERNALIZATION

Context Internalization (CI). Let f_θ denote an LLM, c a context document, and x a query. Standard prompting answers the query by conditioning on both the query and context, $p_\theta(y \mid x, c)$. *Context internalization (CI)* (Shin et al., 2025; Qi et al., 2025; Kujanpää et al., 2025; Caccia et al., 2025) instead seeks a context-specific parameter update $\Delta\theta_c$ such that $p_{\theta+\Delta\theta_c}(y \mid x) \approx p_\theta(y \mid x, c)$, allowing the model to answer queries without including c in the prompt.

Hypernetworks for CI. Hypernetworks amortize this process by learning a direct mapping from a context to model parameters. In this paper, we study the hypernetwork architecture built by Charakorn et al. (2026), which generates context-specific LoRA adapters for a frozen target LLM. Given a document c, the target LLM first encodes it into layer-wise hidden representations, $Z(c) = \{Z_\ell(c)\}_{\ell=1}^{L}$, where L is the number of layers. The hypernetwork H_ϕ then maps these representations to a collection of LoRA parameters (i.e., pairs of low-rank matrices), $H_\phi(Z(c)) = \{A_{\ell,c}, B_{\ell,c}\}_{\ell=1}^{L}$. For each weight matrix W_ℓ, where $\ell \in [L]$, it is updated by its associated low-rank update $\Delta W_{\ell,c} = B_{\ell,c}A_{\ell,c}$, becoming $W_{\ell,c} = W_\ell + \Delta W_{\ell,c}$. Applying these updates across all layers gives the context-specific model parameters $\theta_c = \theta + \Delta\theta_c$, where $\Delta\theta_c = \{\Delta W_{\ell,c}\}_{\ell=1}^{L}$. The overall procedure is illustrated in Figure 1(i).

2.2 HYPERNETWORK ARCHITECTURE

We describe the internal structure of hypernetwork H_ϕ. For a document c with N tokens, each layer-wise representation is denoted as $Z_\ell(c) \in \mathbb{R}^{N\times D}$, where D denotes the hidden dimension of the target LLM. Demonstrated in Figure 1(ii), the hypernetwork can be decomposed into two parts: a Perceiver-style network (Jaegle et al., 2021) containing an input projection P and encoder/decoder E, and a LoRA generator G. The overall architecture can be summarized as:

$$Z_\ell(c) \xrightarrow{P} \widetilde{Z}_\ell(c) \xrightarrow{E} U_\ell(c) \xrightarrow{G} (A_{\ell,c}, B_{\ell,c}). \tag{1}$$

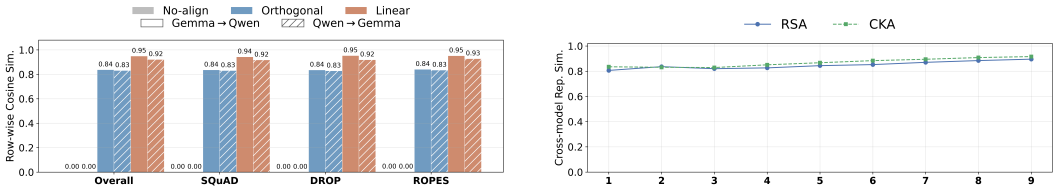
Input Projection. The projection P, which is implemented as an MLP, maps the backbone representations with dimension D into a fixed latent dimension d_h. In short, $\widetilde{Z}_\ell(c) = P(Z_\ell(c)) \in \mathbb{R}^{N\times d_h}$.

Perceiver Encoder/Decoder. The projected representations $\widetilde{Z}_\ell(c)$ are then processed by the Perceiver encocer/decoder E. The Perceiver uses r learned latent queries, where r denotes the LoRA rank, to compress the variable-length token sequence into a fixed set of latent representations. The latents can be denoted as $U_\ell(c) = E(\widetilde{Z}_\ell(c)) \in \mathbb{R}^{r\times d_h}$.

LoRA Generator. Finally, the generator G converts each Perceiver latent into the corresponding LoRA parameters. Specifically, G consists of a Post-Perceiver (PP) transformation F followed by a linear generation head G_{head}, i.e., $G = G_{\text{head}} \circ F$. For the adapted weight matrix W_ℓ at layer ℓ, it maps $U_\ell(c) \in \mathbb{R}^{r\times d_h}$ to $V_\ell(c) = G_\ell(U_\ell(c)) \in \mathbb{R}^{r\times(d_\ell^{\text{in}}+d_\ell^{\text{out}})}$, where d_ℓ^{in} and d_ℓ^{out} denote the input and output dimensions of W_ℓ, respectively. The output $V_\ell(c)$ is split along its last dimension to form the generated LoRA parameters: $A_{\ell,c} \in \mathbb{R}^{r\times d_\ell^{\text{in}}}$ and $B_{\ell,c} \in \mathbb{R}^{d_\ell^{\text{out}}\times r}$.

2.3 CROSS-ARCHITECTURE HYPERNETWORK TRANSFER

Given a trained source hypernetwork $H_{\phi_s}^{(s)}$, source backbone f_{θ_s}, and target backbone f_{θ_t}, our goal is to train a more effective target hypernetwork (denoted as $H_{\phi_t}^{(t)}$) **under limited data and compute**. We denote this transfer procedure by $\mathcal{T}$: $H_{\phi_t}^{(t)} = \mathcal{T}\left(H_{\phi_s}^{(s)}, f_{\theta_s}, f_{\theta_t}\right)$.



(a) Row-wise Cosine Similarity of paired projection outputs with different alignment strategies.

(b) Cross-model RSA and CKA stay high through the 9-layer Perceiver encoder.

Figure 2: Analysis of input projection and Perceivers. (a) An orthogonal/linear map can recover from the projection misalignment. (b) RSA/CKA remains high throughout the Perceiver encoder layers; thus, a direct copy transfers the key knowledge.

3 Motivating Analysis

In this section, we aim to understand which knowledge in a trained source hypernetwork $H_{\phi_s}^{(s)}$ can be reused to improve the training of a target hypernetwork $H_{\phi_t}^{(t)}$. Following the decomposition in Section 2.2, we analyze the hypernetwork at three stages. We first analyze the projected representations $\widetilde{Z}_\ell(c) = P(Z_\ell(c))$ to characterize the backbone-dependent projection P (Section 3.1). Next, we study the Perceiver representations $U_\ell(c)$ to determine whether the intermediate module E learns shared context representations (Section 3.2). Finally, we examine the generation component G to determine whether its learned mapping can be reused across architectures (Section 3.3).

Experiment Setup. We analyze the hypernetwork checkpoints[2] trained on `gemma-2-2b-it` (Team et al., 2024) and `Qwen3-4B-Instruct-2507` (Team, 2025), while holding out `Mistral-7B-Instruct-v0.2` (Jiang et al., 2023) from the main analysis to assess whether the resulting observations generalize to an unseen backbone[3]. We construct a fixed probe set of $N = 600$ documents by randomly sampling 200 examples from each of SQuAD (Rajpurkar et al., 2016), DROP (Dua et al., 2019) and ROPES (Lin et al., 2019)[4].

For each document c_i, we run the studied hypernetwork checkpoints and extract representations at each stage. For each backbone θ and a given stage k, we denote the corresponding per-document vector by $R_k^{(\theta)}(c_i) \in \mathbb{R}^{d_k^{(\theta)}}$, obtained by vectorizing the aggregated representation at that stage as needed. Stacking the N documents gives

$$R_k^{(\theta)} = \left[R_k^{(\theta)}(c_1) \mid R_k^{(\theta)}(c_2) \mid \cdots \mid R_k^{(\theta)}(c_N) \right]^\top \in \mathbb{R}^{N \times d_k^{(\theta)}}, \tag{2}$$

where $d_k^{(\theta)}$ denotes the representation dimension for model θ at stage k.

Metrics. We use three complementary measures to quantify cross-model alignment: *Representational Similarity Analysis (RSA)* (Kriegeskorte et al., 2008), which compares the pairwise relational structure among documents; *Linear Centered Kernel Alignment (Linear CKA)* (Kornblith et al., 2019), which measures similarity between representation spaces; and *Row-wise Cosine Similarity*, which evaluates how well one representation space can be linearly aligned to another on unseen documents. The formal definitions of these metrics are detailed in Appendix A.

3.1 Observation 1: Input Projection can be Linearly Aligned

The input projection maps backbone-specific hidden states into the shared Perceiver input space. Figure 2a shows that projection outputs from independently trained hypernetworks have near-zero raw cosine similarity, yet a linear map recovers high alignment on held-out documents.

[2]These trained checkpoints are given by the Doc-to-LoRA official repository.

[3]We refer to these backbones as **Gemma2-2B**, **Qwen3-4B** and **Mistral-7B**, unless otherwise noted.

[4]These are the evaluation datasets used in Doc-to-LoRA (Charakorn et al., 2026).

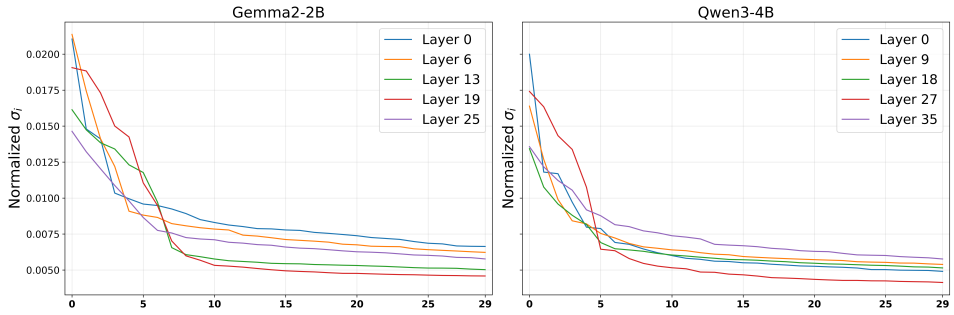


Figure 3: Weight singular value spectra of G_{head}. The x-axis denotes the singular value index and the y-axis shows the normalized singular value σ_i. The long-tail nature indicates that the head uses a near-full-rank subspace rather than a low-rank structure.

Takeaway. The projection mismatch across backbones is largely linear and can therefore be *aligned* with a linear map. More visualizations can be found in Appendix B.

3.2 Observation 2: Perceiver Learns a Transferable Representation

We next examine whether Perceivers trained independently on different backbones learn similar context representations. We compute RSA and CKA across corresponding Perceiver encoder layers and, as shown in Figure 2b, observe consistently high cross-model similarity throughout the encoder.

Takeaway. The Perceiver learns substantial shared representation structure, suggesting that its learned knowledge can be *preserved* across architectures.

3.3 Observation 3: LoRA Generator is Backbone-Specific

Finally, we examine the final linear head G_{head} within the LoRA generator G. Figure 3 shows a slowly decaying singular-value spectrum, indicating that the head uses nearly the full latent space rather than a small transferable subspace.

Takeaway. In contrast to the Perceiver, the LoRA generator remains strongly tied to the target backbone and thus needs to be *re-learned*.

4 HyperBridge: Cross-Architecture Hypernetwork Transfer

In summary, our analysis in Section 3 reveals a distinct stage-wise transferability across the hypernetwork: (i) the input projection exhibits a linear cross-backbone mismatch, (ii) the Perceiver learns representations that can be substantially shared across architectures, and (iii) the LoRA generator remains strongly target-specific.

Inspired by these observations, we introduce HyperBridge, a cross-backbone hypernetwork transfer method that transfers a source hypernetwork to a target backbone (illustrated in Figure 1(iii)). Specifically, HyperBridge (i) *aligns* the input projection, (ii) *preserves* the shared representation module, and (iii) *re-learns* the target-specific output generator. We detail these per-stage transfer operations in Section 4.1 and the final training objectives in Section 4.2.

4.1 HyperBridge's Per-Stage Transfer

Aligning the input projection. The source projection $P^{(s)}$ takes backbone representations $Z_\ell^{(s)}(c) \in \mathbb{R}^{N\times D_s}$ and the target backbone produces $Z_\ell^{(t)}(c) \in \mathbb{R}^{N\times D_t}$, where D_s, D_t denote the hidden dimensions of source/target backbones. We first adapt $P^{(s)}$ from D_s to D_t by direct truncation or padding, yielding an initial target projection $\bar{P}^{(t)}$. Then we fit a linear map $W_{\text{align}} \in \mathbb{R}^{d_h\times d_h}$ on the same probe set (detailed in the Setup of Section 3) and fold it into $\bar{P}^{(t)}$. The map can be derived as follows:

$$W_{\text{align}} = \arg\min_{W} \sum_{c \in \mathcal{D}_{\text{probe}}} \left\| \bar{P}^{(t)}\left(Z^{(t)}(c)\right) W - P^{(s)}\left(Z^{(s)}(c)\right) \right\|_F^2 + \lambda_{\text{align}} \|W\|_F^2. \tag{3}$$

Preserving the Perceiver core modules. We initialize the target Perceiver encoder/decoder directly from the source. To accommodate remaining target-specific shifts, we additionally consider attaching lightweight LoRA adapters to $E^{(t)}$.

Re-learning the LoRA generator. The LoRA generator G consists of a layer-dependent post-perceiver (PP) transformation F followed by the final generation head G_{head}. We directly initialize $F^{(t)}$ from $F^{(s)}$ using uniform layer interpolation to resolve the layer number mismatch (Jiao et al., 2020; 2021). For G_{head}, we simply do a re-initialization.

4.2 Training Procedure

We follow the standard context-internalization objective used in Charakorn et al. (2026) to force the generated LoRAs to reproduce the context-conditioned teacher behavior. Given a document c, query x, and teacher distribution $p_{\text{teacher}}(\cdot \mid x, c)$, the target hypernetwork generates $\Delta\theta_c^{(t)}$ and minimizes

$$\mathcal{L}_{\text{KD}} = \sum_{(c,x)\in\mathcal{D}} \sum_{j=1}^{|y|} \text{KL}\left(p_{\text{teacher}}^{(j)}(\cdot \mid x, c) \,\middle\|\, p_{\theta_t + \Delta\theta_c^{(t)}}^{(j)}(\cdot \mid x) \right). \tag{4}$$

The ℓ_1 regularization term can also be added to the LoRA factors, yielding the overall objective

$$\mathcal{L} = \mathcal{L}_{\text{KD}} + \lambda_{\text{reg}} \sum_{\ell} \left(\left\| A_{\ell,c}^{(t)} \right\|_1 + \left\| B_{\ell,c}^{(t)} \right\|_1 \right). \tag{5}$$

5 Experiment

5.1 Experiment Setup

LLM Backbones. We study cross-architecture transfer among three LLM backbones from distinct families, with different sizes, tokenizers, hidden dimensions, and number of layers: `gemma-2-2b-it` (Team et al., 2024), `Qwen3-4B-Instruct-2507` (Team, 2025) and `Mistral-7B-Instruct-v0.2` (Jiang et al., 2023). We consider all six transfer directions, in which each transfer reuses a trained source hypernetwork from Charakorn et al. (2026).

Datasets. Following Charakorn et al. (2026), the training datasets include a subset of FineWeb-Edu (Penedo et al., 2024) and full set of SQuAD (Rajpurkar et al., 2016), DROP (Dua et al., 2019) and ROPES (Lin et al., 2019). We pack documents into 512-token groups, each containing up to 4 QA pairs. We cap the FineWeb-Edu subset to at most 250K documents.

Training configuration. All hypernetworks are trained for up to 20K steps on 8 H100 GPUs with gradient accumulation of 4. We use AdamW at a learning rate 5×10^{-5} with 1000 warmup steps, gradient clipping 1.0, weight decay 0.01, and LoRA regularization coefficient $\lambda_{\text{reg}} = 0.1$.

Evaluation and metrics. We evaluate each model with greedy decoding and max_tokens $= 256$ on 500 test examples per dataset[5]. Following Charakorn et al. (2026), if the target model is Gemma2-2B, we report ROUGE-L F1 scores; if the target model is Qwen3-4B or Mistral-7B, we report ROUGE-L recall scores.

Baseline and HyperBridge. The *Native* baseline is a target hypernetwork **trained from scratch**. For HyperBridge, we default to aligning the input with a *linear mapping* and a *rank-64* LoRA on the Perceiver encoder/decoder[6]. For each target backbone, we apply HyperBridge to the two remaining backbones as sources (denoted as `source` $\rightarrow$ in the tables).

[5] Same settings as in Charakorn et al. (2026).

[6] Following (Hu et al., 2022) LoRA adapters are only applied to the query and value projection matrices.

Table 1: HYPERBRIDGE vs. *Native (from-scratch training)* at 5K, 10K, 15K and 20K steps. For each target the two rows below *Native* are HYPERBRIDGE from two source models. **Bold** and underscore denote top-1/2 accuracy.

Target	Method	SQuAD				DROP				ROPES				Avg.			
		5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Gemma2-2B	No context	0.250				0.135				0.383				0.256			
	Native	0.392	0.450	0.494	0.498	0.220	0.255	0.287	0.294	0.410	0.429	0.477	0.488	0.341	0.378	0.419	0.427
	Qwen3-4B→	**0.458**	0.449	0.479	0.464	0.249	**0.285**	0.285	0.287	0.415	0.470	0.481	0.479	0.374	0.401	0.415	0.410
	Mistral-7B→	0.444	**0.486**	**0.501**	**0.499**	**0.255**	0.271	**0.299**	**0.303**	**0.467**	**0.486**	**0.490**	**0.503**	**0.389**	**0.414**	**0.430**	**0.435**
Qwen3-4B	No context	0.243				0.140				0.440				0.274			
	Native	0.332	0.365	0.426	0.434	0.252	**0.299**	**0.330**	**0.332**	0.481	0.499	0.517	0.533	0.355	0.388	0.424	0.433
	Gemma2-2B→	**0.402**	**0.462**	0.488	**0.505**	**0.295**	0.294	0.311	0.308	**0.526**	**0.542**	**0.550**	0.536	**0.408**	**0.433**	**0.449**	0.450
	Mistral-7B→	0.380	0.459	**0.490**	0.488	0.270	0.298	0.313	0.309	0.501	0.541	0.540	**0.559**	0.383	0.433	0.448	**0.452**
Mistral-7B	No context	0.411				0.188				0.380				0.326			
	Native	0.445	0.478	0.533	0.568	0.250	0.283	0.285	0.310	0.426	0.428	0.444	0.449	0.374	0.397	0.421	0.442
	Gemma2-2B→	**0.591**	**0.643**	**0.663**	**0.649**	**0.298**	**0.344**	**0.348**	**0.364**	**0.487**	**0.543**	**0.597**	**0.598**	**0.459**	**0.510**	**0.536**	**0.537**
	Qwen3-4B→	0.572	0.595	0.612	0.620	0.295	0.321	0.340	0.354	0.467	0.498	0.512	0.528	0.445	0.471	0.488	0.501

5.2 MAIN RESULTS

HYPERBRIDGE outperforms *Native* across datasets. Table 1 presents our main results. Across datasets, we observe that HYPERBRIDGE substantially improves QA performance for most transfer directions, with both source hypernetworks outperforming the corresponding *Native* baseline. The main advantage of HYPERBRIDGE stems from reusing backbone-shared representations learned by the source hypernetwork, which provides a stronger initialization and improves learnability for target hypernetwork training.

HYPERBRIDGE surpasses *Native* under different training budgets. We also study how HYPERBRIDGE performs across training steps. At 5K and 10K steps, HYPERBRIDGE improves performance by 14.7% and 14.3% on average across all transfer directions, respectively. This advantage persists as the training budget increases and is especially pronounced on the Mistral-7B backbone, where HYPERBRIDGE yields up to 28.5% relative gains across data budgets. In addition, HYPERBRIDGE can reach competitive quality with fewer updates: the best 5K transfer already surpasses *Native* trained for 10K steps on all three targets, and on Mistral even exceeds *Native* trained for 20K steps. These trends show that HYPERBRIDGE substantially improves the data efficiency of hypernetwork adaptation.

5.3 ABLATION STUDY

In this subsection, we ablate the key design choices of HYPERBRIDGE to understand their individual contributions. Specifically, we study the effects of input alignment (Section 5.3.1), Perceiver adaptation strategy (Section 5.3.2) and training data composition (Section 5.3.3), varying one factor at a time while keeping the remaining configuration fixed.

Setup. For brevity, we consider `Qwen3-4B-Instruct-2507` (i.e. **Qwen3-4B**) as the target backbone in the main text. Results on two other backbones are in Appendix C. Unless otherwise specified, each ablation starts from the default HYPERBRIDGE configuration and varies only one factor at a time. We report ROUGE-L recall scores on the test split at 5K, 10K, 15K, and 20K steps.

5.3.1 INPUT ALIGNMENT

We study how the source projection aligns with the target backbone. We compare: (i) **Direct**, which directly adapts dimension mismatch via truncation/padding; (ii) **Orthogonal**, which fits an orthogonal mapping; and (iii) **Linear**, which fits an unconstrained linear mapping. The *Native* row denotes a target hypernetwork trained from scratch under the same budget.

As shown in Table 2, both Orthogonal and Linear alignment mostly yield better results across training budgets , indicating that aligning the representation at the input projection stage can improve training on the target backbone. We observe that *Direct* can still be competitive in some cases, likely because subsequent target training can compensate for moderate representation mismatch, while a fitted mapping may introduce estimation error.

Table 2: Input-alignment ablation at 5K, 10K, 15K and 20K steps. The target model is Qwen3-4B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.332	0.365	0.426	0.434	0.252	0.299	0.330	0.332	0.481	0.499	0.517	0.533	0.355	0.388	0.424	0.433
Gemma2-2B	Direct	**0.411**	0.474	0.472	0.479	**0.308**	0.287	0.320	**0.319**	0.524	0.538	0.537	0.540	**0.414**	0.433	0.443	0.446
	Orthogonal	0.398	**0.476**	0.482	0.485	**0.298**	**0.319**	**0.323**	**0.331**	**0.529**	**0.565**	**0.568**	**0.582**	**0.408**	**0.453**	**0.458**	**0.466**
	Linear	**0.402**	0.462	0.488	**0.505**	0.295	0.294	0.311	0.308	0.526	**0.542**	0.550	0.536	0.408	0.433	0.449	0.450
Mistral-7B	Direct	0.387	0.445	0.473	0.462	0.289	**0.307**	**0.329**	0.316	**0.526**	0.540	**0.550**	**0.565**	0.401	0.431	**0.450**	0.448
	Orthogonal	0.396	**0.497**	**0.489**	0.486	0.291	0.297	0.281	0.292	0.514	0.532	0.545	0.546	0.400	**0.442**	0.438	0.441
	Linear	0.380	0.459	**0.490**	**0.488**	0.270	0.298	0.313	0.309	0.501	0.541	0.540	0.559	0.383	0.433	0.448	**0.452**

Table 3: Perceiver-adaptation ablation at 5K, 10K, 15K and 20K steps. The target model is Qwen3-4B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.332	0.365	0.426	0.434	0.252	0.299	0.330	0.332	0.481	0.499	0.517	0.533	0.355	0.388	0.424	0.433
Gemma2-2B	Frozen (random init)	0.277	0.280	0.296	0.293	0.209	0.231	0.229	0.241	0.474	0.480	0.483	0.484	0.320	0.330	0.336	0.339
	Full fine-tune	**0.420**	0.480	0.494	0.500	**0.314**	0.309	0.324	0.309	0.525	0.533	0.522	0.532	**0.420**	0.441	0.447	0.447
	Frozen (no LoRA)	**0.415**	0.476	0.483	0.486	0.300	**0.317**	0.308	**0.321**	0.523	0.540	0.540	0.537	0.412	0.444	0.444	0.448
	LoRA rank-16	0.414	0.476	0.483	0.498	0.305	0.309	**0.330**	0.311	**0.532**	0.548	0.529	0.539	**0.417**	0.444	0.447	0.449
	LoRA rank-64	0.402	0.462	0.488	**0.505**	0.295	0.294	0.311	0.308	0.526	0.542	**0.550**	0.536	0.408	0.433	0.449	0.450
Mistral-7B	Frozen (random init)	0.285	0.301	0.295	0.287	0.241	0.229	0.232	0.232	0.502	0.497	0.487	0.501	0.343	0.342	0.338	0.340
	Full fine-tune	0.385	0.485	**0.509**	**0.522**	0.301	0.298	0.320	0.315	0.521	**0.552**	0.542	0.531	0.403	**0.445**	**0.457**	0.456
	Frozen (no LoRA)	0.385	**0.498**	**0.500**	0.497	**0.308**	**0.309**	**0.326**	**0.332**	0.518	**0.557**	0.549	**0.559**	0.404	**0.455**	**0.458**	**0.463**
	LoRA rank-16	0.389	**0.493**	0.479	0.492	0.288	0.302	0.304	0.317	**0.527**	0.536	**0.556**	**0.564**	0.401	0.443	0.446	**0.458**
	LoRA rank-64	0.380	0.459	0.490	0.488	0.270	0.298	0.313	0.309	0.501	0.541	0.540	0.559	0.383	0.433	0.448	0.452

5.3.2 PERCEIVER ADAPTATION STRATEGY

We next vary how we adapt the transferred Perceiver to the target backbone. We compare: (i) **Frozen (random-init)**, which uses a randomly initialized and frozen Perceiver; (ii) **Full fine-tuning**, which updates all Perceiver parameters; (iii) **Frozen (no LoRA)**, which directly reuses the source Perceiver without additional learnable parameters; and (iv) **LoRA rank-16/64**, which adds rank-16 or rank-64 adapters to the copied Perceiver.

As shown in Table 3, randomly initializing the Perceiver substantially degrades performance for both source backbones, confirming that the transferred Perceiver weights contain critical cross-backbone knowledge. Additionally, *directly freezing the Perceiver already performs strongly*, while its LoRA variants provide competitive or improved performance later in training without requiring full fine-tuning. Overall, these results show that preserving the source Perceiver is important for better training on the target backbone.

5.3.3 TRAINING DATA COMPOSITION

Finally, we vary the amount of data available for training the target hypernetwork. We consider: (i) **FW-100K**, (ii) **FW-100K + QA**, (iii) **FW-250K**, and (iv) **FW-250K + QA**. Here, **QA** data refers to the full in-domain training sets (i.e., SQuAD, DROP, and ROPES). For each data setting, we compare HYPERBRIDGE against *Native*, which trains a target hypernetwork from scratch under the same data budget.

As shown in Table 4, HYPERBRIDGE remains effective across all four data settings and generally provides larger performance gains when the data is more limited. In particular, under **FW-100K**, both source transfers substantially outperform *Native* throughout training, indicating that HYPERBRIDGE can compensate for reduced training data.

5.4 DISCUSSION

Transfer effectiveness can depend on specific source–target pair. From Table 1, we can observe that Gemma2-2B and Mistral-7B exhibit strong mutual transfer: both transfer directions outperform their *Native* baselines across all compute budgets. Transfer to Qwen3-4B is more source-dependent:

Table 4: Data ablation at 5K, 10K, 15K and 20K steps. The target model is Qwen3-4B. **Bold** and underscore denote top-1/2 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Data	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
FW-100K	Native	0.293	0.295	0.247	0.233	0.191	0.182	0.182	0.172	0.461	0.453	0.495	0.501	0.315	0.310	0.308	0.302
	Gemma2-2B→	**0.405**	**0.384**	**0.399**	**0.390**	**0.215**	0.188	0.203	0.196	0.492	**0.509**	0.505	**0.512**	**0.371**	**0.360**	0.369	**0.366**
	Mistral-7B→	0.377	0.377	0.386	0.365	0.182	**0.209**	**0.214**	**0.196**	**0.512**	0.494	**0.516**	0.505	0.357	0.360	**0.372**	0.356
FW-100K + QA	Native	0.315	0.331	0.340	0.325	0.279	0.327	**0.355**	**0.355**	**0.541**	**0.553**	**0.576**	**0.579**	0.378	0.404	0.424	0.420
	Gemma2-2B→	**0.401**	**0.441**	0.434	0.423	**0.287**	0.328	0.327	0.331	0.511	0.544	0.547	0.541	**0.400**	**0.438**	0.436	0.432
	Mistral-7B→	0.369	0.434	**0.444**	**0.427**	0.280	**0.341**	0.339	0.355	0.519	0.535	0.532	0.535	0.389	0.437	**0.438**	**0.439**
FW-250K	Native	0.282	0.326	0.357	0.370	0.168	0.197	0.208	0.203	0.455	0.455	0.479	0.502	0.302	0.326	0.348	0.358
	Gemma2-2B→	**0.428**	**0.471**	0.470	**0.478**	0.206	0.213	0.222	0.222	**0.515**	0.520	0.515	0.517	**0.383**	0.401	0.403	0.406
	Mistral-7B→	0.387	0.458	**0.480**	0.476	**0.208**	**0.222**	**0.234**	**0.235**	0.497	**0.526**	**0.527**	**0.523**	0.364	**0.402**	**0.414**	**0.411**
FW-250K + QA	Native	0.332	0.365	0.426	0.434	0.252	**0.299**	**0.330**	**0.332**	0.481	0.499	0.517	0.533	0.355	0.388	0.424	0.433
	Gemma2-2B→	**0.402**	**0.462**	0.488	**0.505**	**0.295**	0.294	0.311	0.308	**0.526**	**0.542**	**0.550**	0.536	**0.408**	**0.433**	**0.449**	0.450
	Mistral-7B→	0.380	0.459	**0.490**	0.488	0.270	0.298	0.313	0.309	0.501	0.541	0.540	**0.559**	0.383	0.433	0.448	**0.452**

Mistral-7B consistently improves over *Native* while Gemma2-2B provides stronger gains primarily during early training (e.g., on DROP). This shows that reusable hypernetwork knowledge exists across architectures, but its benefit can vary by source–target backbone combination.

6 Related Work

6.1 Hypernetworks for LLMs

Hypernetworks provide an efficient alternative to conventional fine-tuning by directly generating model adaptations conditioned on task or context information (Mahabadi et al., 2021; Abdalla et al., 2025; Tan et al., 2026). Existing approaches can be broadly divided into two groups. *Task-conditioned hypernetworks* generate parameter-efficient modules from task descriptions, demonstrations, or instructions, enabling rapid adaptation without per-task optimization (Ivison & Peters, 2022; Phang et al., 2023; Lv et al., 2024). More recently, *context-conditioned hypernetworks* map richer textual contexts directly into persistent model adaptations, such as Text-to-LoRA (Charakorn et al., 2025), SHINE (Liu et al., 2026), and Doc-to-LoRA (Charakorn et al., 2026), which generate context- or document-specific LoRA parameters in a single forward pass. Despite different parameterizations, these methods only learn hypernetworks for a fixed target backbone.

6.2 Cross-Model Adaptation for LLMs

A complementary line of work studies how model-specific adaptations can be reused when the underlying backbone changes (Huang et al., 2025). Existing methods mainly follow three paradigms. *Data-driven migration* reconstructs supervision or performs limited re-optimization to recover an adapter for the target model, as in Trans-LoRA (Wang et al., 2024) and ReLoRA (Xu et al., 2026). *Structure-based transfer* exploits correspondences between source and target models, such as layer matching, transfer matrices, or low-rank subspace alignment, to transport learned adaptations across heterogeneous parameter spaces (Li et al., 2026; Xia et al., 2025). *Multi-backbone generation* instead jointly learns a shared generator over multiple LLMs to synthesize model-specific adaptations and generalize to unseen backbones, as exemplified by TeleLoRA (Lin et al., 2025). However, to the best of our knowledge, prior work has not studied transfer across hypernetworks associated with different LLM backbones.

7 Conclusion

In this paper, we study how knowledge learned by a context-internalization hypernetwork can be reused when adapting to a new LLM backbone under limited data and compute. We conduct a thorough analysis and propose HyperBridge, which provides a stronger initialization for target training while relearning only the components that are inherently backbone-specific. Across multiple LLM architectures, datasets, and training budgets, HyperBridge consistently matches or improves training a target hypernetwork from scratch. For future work, we envision the existence of *hyper foundation models*, which can serve various model backbones at once.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting and polishing the manuscript, improving readability, analyzing and summarizing existing literature, and identifying potentially relevant prior work. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, design the research methodology or experiments, implement the proposed methods, clean or reformat datasets, conduct qualitative or thematic data analysis, or interpret experimental results; the remaining required-disclosure tasks are not applicable to this work. All AI-assisted text and literature suggestions were reviewed and verified by the authors, including checking cited and suggested prior work against the original sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

During submission, we will provide our source code as supplementary material in a zip file. Section 5.1 provides details of the datasets, including their references and evaluation setup. Section 5.1 also describes implementation details of the compared methods and their training hyperparameters.

REFERENCES

Mohamed Hesham Ibrahim Abdalla, Zhipin Wang, Christian Frey, Steffen Eger, and Josif Grabocka. Zhyper: Factorized hypernetworks for conditioned llm fine-tuning. *arXiv preprint arXiv:2510.19733*, 2025.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pp. 1–4. Springer, 2009.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. Training plug-n-play knowledge modules with deep context distillation. *arXiv preprint arXiv:2503.08727*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-lora: Instant transformer adaption. *arXiv preprint arXiv:2506.06105*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Youcheng Huang, Chen Huang, Duanyu Feng, Wenqiang Lei, and Jiancheng Lv. Cross-model transferability among large language models on the platonic representations of concepts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3686–3704, 2025.

Hamish Ivison and Matthew E Peters. Hyperdecoders: Instance-specific decoders for multi-task nlp. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1715–1730, 2022.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

Xiaoqi Jiao, Huating Chang, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Improving task-agnostic bert distillation with layer mapping search. *Neurocomputing*, 461:194–203, 2021.

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

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

Kalle Kujanpää, Pekka Marttinen, Harri Valpola, and Alexander Ilin. Efficient knowledge injection in LLMs via self-distillation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=drYpdSnRJk.

Yanan Li, Fanxu Meng, Muhan Zhang, Shiai Zhu, Shangguang Wang, and Mengwei Xu. Lorasuite: Efficient lora adaptation across large language model upgrades. *Advances in Neural Information Processing Systems*, 38:170149–170176, 2026.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pp. 58–62, 2019.

Xiao Lin, Manoj Acharya, Anirban Roy, and Susmit Jha. Telelora: Teleporting model-specific alignment across llms. *arXiv preprint arXiv:2503.20228*, 2025.

Yewei Liu, Xiyuan Wang, Yansheng Mao, Yoav Gelberg, Haggai Maron, and Muhan Zhang. SHINE: A scalable in-context hypernetwork for mapping context to loRA in a single pass. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZMexYcAibv.

Chuancheng Lv, Lei Li, Shitou Zhang, Gang Chen, Fanchao Qi, Ningyu Zhang, and Hai-Tao Zheng. Hyperlora: Efficient cross-task generalization via constrained low-rank adapters generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16376–16393, 2024.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 565–576, 2021.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*, pp. 27854–27875. pmlr, 2023.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions. In *International Conference on Learning Representations*, volume 2025, pp. 77563–77585, 2025.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Haebin Shin, Lei Ji, Yeyun Gong, Sungdong Kim, Eunbi Choi, and Minjoon Seo. 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)*, pp. 7338–7363, 2025.

Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, et al. Instant personalized large language model adaptation via hypernetwork. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23557–23580, 2026.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Runqian Wang, Soumya Ghosh, David Cox, Diego Antognini, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. Trans-lora: towards data-free transferable parameter efficient finetuning. *Advances in Neural Information Processing Systems*, 37:61217–61237, 2024.

Feifan Xia, Mingyang Liao, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Cross-lora: A data-free lora transfer framework across heterogeneous llms. *arXiv preprint arXiv:2508.05232*, 2025.

Yang Xu, Zihuai Xu, Hongli Xu, Yunming Liao, Zhiwei Yao, and Xitong Fu. Relora: Knowledge-reusing adaptation for fast rollout of evolving llm services. *arXiv preprint arXiv:2606.02606*, 2026.

A METRICS FOR ANALYSIS

- *Representational Similarity Analysis (RSA) (Kriegeskorte et al., 2008).* For backbone f_θ, we construct a document-similarity matrix: $S_k^{(\theta)}$ where each ij-th element computes a cosine similarity: $[S_k^{(\theta)}]_{ij} = \cos\Big(R_k^{(\theta)}(c_i), R_k^{(\theta)}(c_j)\Big)$. Given two backbones f_{θ_a} and f_{θ_b}, RSA is computed as the Pearson correlation (Benesty et al., 2009) between the off-diagonal entries of their similarity matrices:

$$\mathrm{RSA}_k(\theta_a, \theta_b) = \mathrm{Corr}\big(\{[S_k^{(\theta_a)}]_{ij} : i < j\}, \{[S_k^{(\theta_b)}]_{ij} : i < j\}\big). \tag{6}$$

- *Linear Centered Kernel Alignment (Linear CKA) (Kornblith et al., 2019).* Let $R_k^{(\theta_a)}$ and $R_k^{(\theta_b)}$ denote the column-centered representation matrices of two backbones at stage k. Linear CKA is defined as

$$\mathrm{CKA}_k(\theta_a, \theta_b) = \frac{\left\| {R_k^{(\theta_a)}}^\top R_k^{(\theta_b)} \right\|_F^2}{\left\| {R_k^{(\theta_a)}}^\top R_k^{(\theta_a)} \right\|_F \left\| {R_k^{(\theta_b)}}^\top R_k^{(\theta_b)} \right\|_F}. \tag{7}$$

- *Row-wise Cosine Similarity.* Given two backbones f_{θ_a} and f_{θ_b}, we randomly split the N documents into 75% training and 25% test sets. We fit a through-origin ridge map M_k from the stage-k representation space of θ_a to that of θ_b:

$$M_k = \arg\min_M \left\| R_{k,\mathrm{tr}}^{(\theta_a)} M - R_{k,\mathrm{tr}}^{(\theta_b)} \right\|_F^2 + \lambda \|M\|_F^2, \tag{8}$$

  where λ is the ridge regularization coefficient (default to 1.0). We then report the mean cosine similarity between the mapped source representations and the target representations on the held-out documents:

$$\mathrm{Cos}_k(\theta_a \to \theta_b) = \frac{1}{|\mathcal{I}_{\mathrm{te}}|} \sum_{i \in \mathcal{I}_{\mathrm{te}}} \cos\left(R_k^{(\theta_a)}(c_i) M_k, R_k^{(\theta_b)}(c_i)\right), \tag{9}$$

  where $\mathcal{I}_{\mathrm{te}}$ denotes the indices of the held-out documents.

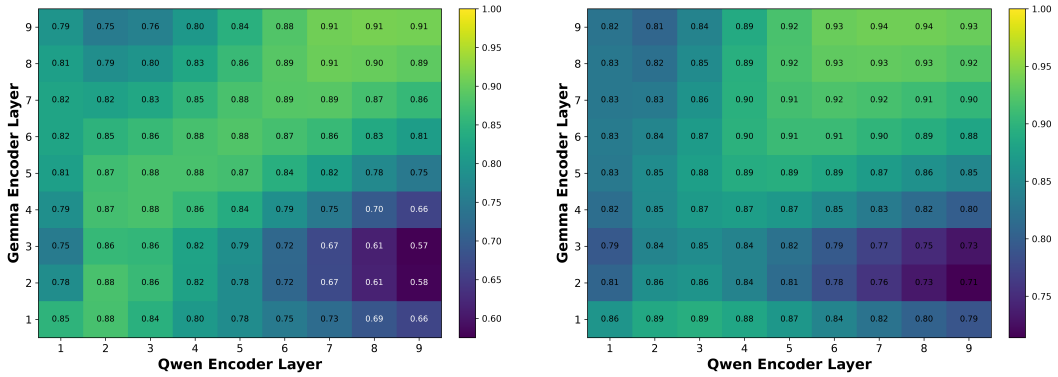
B VISUALIZATION

Per-layer alignment. For each ordered pair (i.e. Gemma2-2B encoder layer i and Qwen3-4B encoder layer j), we compare the two encoders' mean-pooled outputs over the same 600 documents drawn from the three QA datasets. Linear CKA and RSA are both measured. Figure 4 shows that both metrics increase toward the deeper layers of the two perceiver encoders.

Shared document geometry. We take the normalized final output of the perceiver-encoder for the same 600 documents of each model and jointly embed both models with t-SNE. Figure 5 shows that the raw outputs are concatenated, but a single closed-form linear alignment allows data points from both models to coincide and cluster by dataset rather than by model.

C ABLATIONS ON MORE BACKBONES

In this section, we conduct the ablation study on the two remaining backbones (i.e., Gemma2-2B and Mistral-7B). We observe trends consistent with those reported in the main text. As discussed in Section 5.4, transfer between Gemma2-2B and Mistral-7B is generally more effective, whereas using Qwen as a source model is more challenging, likely due to its distinctively verbose generation style. For Gemma2-2B as the target model, the ablations on input alignment, perceiver adaptation, and data are presented in Table 5, 6 and 7. For Mistral-7B as the target model, the ablations on input alignment, perceiver adaptation, and data are presented in Table 8, 9 and 10.



(a) RSA (b) CKA

Figure 4: Cross-model alignment metrics across all perceiver-encoder layer pairs. It is shown that both metrics remain high across the layers.

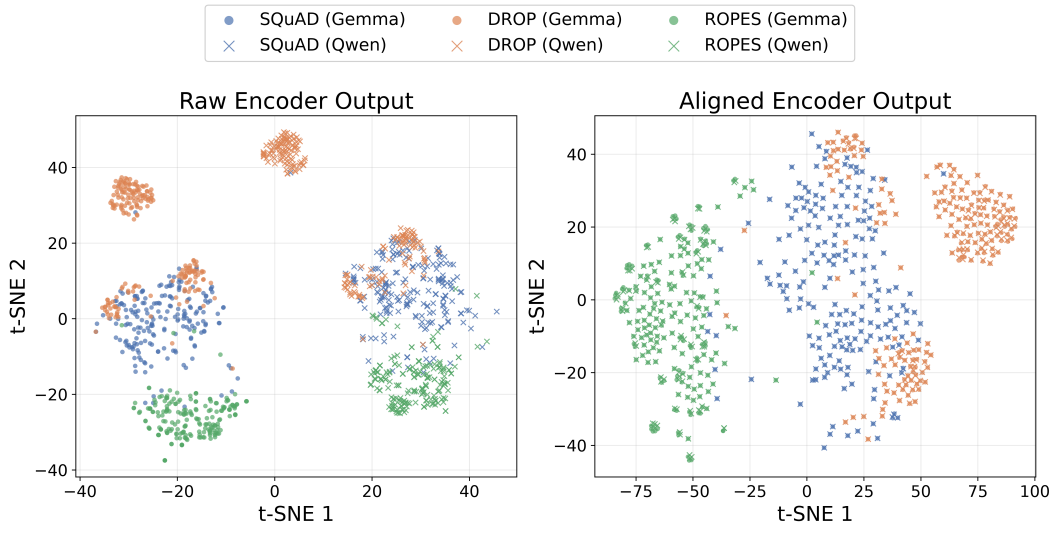


Figure 5: Perceiver encoder outputs across models have a shared document geometry. *Left* shows the raw outputs, where two models form separate islands; while *Right* demonstrates that a linear alignment can pair document representations across backbones perfectly.

Table 5: Input-alignment ablation at 5K, 10K, 15K and 20K steps. The target model is Gemma2-2B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.392	0.450	0.494	0.498	0.220	0.255	0.287	0.294	0.410	0.429	0.477	0.488	0.341	0.378	0.419	0.427
Qwen3-4B	Direct	0.425	0.462	0.462	0.464	**0.255**	0.266	0.275	0.283	0.448	**0.484**	0.479	0.479	0.376	0.404	0.405	0.409
	Orthogonal	0.417	0.451	0.488	0.470	0.236	0.254	0.271	0.272	0.435	0.473	**0.491**	0.486	0.363	0.393	0.417	0.409
	Linear	**0.458**	0.449	0.479	0.464	0.249	**0.285**	0.285	**0.287**	0.415	0.470	0.481	0.479	0.374	0.401	0.415	0.410
Mistral-7B	Direct	0.441	**0.476**	**0.509**	**0.517**	**0.264**	**0.278**	**0.306**	0.280	**0.458**	0.466	**0.498**	**0.497**	**0.387**	**0.406**	**0.438**	**0.431**
	Orthogonal	0.436	0.473	0.494	0.490	0.253	0.260	0.275	0.276	0.434	0.478	0.490	0.488	0.374	0.404	0.420	0.418
	Linear	**0.444**	**0.486**	**0.501**	**0.499**	0.255	0.271	**0.299**	**0.303**	**0.467**	**0.486**	0.490	**0.503**	**0.389**	**0.414**	**0.430**	**0.435**

Table 6: Perceiver-adaptation ablation at 5K, 10K, 15K and 20K steps. The target model is Gemma2-2B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.392	0.450	0.494	0.498	0.220	0.255	0.287	0.294	0.410	0.429	0.477	0.488	0.341	0.378	0.419	0.427
Qwen3-4B	Frozen (random init)	0.295	0.326	0.345	0.353	0.192	0.215	0.220	0.222	0.398	0.410	0.441	0.437	0.295	0.317	0.335	0.337
	Full fine-tune	0.397	0.466	0.498	0.482	0.250	**0.287**	0.284	0.282	0.448	0.464	**0.495**	0.489	0.365	0.406	0.426	0.418
	Frozen (no LoRA)	0.418	0.479	0.479	0.473	**0.257**	0.278	0.292	**0.306**	0.456	0.474	0.491	0.487	0.377	0.410	0.421	0.422
	LoRA rank-16	0.434	**0.508**	**0.515**	**0.512**	0.244	0.277	**0.305**	**0.304**	**0.468**	0.481	**0.497**	0.499	**0.382**	**0.422**	**0.439**	**0.438**
	LoRA rank-64	**0.458**	0.449	0.479	0.464	0.249	0.285	0.285	0.287	0.415	0.470	0.481	0.479	0.374	0.401	0.415	0.410
Mistral-7B	Frozen (random init)	0.320	0.324	0.353	0.353	0.201	0.217	0.227	0.236	0.381	0.402	0.415	0.409	0.301	0.314	0.332	0.333
	Full fine-tune	**0.457**	0.459	0.502	**0.505**	0.234	0.272	0.281	0.285	0.426	0.472	0.474	0.471	0.372	0.401	0.419	0.421
	Frozen (no LoRA)	0.429	0.467	**0.507**	0.498	**0.255**	0.266	0.277	0.278	0.458	0.478	0.476	0.484	0.381	0.404	0.420	0.420
	LoRA rank-16	0.425	0.466	0.498	0.499	0.244	**0.299**	0.286	0.286	0.448	**0.494**	0.490	**0.516**	0.372	**0.420**	0.425	0.433
	LoRA rank-64	0.444	**0.486**	0.501	0.499	0.255	0.271	**0.299**	0.303	**0.467**	**0.486**	0.490	**0.503**	**0.389**	0.414	**0.430**	**0.435**

Table 7: Data ablation at 5K, 10K, 15K and 20K steps. The target model is Gemma2-2B. **Bold** and underscore denote top-1/2 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Data	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
FW-100K	Native	0.297	0.273	0.245	0.210	0.135	0.129	0.130	0.136	0.344	0.366	0.334	0.335	0.259	0.256	0.236	0.227
	Qwen3-4B→	0.395	0.391	0.387	0.386	**0.182**	**0.194**	0.183	**0.187**	0.404	0.425	0.421	0.419	0.327	0.337	0.331	0.331
	Mistral-7B→	**0.444**	**0.425**	**0.415**	**0.413**	0.172	0.181	**0.184**	0.186	**0.462**	**0.471**	**0.475**	**0.469**	**0.359**	**0.359**	**0.358**	**0.356**
FW-100K + QA	Native	0.341	0.391	0.358	0.368	0.232	0.274	0.266	0.275	0.378	0.406	0.405	0.417	0.317	0.357	0.343	0.353
	Qwen3-4B→	**0.437**	**0.459**	0.430	0.431	0.234	0.287	0.276	0.275	**0.438**	**0.491**	**0.477**	**0.482**	**0.370**	**0.412**	0.394	**0.396**
	Mistral-7B→	0.404	0.443	**0.444**	**0.433**	**0.257**	**0.302**	**0.283**	**0.276**	0.435	0.465	0.470	0.460	0.365	0.403	**0.399**	0.390
FW-250K	Native	0.282	0.383	0.428	0.439	0.149	0.156	0.160	0.177	0.369	0.367	0.376	0.401	0.267	0.302	0.321	0.339
	Qwen3-4B→	0.434	0.474	**0.494**	**0.483**	**0.230**	**0.220**	**0.222**	**0.206**	0.410	0.444	0.455	0.451	**0.358**	**0.380**	**0.390**	0.380
	Mistral-7B→	**0.434**	**0.484**	0.481	0.480	0.184	0.199	0.216	0.202	**0.416**	**0.446**	**0.457**	**0.470**	0.345	0.377	0.385	**0.384**
FW-250K + QA	Native	0.392	0.450	0.494	0.498	0.220	0.255	0.287	0.294	0.410	0.429	0.477	0.488	0.341	0.378	0.419	0.427
	Qwen3-4B→	**0.458**	0.449	0.479	0.464	0.249	**0.285**	0.285	0.287	0.415	0.470	0.481	0.479	0.374	0.401	0.415	0.410
	Mistral-7B→	0.444	**0.486**	**0.501**	**0.499**	**0.255**	0.271	**0.299**	**0.303**	**0.467**	**0.486**	**0.490**	**0.503**	**0.389**	**0.414**	**0.430**	**0.435**

Table 8: Input-alignment ablation at 5K, 10K, 15K and 20K steps. The target model is Mistral-7B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.445	0.478	0.533	0.568	0.250	0.283	0.285	0.310	0.426	0.428	0.444	0.449	0.374	0.397	0.421	0.442
Gemma2-2B	Direct	0.565	0.604	**0.637**	0.643	0.284	0.340	0.355	**0.372**	0.480	**0.577**	0.583	**0.592**	0.443	**0.507**	0.525	**0.535**
	Orthogonal	0.538	0.613	0.618	**0.643**	0.284	**0.350**	**0.376**	0.363	**0.489**	**0.553**	**0.583**	0.554	0.437	0.505	**0.526**	0.520
	Linear	**0.591**	**0.643**	**0.663**	**0.649**	**0.298**	0.344	0.348	0.364	0.487	0.543	**0.597**	**0.598**	**0.459**	**0.510**	**0.536**	**0.537**
Qwen3-4B	Direct	0.554	0.607	0.621	0.640	0.285	0.323	0.352	0.361	0.481	0.517	0.542	0.561	0.440	0.482	0.505	0.521
	Orthogonal	0.548	**0.633**	0.623	0.639	**0.305**	**0.347**	**0.363**	**0.374**	**0.494**	0.528	0.549	0.557	**0.449**	0.503	0.512	0.523
	Linear	**0.572**	0.595	0.612	0.620	0.295	0.321	0.340	0.354	0.467	0.498	0.512	0.528	0.445	0.471	0.488	0.501

Table 9: Perceiver-adaptation ablation at 5K, 10K, 15K and 20K steps. The target model is Mistral-7B. **Bold with underscore**, **bold** and underscore denote top-1/2/3 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Source	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
Native		0.445	0.478	0.533	0.568	0.250	0.283	0.285	0.310	0.426	0.428	0.444	0.449	0.374	0.397	0.421	0.442
Gemma2-2B	Frozen (random init)	0.475	0.495	0.513	0.525	0.229	0.237	0.257	0.259	0.436	0.447	0.470	0.482	0.380	0.393	0.413	0.422
	Full fine-tune	0.547	0.608	0.633	0.633	0.307	0.317	0.334	0.343	0.466	0.508	0.526	0.531	0.440	0.477	0.498	0.503
	Frozen (no LoRA)	0.560	0.601	0.616	0.632	0.286	0.331	**0.351**	0.356	0.459	0.525	0.539	0.556	0.435	0.485	0.502	0.515
	LoRA rank-16	**0.586**	0.622	0.636	0.635	**0.324**	**0.351**	**0.365**	**0.363**	**0.499**	**0.541**	0.536	0.542	**0.470**	**0.505**	0.512	0.513
	LoRA rank-64	**0.591**	**0.643**	**0.663**	**0.649**	0.298	**0.344**	0.348	**0.364**	**0.487**	**0.543**	**0.597**	**0.598**	0.459	**0.510**	**0.536**	**0.537**
Qwen3-4B	Frozen (random init)	0.489	0.495	0.501	0.506	0.234	0.235	0.239	0.257	0.436	0.452	0.454	0.494	0.386	0.394	0.398	0.419
	Full fine-tune	0.550	0.614	0.614	0.628	0.271	0.326	0.338	0.333	0.472	0.502	0.531	0.551	0.431	0.481	0.494	0.504
	Frozen (no LoRA)	0.549	**0.638**	**0.647**	**0.655**	0.313	0.319	0.347	0.355	0.486	0.528	**0.553**	**0.559**	0.449	0.495	**0.516**	**0.523**
	LoRA rank-16	0.585	0.618	0.615	0.634	**0.320**	0.327	0.340	0.344	0.480	0.515	0.530	0.555	**0.462**	0.487	0.495	0.511
	LoRA rank-64	0.572	0.595	0.612	0.620	0.295	0.321	0.340	0.354	0.467	0.498	0.512	0.528	0.445	0.471	0.488	0.501

Table 10: Data ablation at 5K, 10K, 15K and 20K steps. The target model is Mistral-7B. **Bold** and underscore denote top-1/2 accuracy.

		SQuAD				DROP				ROPES				Avg.			
Data	**Method**	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K	5K	10K	15K	20K
FW-100K	Native	0.425	0.407	0.361	0.350	0.175	0.159	0.156	0.169	0.405	0.376	0.382	0.365	0.335	0.314	0.300	0.295
	Gemma2-2B→	**0.527**	0.537	**0.539**	0.526	**0.221**	**0.267**	**0.242**	0.230	**0.470**	**0.490**	**0.498**	**0.497**	**0.406**	**0.431**	**0.426**	**0.418**
	Qwen3-4B→	0.524	**0.548**	0.538	**0.533**	0.219	0.247	0.230	**0.233**	0.448	0.476	0.452	0.466	0.397	0.424	0.407	0.411
FW-100K + QA	Native	0.438	0.427	0.471	0.464	0.282	0.320	0.328	0.317	0.409	0.427	0.425	0.434	0.377	0.391	0.408	0.405
	Gemma2-2B→	0.553	0.592	0.578	0.558	**0.342**	**0.362**	**0.376**	**0.372**	0.501	**0.546**	**0.567**	**0.571**	**0.466**	**0.500**	**0.507**	**0.500**
	Qwen3-4B→	**0.564**	**0.598**	**0.584**	**0.588**	0.321	0.338	0.329	0.310	**0.511**	0.516	0.548	0.564	0.465	0.484	0.487	0.487
FW-250K	Native	0.420	0.444	0.490	0.543	0.170	0.156	0.195	0.196	0.398	0.424	0.436	0.464	0.329	0.341	0.373	0.401
	Gemma2-2B→	**0.575**	**0.607**	**0.616**	**0.620**	0.228	0.259	**0.274**	**0.293**	**0.479**	**0.494**	**0.497**	**0.504**	**0.427**	**0.453**	**0.463**	**0.472**
	Qwen3-4B→	0.522	0.585	0.580	0.578	**0.229**	**0.262**	0.252	0.251	0.471	0.490	0.484	0.492	0.407	0.446	0.439	0.440
FW-250K + QA	Native	0.445	0.478	0.533	0.568	0.250	0.283	0.285	0.310	0.426	0.428	0.444	0.449	0.374	0.397	0.421	0.442
	Gemma2-2B→	**0.591**	**0.643**	**0.663**	**0.649**	**0.298**	**0.344**	**0.348**	**0.364**	**0.487**	**0.543**	**0.597**	**0.598**	**0.459**	**0.510**	**0.536**	**0.537**
	Qwen3-4B→	0.572	0.595	0.612	0.620	0.295	0.321	0.340	0.354	0.467	0.498	0.512	0.528	0.445	0.471	0.488	0.501