Scaling Laws for HyperNetwork-Based Knowledge Injection in Large Language Models

Nischay Dhankhar
Nace AI
nischay@nace.ai

Dos Baha
Nace AI
dos@nace.ai

Abulhair Saparov
Nace AI
Purdue University
asaparov@purdue.edu

Abstract

Injecting factual knowledge into large language models (LLMs) reliably and at scale remains an open challenge.

Hypernetworks provide a promising solution to large-scale knowledge injection. Although hypernetworks are typically applied for *test-time adaptation*, we explore their use in *train-time knowledge injection*, where, given a large corpus of facts, we train a hypernetwork to generate a *fixed* LoRA adapter that, when inserted into the target model, enable the model to answer questions about those facts.

In this work, we investigate whether hypernetworks can be used to perform train-time knowledge injection and how this ability varies with scale. The scaling behavior of hypernetworks themselves remains largely unstudied.

Our design decouples the injection capacity of the hypernetwork from the general capability of the target model, enabling, for the first time, a rigorous and controlled study of **scaling laws for hypernetwork architectures** in the context of knowledge injection. We characterize how loss, reasoning accuracy, and out-of-distribution (OOD) generalization vary as functions of hypernetwork depth, width, and target network size. To this end, we construct a large-scale dataset, called `MegaWikiQA`, containing tens of millions of multi-hop question-answer examples across 39 knowledge domains constructed from examples in `Wikidata5M`.

Our results reveal: *(i)* hypernetwork-based injection exhibits broadly predictive power law scaling along all architecture axes; and *(ii)* hypernetworks are capable of reliable OOD generalization to unseen entities and relations at increasing scales, suggesting that hypernetwork adaptation provides a promising alternative to other train-time adaptation methods such as LoRA finetuning and full fine-tuning, exhibiting steeper scaling exponents in all OOD evaluations. Together, these results establish hypernetworks as a principled and scalable substrate for train-time knowledge injection, and provide the first empirically grounded **scaling laws** to guide hypernetwork design for factual reasoning in large language models.[1]

1 Introduction

The ability to inject and generalize factual knowledge in large language models (LLMs) is fundamental to their deployment in knowledge-intensive settings, where models must reliably condition their outputs on specific facts from a large fixed text corpus, particularly involving knowledge that may not be well-represented in pretraining. For example, in medical applications, API-based LLM solutions are inappropriate due to the sensitivity of patient data, and adaptation of a local LLM is preferable [Kothari and Gupta, 2025]. In specific domains, such as law and finance, questions may depend on

[1] Code and data are publicly available at `https://huggingface.co/collections/nace-ai/hypernetwork-datasets`.

domain-specific knowledge that is missing from the LLM [Ling et al., 2026], such as new financial regulations [Yang et al., 2023], enterprise internal policies [Liu et al., 2025a], etc.

A natural approach to address this challenge is fine-tuning, but full fine-tuning can be prohibitively expensive. Parameter-efficient fine-tuning (PEFT) methods such as LoRA [Hu et al., 2022] can help to reduce the cost of fine-tuning, but both fine-tuning and PEFT methods are prone to catastrophic forgetting [Kotha et al., 2024, Li et al., 2024] and can struggle with out-of-distribution (OOD) generalization [Hajipour et al., 2024].

Due to these limitations, we explore an alternative and promising approach for knowledge injection that does not share the same limitations. In this work, we study the injection of knowledge via hypernetworks [Ha et al., 2017] a paradigm in which a secondary network, conditioned on a set of injected facts, generates contextual weight adaptations (e.g., LoRA adapters) for the target model. This approach avoids modifying the base parameters of the target model, which can help to mitigate catastrophic forgetting and facilitate OOD generalization. Hypernetworks can incorporate additional information from their input which can lead to higher-quality representations and more compute-efficient learning of high-quality adapters for the target model. See Figure 1 in Section 4.2 for an overview.

We emphasize that our work focuses specifically on *train-time knowledge internalization*: both the hypernetwork and all finetuning baselines we compare against are trained to compress a fixed fact corpus Ω into target model weights, and are evaluated on queries without any inference-time access to Ω. This isolates the effect of the training-time adaptation mechanism, which is the object of study in our scaling laws. Our novel approach is in stark contrast to the typical usage of hypernetworks for *test-time adaptation*, which is more akin to *knowledge editing* [Mitchell et al., 2022, Meng et al., 2022, 2023, Cohen et al., 2024], which we not consider in this paper.

Despite their theoretical appeal, there exists no systematic characterization of the scaling behavior of hypernetwork-based knowledge injection. The central contribution of this paper is the first empirical and analytical study of **scaling laws for hypernetwork-based knowledge injection**. Concretely, we investigate how performance varies as a function of: (i) the width of the hypernetwork; (ii) the depth of the hypernetwork; (iii) the size of the target language model; and (iv) the quantity of injected facts. In our experiments, we consider hypernetworks with sizes ranging from 167M to 2.8B parameters. In order to perform such scaling experiments, we require a large labeled dataset of question-answer pairs. To this end, we construct such a dataset, called `MegaWikiQA`, by extracting subject-relation-object triplets from `Wikidata5M` [Wang et al., 2021b], enabling controlled scaling over millions of facts with rigorous evaluation of the model's multi-hop reasoning ability.

Our results reveal that hypernetwork-based knowledge injection exhibits smooth, predictable scaling behavior in contrast to the sharp failure modes documented for parametric editing. We further show that hypernetwork-generated adaptations generalize across unseen entities and relations, support multi-hop compositional queries, and exhibit steeper OOD generalization scaling than both LoRA finetuning and full finetuning as the target model size increases, while achieving comparable performance on in-distribution validation.

We make the following core contributions in this work:

1. We provide the first systematic study of scaling laws for hypernetwork-based knowledge injection along multiple axes: (1) hypernetwork architecture parameters including transformer depth and width; (2) the size of the target language model; and (3) the number of injected facts (Section 5).
2. We characterize the regimes where hypernetwork scaling yields diminishing returns. We find that scaling the depth and width yields comparable performance improvement, and that scaling the target model offers substantially better improvement than scaling the hypernetwork (Section 5.3 and Table 1).
3. We provide direct scaling law comparisons between hypernetwork adaptation, LoRA finetuning, and full finetuning across target model size. While finetuning methods scale slightly better on in-distribution validation, the hypernetwork exhibits substantially steeper OOD generalization scaling across all three OOD splits, and this advantage grows with target model scale (Sections 5.5 and 5.6).
4. We release a reproducible large-scale benchmark, `MegaWikiQA`, built on `Wikidata5M` with deterministic multi-hop question generation and explicit OOD evaluation sets, suitable for knowledge injection research, post-training, and evaluation of factual reasoning in LLMs (Section 3).

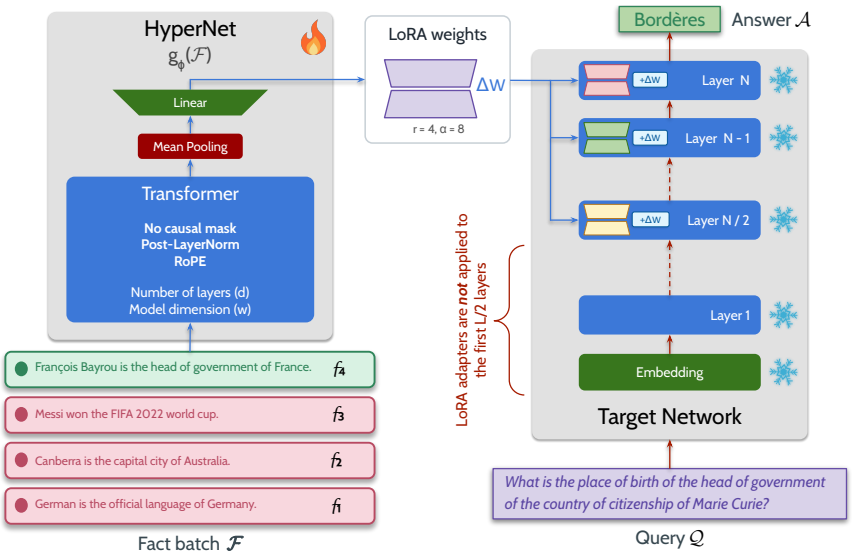

FIGURE 1: Diagram of the proposed hypernetwork-based approach for knowledge injection, showcasing a single example. Only hypernetwork parameters are learned whereas target model parameters are frozen.

2 Related Work

Fine-tuning and PEFT for Knowledge Injection. The term *knowledge injection*[2] was introduced in early work on injecting structured knowledge into pretrained language models, including K-BERT [Liu et al., 2020], K-Adapter [Wang et al., 2021a], and ERNIE [Zhang et al., 2019], and has since been widely adopted to describe fine-tuning-based approaches for incorporating knowledge from a corpus into models [Ovadia et al., 2024]. Full fine-tuning is the most straightforward approach to knowledge injection, but is prohibitively expensive at scale and prone to catastrophic forgetting [Kotha et al., 2024, Li et al., 2024]. Parameter-efficient fine-tuning (PEFT) methods such as LoRA [Hu et al., 2022] reduce adaptation cost but do not eliminate forgetting [Li et al., 2024], and struggle with out-of-distribution generalization to unseen entity and relation combinations [Hajipour et al., 2024]. More fundamentally, injecting new factual knowledge through supervised fine-tuning is inherently difficult: new facts are learned significantly slower than facts that are consistent with pretraining, and as they are eventually learned, they increase the model's tendency to hallucinate [Gekhman et al., 2024].

Hypernetworks for Knowledge Injection. Hypernetworks [Ha et al., 2017] are networks whose outputs are the weights of another network, enabling dynamic and input-conditioned parameterization. Beyond knowledge editing, hypernetworks have been applied to parameter-efficient adaptation across tasks and languages: notably, Hyper-X [Üstün et al., 2022] generates adapter module weights conditioned on joint task and language embeddings, enabling zero-shot transfer to unseen task-language combinations without training separate adapters per configuration. Two prior studies are most closely related to ours in the context of knowledge editing. MEND [Mitchell et al., 2022] trains a hypernetwork to transform the gradient of a language modeling loss on an injected fact, applying the transformed gradient to update the base model. Building on this, PropMEND [Liu et al., 2025b] extends MEND by modifying the meta-training objective to encourage multi-hop propagation of injected knowledge, showing improved performance on non-verbatim propagation questions. Critically, both MEND and PropMEND operate in gradient space: the hypernetwork transforms a gradient signal, after which the resulting update is applied to the base model parameters. These methods are prone to model instability when the number of injected facts grows beyond ~ 1000 [Gupta et al., 2024]. Our approach departs from this paradigm by using the hypernetwork to

[2]Following prior usage of the term [Ovadia et al., 2024, Wang et al., 2021a, Liu et al., 2020, Zhang et al., 2019], we use *knowledge injection* to refer to the general task of teaching a pre-trained model a body of knowledge from a text corpus, regardless of whether that knowledge is entirely novel to the model. In particular, the model may already have partial exposure to the injected facts through pretraining; the goal is to reliably condition the model's outputs on the specified corpus.

directly predict LoRA-style weight adaptations [Hu et al., 2022] conditioned on the injected facts at inference time, with no modification of base parameters, which helps to avoid instability.

Scaling Laws for Hypernetworks. The study of neural scaling laws gained prominence with Kaplan et al. [2020], who established power-law relationships between model size, dataset size, training compute, and validation loss for autoregressive language models. Subsequent work extended scaling analyses to specific capabilities including in-context learning [Brown et al., 2020], instruction following [Ouyang et al., 2022], and reasoning Wei et al. [2022]. More recently, Béthune et al. [2025] extended scaling law analyses to the finetuning regime, deriving power-law relationships that quantify catastrophic forgetting as a function of model size, finetuning data volume, and the fraction of pretraining data injected into the finetuning mixture. The closest prior work to ours is Gu and Yeung-Levy [2025], who show that hypernetwork performance improves as the backbone foundation model is scaled up in the context of implicit neural representation tasks. However, their scaling analysis is restricted to varying the target network size, focuses on image and audio reconstruction rather than knowledge injection, and does not explore scaling the hypernetwork architecture itself. The scaling behavior of hypernetworks, which must learn to generate weights for a separate frozen target network conditioned on structured inputs, introduces qualitatively different questions: how does depth versus width of the hypernetwork affect the quality of generated adaptations, and how does hypernetwork capacity interact with target model size? To our knowledge, no prior work has studied these questions empirically. Our work fills this gap by providing the first empirical scaling laws for hypernetwork architectures in the context of knowledge injection.

Knowledge Injection Datasets. Existing benchmarks for knowledge injection and editing vary significantly in scale and construction methodology. `ZsRE` [Levy et al., 2017] contains tens of thousands of examples and predates modern LLMs, limiting its suitability for large-scale evaluation. `UniEdit` [Chen et al., 2025] and `WikiBigEdit` [Thede et al., 2025] represent more recent efforts at 311K and 500K examples respectively, with broader domain coverage. However, both are designed primarily for parametric inference-time knowledge editing rather than knowledge injection. In inference-time knowledge editing, the task is to answer questions where each question is conditioned on new (possibly counterfactual) facts. Furthermore, past benchmarks rely on LLM-generated questions which may introduce noise and stylistic bias. More broadly, none of these datasets are designed to support scaling experiments across multiple orders of magnitude, where larger models may become increasingly sensitive to confounding effects such as noise and bias. We construct `MegaWikiQA` to fill this gap, with fully deterministic grammar-based question generation, over 10 million examples per number of hops, and explicit OOD domain splits for controlled evaluation.

3 Dataset Construction: `MegaWikiQA`

A rigorous study of scaling laws for knowledge injection requires a dataset that is large-scale, structurally rich, deterministic in its ground-truth labels, and capable of supporting both single hop and multi-hop compositional evaluation. Support for multi-hop evaluation is critical because, in the knowledge injection setting, the model may be required to perform *multi-hop and compositional reasoning over multiple injected facts* in order to answer a question, as opposed to recall individual memorized facts. Ideally, the dataset should contain several orders of magnitude in its number of examples to enable rigorous scaling experiments. To that end, we construct our dataset, `MegaWikiQA`, entirely from `Wikidata5M` [Wang et al., 2021b], a large scale knowledge graph containing approximately 4.6 million entities, 822 relations, and over 22 million triplets of the form (s, r, o), where s is a subject entity, r is a relation, and o is the object entity (e.g., $(\texttt{germany}, \texttt{capital}, \texttt{berlin})$ is a triplet representing the fact "The capital of Germany is Berlin"). `Wikidata5M` satisfies all of the above requirements: its scale provides ample coverage for training and evaluation across 39 knowledge domains, its graph structure supports deterministic construction of multi-hop reasoning examples on the scale of tens of millions, and its triplet format yields unambiguous ground-truth answers for all question types we consider.

3.1 Question Answer Generation

Questions are generated by first performing a random walk in the `Wikidata5M` knowledge graph. We begin the walk with a seed triplet (s_1, r_1, o_1), where s_1 is the subject entity, o_1 is the object entity,

and r_1 is the relation. We then consider all neighboring edges—i.e., where the object of the first fact o_1 is the subject of the next fact—and select an edge uniformly at random. We repeat this process until we have a k-hop walk $(s_1, r_1, o_1, r_2, o_2, \ldots, r_k, o_k)$. We generate questions for multiple hop counts $k \in \{1, 2, 3, 4\}$. This process is repeated to generate multiple k-hop walks for each seed triplet, which we repeat again by considering every triplet in `Wikidata5M` as the seed triplet.

The k-hop walk is converted into a natural language question-answer pair using a **grammar-based approach**. Let f be a (recursive) function that converts a given k-hop walk into a noun phrase:

$$f(s_1, r_1, o_1, r_2, o_2, \ldots, r_k, o_k) := \begin{cases} \text{"the country of citizenship of"} f(s_1, r_1, \ldots, r_{k-1}, o_{k-1}) & \text{if } r_k = \texttt{citizen\_of\_country}, \\ \text{"the head of government of"} f(s_1, r_1, \ldots, r_{k-1}, o_{k-1}) & \text{if } r_k = \texttt{head\_of\_govt}, \\ \text{"the place of birth of"} f(s_1, r_1, \ldots, r_{k-1}, o_{k-1}) & \text{if } r_k = \texttt{place\_of\_birth}, \\ \vdots & \vdots \end{cases}$$

In the base case, where $k = 0$, the input sequence consists of a single entity s_1, which we verbalize: $f(s_1) =$ "Marie Curie" if $s_1 =$ `marie_curie`, etc.

We apply this function to convert the full k-hop walk into a noun phrase (e.g., "the country of citizenship of Marie Curie"). Next, we convert it into a question ("What is the country of citizenship of Marie Curie?"). To perform this last conversion step, we manually curated and evaluated over 400 candidate question templates using a representative subset of triplets sampled from the dataset and constructed a deterministic mapping from each of the 822 `Wikidata` relations to its most appropriate template. Question templates have forms such as *"What is* $f(s_1, r_1, o_1, \ldots, r_k, o_k)$*?"* and *"Which is* $f(s_1, r_1, o_1, \ldots, r_k, o_k)$*?"*, with the root selected based on the semantic type of the last object entity (e.g., *"Who is"* for objects with person type, *"What is"* otherwise). The answer to the question is simply given by verbalizing the last object entity o_k (e.g., $f(o_k) =$ "France" if $o_k =$ `france`, etc).

To ensure the answers to our questions are unambiguious, we restrict our dataset to relations that are one-to-one or many-to-one, excluding one-to-many and many-to-many relations whose answers are inherently non-deterministic. Although scaling experiments can be performed with ambiguous question-answer pairs, this restriction reduces label uncertainty and enables more reliable and interpretable evaluation using metrics such as accuracy.

Illustrative example. Suppose we perform a random walk and obtain the sequence *Marie Curie* $\rightarrow$ *country of citizenship* $\rightarrow$ *France* $\rightarrow$ *head of government* $\rightarrow$ *François Bayrou* $\rightarrow$ *place of birth* $\rightarrow$ *Bordères*. Our procedure would then convert this into the question: *"What is the place of birth of the head of government of the country of citizenship of Marie Curie?"* with answer *Bordères*. This approach is fully deterministic, requires no neural generation at the question generation stage, and produces grammatically consistent questions across all hop counts.

After performing this generation procedure over the full `Wikidata5M` dataset, we generated a multi-hop dataset containing approximately **10 million examples per hop count** up to 4 hops. We classify each example into one of 39 domains using a two-stage classification pipeline described in Section B. We perform further filtering to remove examples on which the domain classification was uncertain, and to balance the number of examples across all hop counts. This resulted in a final dataset containing 1.25 million training samples stratified across knowledge domains and reasoning complexity (i.e., number of hops).

3.2 Fact Injection Protocol

During training, our goal is to inject a large set of facts Ω. Each training example consists of an injected fact set $\mathcal{F} \subset \Omega$, a natural language query q, and a ground truth answer a. The injected fact set always contains one *relevant fact*: from the k-hop sequence of facts that were used to generate the question-answer pair, the relevant fact is randomly selected from this sequence. The remaining $N - 1$ facts in $\mathcal{F}$ are **negative facts** sampled uniformly at random from Ω.

3.3 Evaluation Protocol and OOD Splits

We construct three disjoint evaluation sets to support comprehensive assessment of knowledge injection and generalization.

In-distribution (ID) evaluation. 10,000 randomly sampled examples stratified by domain and number of hops, constructed to be disjoint from the training set at the triplet level, i.e., no triplet appearing in the evaluation set appears in the training set, and vice versa.

Out-of-distribution (OOD) evaluation. 10,000 examples randomly drawn exclusively from three held-out domains: *philosophy*, *linguistics*, and *civil engineering* as seen in Figure 8. These domains were selected due to their internal diversity and the fact that they were the most distinct as compared to the other domains. OOD examples further include **rephrased questions** generated by prompting GPT 4.1 [OpenAI, 2023] to paraphrase the example, with the aim to reduce potential overfitting to the highly-regular language produced by our grammar-based generation procedure, providing a stricter test of generalization by controlling for confounding due to transfer learning of surface-level linguistic features. Finally, we include a **multiple-choice question (MCQ)** evaluation split, constructed from the same OOD examples where each question includes four answer choices: the correct entity and three distractor entities sampled uniformly at random from Ω. This format tests whether the hypernetwork-adapted model can identify the correct answer under a fixed candidate set, providing a complementary evaluation signal that is robust to variations in generation style. Full dataset statistics, including split sizes and hop distributions, are summarized in Table 6 in the Appendix.

4 Method

4.1 Problem Formulation

We study the problem of knowledge injection of a large corpus of facts Ω into a frozen language model $\mathcal{M}_\theta$ with fixed parameters θ. Each training example consists of a natural language query q and a set of N injected facts $\mathcal{F} = \{f_1, \ldots, f_N\} \subset \Omega$, where each fact f_i is a natural language verbalization of a triplet $(s, r, o) \in \Omega$. Exactly one fact in $\mathcal{F}$ is *relevant* to answering q, while the remaining $N - 1$ facts are *negatives* sampled uniformly at random from Ω, encouraging the hypernetwork to identify and utilize the relevant fact and improving generalization in the presence of irrelevant information. The objective is to produce an answer a consistent with the relevant fact in $\mathcal{F}$, without modifying the base model parameters θ at any point. Only the hypernetwork parameters ϕ are updated during training; the target model $\mathcal{M}_\theta$ remains fully frozen throughout both training and inference, which helps to mitigate the high cost of fine-tuning and may help to facilitate OOD generalization.

This formulation captures the core challenge of knowledge injection: the hypernetwork must learn to identify and encode the relevant fact from a noisy input context, and translate it into weight adaptations that steer the frozen target model toward the correct answer. Unless otherwise stated, we fix $N = 4$ facts per example across all experiments, and study the effect of varying N in dedicated experiments where we scale the number of facts (Section 5.4).

4.2 Hypernetwork Architecture

We introduce a transformer-based hypernetwork g_ϕ with learnable parameters ϕ, initialized entirely from random weights (see Figure 1 for an architectural overview). A key design principle in our study is that the hypernetwork is never initialized from pretrained weights, as doing so would conflate the effect of pretraining with that of architectural capacity, undermining the validity and controlled nature of our exploration on the effect of scale on hypernetwork performance. The hypernetwork maps an input fact set $\mathcal{F}$ to a collection of LoRA-style weight adaptations [Hu et al., 2022], which are applied to the frozen target model in the forward pass. We use LoRA rank $r = 4$ and scaling factor $\alpha = 8$ across all experiments. Full details of the encoder architecture, input encoding, LoRA weight generation, and target layer selection are provided in Appendix D.

Architecture Scaling Axes. We systematically scale the transformer hypernetwork along three independent axes to characterize the resulting scaling laws. For **depth scaling** (Section 5.2), we vary the number of transformer layers $L_{\text{HN}} \in \{1, 2, 4, 8, 16\}$ while holding width fixed. For **width scaling** (Section 5.1), we vary the hidden dimension $d_{\text{model}} \in \{32, 64, 128, 256, 512, 1024\}$ while holding depth fixed. For **fact count scaling** (Section 5.4), we vary the number of injected facts N per example while holding the hypernetwork architecture fixed, studying how the quantity of available evidence per example affects injection performance. In all cases, the hypernetwork is initialized from

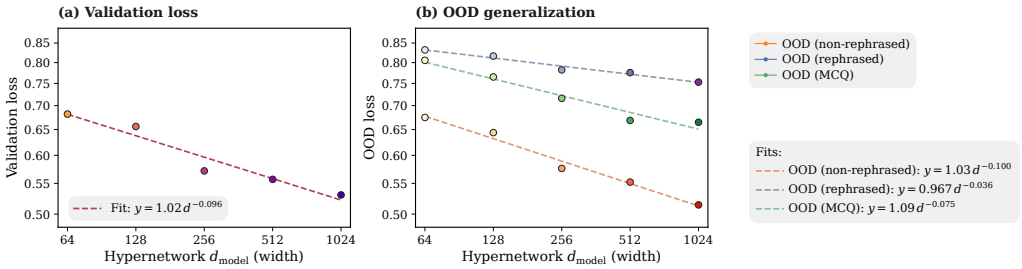

FIGURE 2: Hypernetwork width scaling results, showing final epoch loss vs. d_{model} on a log-log scale. **(a)** Validation loss follows a power-law fit $\mathcal{L}_{\text{val}} = 1.02 \cdot d^{-0.096}$. **(b)** All three OOD metrics improve with width, with fitted exponents -0.100 (OOD non-rephrased), -0.036 (OOD rephrased), and -0.075 (OOD MCQ).

random weights. In each experiment, where we vary one of the above scaling variables, all other variables are held constant.

5 Experiments

All scaling law experiments use `MegaWikiQA` as the knowledge source, with training and evaluation splits constructed as in Section 3. Unless otherwise stated, hypernetwork width scaling, depth scaling, and fact count scaling experiments use `Qwen2.5-1.5B-Instruct` as the frozen target model. We choose the `Qwen2.5` family since it offers a wide range of model sizes within a consistent architecture, enabling controlled scaling comparisons; newer families such as `Qwen3` have fewer variants and less consistent training across models, making them less suitable for scaling analysis.

We evaluate all models on four metrics: in-distribution validation loss (ID validation), OOD loss on held-out domains (OOD non-rephrased), OOD loss on rephrased questions on held-out domains (OOD rephrased), and OOD loss on multiple-choice questions on held-out domains (OOD MCQ), where each OOD split tests a qualitatively distinct form of generalization as described in Section 3. Power-law fits of the form $\mathcal{L} = ax^b$ are estimated via least-squares regression in log-log space using the final epoch loss, where x is the scaling variable of interest (e.g., hypernetwork width, etc).

5.1 Hypernetwork Width Scaling

We vary the hypernetwork hidden dimension $d_{\text{model}} \in \{64, 128, 256, 512, 1024\}$, holding the number of transformer layers fixed and all other hyperparameters constant. Figure 2 shows the final epoch loss vs d_{model} on a log-log scale. Loss trajectories during training are provided in Appendix F (Figure 11).

All four evaluation metrics follow smooth power-law relationships vs hypernetwork width. The validation loss exponent (-0.096) is comparable to the OOD non-rephrased exponent (-0.100), indicating that width scaling benefits both in-distribution and OOD recall roughly equally, if the questions have similar phrasing. However, the OOD rephrased exponent (-0.036) is substantially flatter, suggesting that width alone does not substantially improve robustness to linguistic surface variation. The OOD MCQ exponent (-0.075) falls between the two, reflecting that multiple-choice reformulation is a moderately harder generalization target as compared to open-ended generation.

5.2 Hypernetwork Depth Scaling

We scale the number of transformer layers in the hypernetwork across $L_{\text{HN}} \in \{1, 2, 4, 8, 16\}$, holding $d_{\text{model}} = 512$ and all other hyperparameters fixed. Figure 3 shows the final epoch loss against L_{HN} on a log-log scale for all four evaluation metrics. Loss trajectories during training are shown in Figure 12 in Appendix F.

All four evaluation metrics follow smooth power-law relationships with hypernetwork depth. The fitted exponents are -0.088 for validation loss, -0.096 for OOD non-rephrased, -0.042 for OOD rephrased, and -0.063 for OOD MCQ. As in the case with width scaling, the exponent on in-distribution and OOD non-rephrased metrics is steeper as compared to OOD rephrased, suggesting that increasing hypernetwork depth does not exhibit better surface-level linguistic generalization

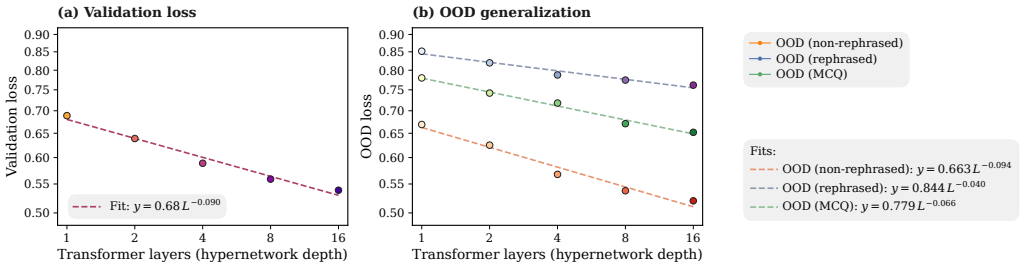

FIGURE 3: Hypernetwork depth scaling results, showing final epoch loss vs. number of transformer layers L_{HN} on a log-log scale. **(a)** Validation loss follows a power-law fit $\mathcal{L}_{\text{val}} = 0.677 \cdot L^{-0.088}$. **(b)** All three OOD metrics improve with depth, with fitted exponents -0.096 (OOD non-rephrased), -0.042 (OOD rephrased), and -0.063 (OOD MCQ).

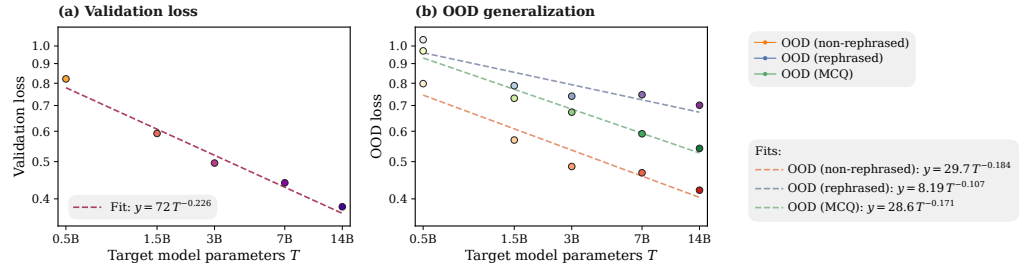

FIGURE 4: Target model scaling results, showing final epoch loss vs. target model parameter count T on a log-log scale. **(a)** Validation loss follows a power-law fit $\mathcal{L}_{\text{val}} = 72 \cdot T^{-0.226}$. **(b)** All three OOD metrics improve with target model size, with fitted exponents -0.184 (OOD non-rephrased), -0.107 (OOD rephrased), and -0.171 (OOD MCQ).

as compared to increasing width. Comparing the width exponents to the depth exponents reported in Section 5.2, we observe broadly similar scaling rates (-0.096 vs -0.088 for validation loss), indicating that hypernetwork depth and width contribute comparably to overall knowledge injection performance within the ranges studied.

5.3 Target Model Scaling

Figure 4 shows the final epoch loss against target model size on a log-log scale. Loss trajectories are provided in Appendix F (Figure 14).

Target model scaling yields the steepest power-law exponents across all axes studied: -0.226 for validation loss and -0.184 for OOD non-rephrased loss, compared to -0.088 and -0.096 for depth and width scaling respectively. This indicates that scaling the target model yields substantially larger absolute gains per unit compute as compared to scaling the hypernetwork architecture alone, and suggests a practical guideline: when a fixed compute budget must be allocated between hypernetwork capacity and target model size, the latter offers a more favorable return.

The 0.5B target model lies noticeably above the power-law fit on the right-side of the figure, suggesting the existence of a minimum target model capacity threshold below which hypernetwork-generated LoRA adaptations cannot be effectively leveraged.

5.4 Fact Count Scaling

We vary the number of training facts $N \in \{2, 4, 8, 16, 32, 52\}$ injected per example, holding the hypernetwork architecture and target model fixed.[3] Figure 5 shows the final epoch loss against N on a log-log scale. Loss trajectories during training are provided in Appendix F (Figure 13).

[3]The maximum number of facts that we consider is 52 since the hypernetwork context is limited to 512 tokens.

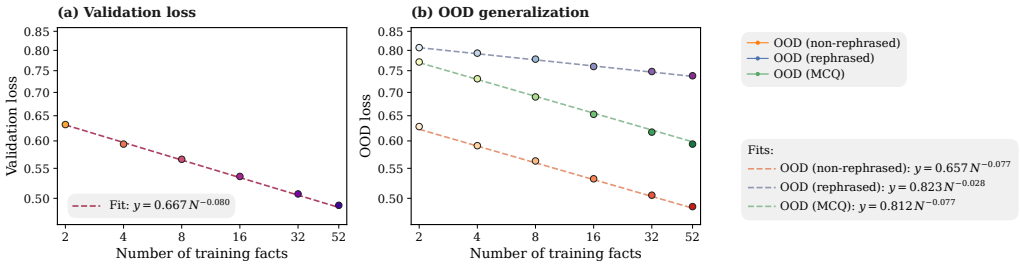

FIGURE 5: Fact count scaling results, showing final epoch loss vs. num. of injected facts N per example on a log-log scale. **(a)** Validation loss follows a power-law $\mathcal{L}_{\text{val}} = 0.667 \cdot N^{-0.080}$. **(b)** OOD metrics improve with fact count. Fitted exponents: -0.077 (OOD non-rephrased), -0.028 (OOD rephrased), -0.077 (OOD MCQ).

Performance improves as the number of injected facts per example grows, following power-law trends across all metrics. The validation loss exponent (-0.080) and OOD non-rephrased exponent (-0.077) are modest but stable, indicating that providing more factual context reliably aids the hypernetwork's ability to identify and encode the relevant fact. The OOD rephrased exponent (-0.028) is again the flattest, consistent with the pattern observed in depth and width scaling. The OOD MCQ exponent (-0.077) matches the OOD non-rephrased exponent closely, suggesting that multiple-choice and free-form evaluation are comparably affected by the quantity of available evidence.

5.5 LoRA Scaling

To complete the head-to-head comparison, we also experiment with scaling the target model under LoRA finetuning with fixed adapter configuration $r = 16, \alpha = 32$, across the same five `Qwen2.5` sizes used in the hypernetwork target scaling experiments. Figure 6 shows the final epoch loss against target model size on a log-log scale.

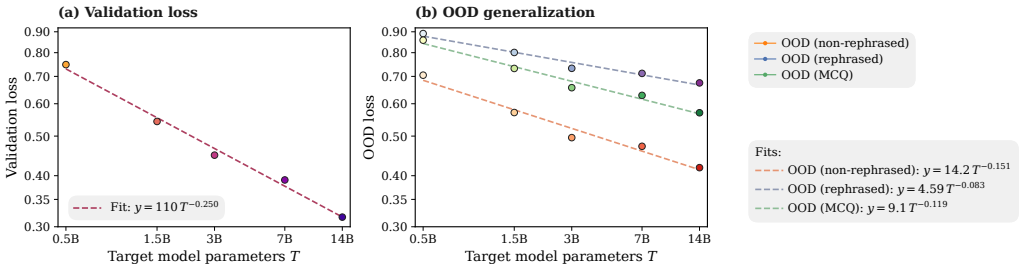

FIGURE 6: LoRA target model scaling results ($r = 16, \alpha = 32$), showing final epoch loss vs. target model parameter count T on a log-log scale. **(a)** Validation loss follows a power-law fit $\mathcal{L}_{\text{val}} = 110 \cdot T^{-0.250}$. **(b)** OOD metrics improve with target model size, with fitted exponents -0.151 (OOD non-rephrased), -0.083 (OOD rephrased), and -0.119 (OOD MCQ).

Comparing LoRA target scaling to hypernetwork target scaling reveals a consistent pattern: LoRA achieves a marginally steeper ID validation loss exponent (-0.250 vs. -0.226 for the hypernetwork), but its OOD scaling exponents are substantially flatter across all three OOD splits (-0.151 vs. -0.184 on non-rephrased, -0.083 vs. -0.107 on rephrased, and -0.119 vs. -0.171 on MCQ). In other words, increasing target model size leads to a larger improvement in LoRA fitting the training distribution as compared to the hypernetwork fitting the training distribution, but it leads to a substantially larger improvement in the hypernetwork's OOD generalization as compared to LoRA. This gap is consistent with the OOD advantage observed in the comparison at fixed target model size (Section 5.8) and demonstrates that the OOD advantage of hypernetwork-based injection persists and, in fact, widens as the target model is scaled.

5.6 Full Finetuning Scaling

For completeness, we also experiment with scaling the target model under full finetuning, where all target model parameters are updated during training. Due to compute constraints, we cover four

`Qwen2.5` sizes for full finetuning: 0.5B, 1.5B, 3B, and 7B[4]. Figure 7 shows the final epoch loss against target model size on a log-log scale.

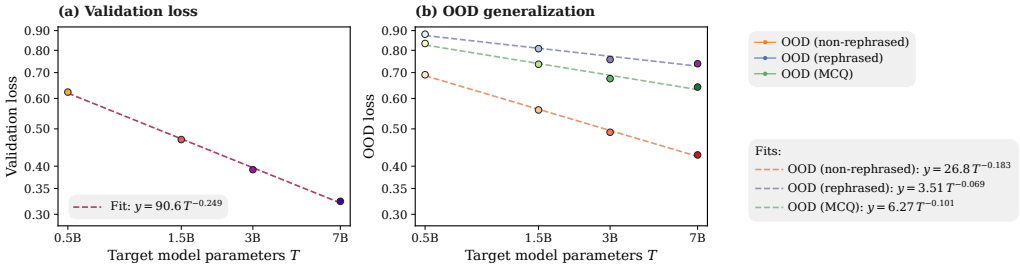

FIGURE 7: Full finetuning target model scaling results, showing final epoch loss vs. target model parameter count T on a log-log scale. **(a)** Validation loss follows a power-law fit $\mathcal{L}_{\text{val}} = 90.6 \cdot T^{-0.249}$. **(b)** OOD metrics improve with target model size, with fitted exponents -0.183 (OOD non-rephrased), -0.069 (OOD rephrased), and -0.101 (OOD MCQ). The 14B point was not evaluated due to compute constraints.

Full finetuning target scaling exhibits a pattern qualitatively similar to LoRA finetuning: a steep ID validation loss exponent (-0.249, nearly matching LoRA's -0.250) but flatter OOD loss exponents (-0.183, -0.069, -0.101) than the hypernetwork (-0.184, -0.107, -0.171). While full finetuning approximately matches the hypernetwork on OOD non-rephrased scaling, it lags substantially on OOD rephrased and OOD MCQ, the two metrics that test robustness to linguistic and format variation. If the observed power-law trends hold, the OOD advantage of the hypernetwork over full finetuning would be expected to widen further at 14B, consistent with the same pattern observed in the LoRA scaling comparison (Section 5.5).

5.7 Summary of Scaling Exponents

Table 1 summarizes the power-law exponents across all four scaling axes and all four evaluation metrics. Target model scaling consistently yields the steepest exponents, followed by depth and width scaling at comparable rates, with fact count scaling yielding the most modest improvements. Across all axes, the OOD rephrased metric exhibits the flattest scaling, suggesting that robustness to linguistic variation is not easily addressed by scaling any single architectural or data dimension.

TABLE 1: Summary of power-law scaling exponents across all target-scaling axes and evaluation metrics. More negative values indicate steeper (more favorable) scaling. HN denotes hypernetwork, FT denotes finetuning. LoRA rank scaling is reported separately in Appendix E since it is better fit by a power-law with an additive constant.

Scaling Axis	ID validation	OOD non-rephrased	OOD rephrased	OOD MCQ
HN width (d_{model})	-0.096	-0.100	-0.036	-0.075
HN depth (L_{HN})	-0.088	-0.096	-0.042	-0.063
HN target model size (T)	-0.226	$\mathbf{-0.184}$	$\mathbf{-0.107}$	$\mathbf{-0.171}$
Fact count (N)	-0.080	-0.077	-0.028	-0.077
LoRA FT target model size (T)	$\mathbf{-0.250}$	-0.151	-0.083	-0.119
Full FT target model size (T)	-0.249	-0.183	-0.069	-0.101

5.8 Comparison of Scaling Laws Across Adaptation Methods

Our target model scaling experiments across the three train-time adaptation methods (hypernetwork, LoRA finetuning, and full finetuning) enable a direct comparison of their scaling behavior as target model capacity grows. Table 1 reports the fitted power-law exponents for all three methods across all four evaluation metrics.

Two consistent patterns emerge. First, on in-distribution validation, finetuning methods scale slightly better than the hypernetwork: LoRA finetuning (-0.250) and full finetuning (-0.249) achieve nearly identical and steeper exponents than the hypernetwork (-0.226). This reflects the ability of direct parameter updates, whether full or low-rank, to fit the training distribution more aggressively as

[4]Full finetuning for the 14B configuration was infeasible under our budget.

target capacity grows. Second, on all three OOD splits, the hypernetwork exhibits the steepest scaling among the three methods, most notably on OOD rephrased (-0.107 vs. -0.083 for LoRA and -0.069 for full FT) and OOD MCQ (-0.171 vs. -0.119 and -0.101). On OOD non-rephrased, the hypernetwork (-0.184) narrowly leads full FT (-0.183) and LoRA (-0.151).

The gap between hypernetwork and finetuning OOD scaling grows monotonically with target model size, meaning that the OOD advantage of hypernetwork-based adaptation is not a fixed offset but widens as target models become larger. This is arguably the most important finding of our scaling law comparison: if the goal is train-time knowledge injection that generalizes robustly outside the training distribution, the hypernetwork paradigm becomes increasingly favorable at scale, precisely the regime where deployment matters most.

6 Conclusion and Future Work

We introduce `MegaWikiQA` and conduct the first systematic study of scaling laws for hypernetwork-based knowledge injection, characterizing power-law improvements across hypernetwork width, depth, target model size, and number of injected facts. Our results show that target model scaling yields the strongest gains, that hypernetworks generalize more effectively to OOD settings than other finetuning methods under matched capacity, and that these trends hold across unseen domains, rephrased queries, and query formats.

A practical concern that emerges at larger scales is that the hypernetwork itself can become prohibitively large relative to the target model it is adapting. In our highest-capacity experiments, the hypernetwork reached approximately 2.5B parameters while the target model was only 1.5B parameters, making the hypernetwork nearly as large as the target model. This parameter overhead significantly limits the practical deployability of large hypernetworks and motivates future work on more parameter-efficient hypernetwork architectures, such as shared weight generation across layers, low-rank hypernetwork designs, or distillation of large hypernetworks into smaller ones after training. Our target model scaling experiments cover the `Qwen2.5` family up to 14B parameters. Extending this analysis to frontier-scale models (70B and beyond) remains an important open question, as the relationship between hypernetwork capacity and target model size may exhibit qualitatively different behavior at much larger scales. Our current evaluation covers single-hop and shallow multi-hop queries from `Wikidata5M`. A natural extension is to study whether hypernetwork-based injection supports deep multi-hop compositional reasoning and long-horizon reasoning, where answering a query requires the simultaneous use of many more injected facts. Whether inference-time weight generation can scale to this regime, and how the required hypernetwork capacity grows with reasoning depth, are open questions that we leave to future work.

References

Louis Béthune, David Grangier, Dan Busbridge, Eleonora Gualdoni, Marco Cuturi, and Pierre Ablin. Scaling laws for forgetting during finetuning with pretraining data injection. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/bethune25a.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Qizhou Chen, Dakan Wang, Taolin Zhang, Zaoming Yan, Chengsong You, Chengyu Wang, and Xiaofeng He. Uniedit: A unified knowledge editing benchmark for large language models. *CoRR*, abs/2505.12345, 2025. doi: 10.48550/ARXIV.2505.12345. URL https://doi.org/10.48550/arXiv.2505.12345.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Trans. Assoc. Comput. Linguistics*, 12:283–298, 2024. doi: 10.1162/TACL_A_00644. URL https://doi.org/10.1162/tacl_a_00644.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7765–7784. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.444. URL https://doi.org/10.18653/v1/2024.emnlp-main.444.

Jeffrey Gu and Serena Yeung-Levy. Foundation models secretly understand neural network weights: Enhancing hypernetwork architectures with foundation models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=cADpvQgnqg.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pages 15202–15232. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.902. URL https://doi.org/10.18653/v1/2024.findings-acl.902.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Hossein Hajipour, Ning Yu, Cristian-Alexandru Staicu, and Mario Fritz. Simscood: Systematic analysis of out-of-distribution generalization in fine-tuned source code models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, Findings of ACL, pages 1400–1416. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.90. URL https://doi.org/10.18653/v1/2024.findings-naacl.90.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VrHiF2hsrm.

Sara Kothari and Ayush Gupta. Question answering on patient medical records with private fine-tuned llms. *CoRR*, abs/2501.13687, 2025. doi: 10.48550/ARXIV.2501.13687. URL https://doi.org/10.48550/arXiv.2501.13687.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia, editors, *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 333–342. Association for Computational Linguistics, 2017. doi: 10.18653/V1/K17-1034. URL https://doi.org/10.18653/v1/K17-1034.

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Findings of ACL, pages 4297–4308. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.249. URL https://doi.org/10.18653/v1/2024.findings-emnlp.249.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, Carl Yang, and Liang Zhao. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *ACM Comput. Surv.*, 58(3): 79:1–79:39, 2026. doi: 10.1145/3764579. URL https://doi.org/10.1145/3764579.

Jiateng Liu, Zhenhailong Wang, Xiaojiang Huang, Yingjie Li, Xing Fan, Xiang Li, Chenlei Guo, Ruhi Sarikaya, and Heng Ji. Analyzing and internalizing complex policy documents for LLM agents. *CoRR*, abs/2510.11588, 2025a. doi: 10.48550/ARXIV.2510.11588. URL https://doi.org/10.48550/arXiv.2510.11588.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press, 2020. doi: 10.1609/AAAI.V34I03.5681. URL https://doi.org/10.1609/aaai.v34i03.5681.

Zeyu Leo Liu, Greg Durrett, and Eunsol Choi. Propmend: Hypernetworks for knowledge propagation in llms. *CoRR*, abs/2506.08920, 2025b. doi: 10.48550/ARXIV.2506.08920. URL https://doi.org/10.48550/arXiv.2506.08920.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 -*

December 9, 2022, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? comparing knowledge injection in LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 237–250. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.15.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/J.NEUCOM.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. Wikibigedit: Understanding the limits of lifelong knowledge editing in llms. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/thede25a.html.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, Gertjan van Noord, and Sebastian Ruder. Hyper-x: A unified hypernetwork for multi-task multilingual transfer. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7934–7949. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.541. URL https://doi.org/10.18653/v1/2022.emnlp-main.541.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL-IJCNLP 2021 of *Findings of ACL*, pages 1405–1418. Association for Computational Linguistics, 2021a. doi: 10.18653/V1/2021.FINDINGS-ACL.121. URL https://doi.org/10.18653/v1/2021.findings-acl.121.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194, 2021b. doi: 10.1162/TACL_A_00360. URL https://doi.org/10.1162/tacl_a_00360.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *CoRR*, abs/2306.06031, 2023. doi: 10.48550/ARXIV.2306.06031. URL https://doi.org/10.48550/arXiv.2306.06031.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1139. URL https://doi.org/10.18653/v1/p19-1139.

A Calculating Training Compute for Hypernetworks

Kaplan et al. [2020] uses $2N$ as the number of floating-point operations (FLOPs) required for a transformer forward pass per token, and $6N$ per training iteration per token, where N is the number of non-embedding parameters. In our experiments, we consider hypernetworks that contain a comparable number of parameters to that of the target model. In order to compute the number of FLOPs during training with hypernetworks, we adapt the formula appropriately:

1. **Cost of hypernetwork forward pass:** $2n_hN_h$ where N_h is the number of non-embedding hypernetwork parameters and n_h is the number of input tokens in the hypernetwork.
2. **Cost of target model forward pass:** $2n_tN_t$ where N_t is the number of non-embedding parameters in the target model and n_t is the number of input tokens in the target model.
3. **Cost of target model backward pass:** $2n_tN_t\frac{L_{target}}{L}$ where L is the number of layers in the target model and L_{target} is the number of layers targeted by the hypernetwork, assuming we target the last L_{target} layers. Note that in order to compute the gradients in the hypernetwork, we must backpropagate the gradients through the last L_{target} layers of the target network (and no further). During this backward pass, we only need to compute the gradients of the loss function with respect to the previous layer's activations, and we do not need to compute the gradients with respect to the weights in the target network. As a result, for any linear layer, we only need one matrix multiplication in the backward pass, hence the coefficient 2.
4. **Cost of hypernetwork backward pass:** $4n_hN_h$.

Thus the total cost C of a training iteration in FLOPs on one example is:

$$C = 6n_hN_h + 2n_tN_t\left(1 + \frac{L_{target}}{L}\right). \tag{1}$$

In our experiments, the average length of the queries for the target model is roughly 15 tokens, and so we fix $n_t = 15$. Each injected fact contributes on average 11.4 tokens to the hypernetwork input, so we set $n_h = 11.4 \times N_{\text{facts}}$. For width, depth, and target-model scaling, which use $N_{\text{facts}} = 4$ injected facts, which gives $n_h = 45.6$.

Tables 2–5 report the per-example FLOPs for each scaling axis in our experiments. For the depth scaling experiments, varying L_{HN} from 1 to 16 increases N_h by only 1.9%, since the LoRA projection heads dominate N_h at $d_{\text{model}} = 256$. Consequently, depth scaling is effectively **iso-compute**, with FLOPs per example varying by less than 2% across the entire depth range, meaning that performance differences observed in Section 5.2 are attributable to architectural capacity rather than additional compute. Width and fact-count scaling are not iso-compute: varying d_{model} from 64 to 1024 introduces a $7.4\times$ compute range, and varying N_{facts} from 2 to 52 introduces a $13.7\times$ compute range through n_h.

TABLE 2: FLOPs per example for hypernetwork width scaling (target: `Qwen2.5-1.5B`, $L_{\text{HN}} = 4$, $n_h = 45.6$).

d_{model}	L_{HN}	N_h (w/ embed)	N_h (w/o embed)	FLOPs/example
64	4	167,398,376	157,691,816	1.021×10^{11}
128	4	323,068,776	313,362,216	1.447×10^{11}
256	4	635,580,904	625,874,344	2.302×10^{11}
512	4	1,265,323,752	1,255,617,192	4.025×10^{11}
1024	4	2,543,683,816	2,533,977,256	7.523×10^{11}

TABLE 3: FLOPs per example for hypernetwork depth scaling (target: `Qwen2.5-1.5B`, $d_{\text{model}} = 256$, $n_h = 45.6$).

d_{model}	L_{HN}	N_h (w/ embed)	N_h (w/o embed)	FLOPs/example
256	1	633,211,624	623,505,064	2.296×10^{11}
256	2	634,001,384	624,294,824	2.298×10^{11}
256	4	635,580,904	625,874,344	2.302×10^{11}
256	8	638,739,944	629,033,384	2.311×10^{11}
256	16	645,058,024	635,351,464	2.328×10^{11}

TABLE 4: FLOPs per example for fact-count scaling (target: `Qwen2.5-1.5B`, $d_{\text{model}} = 256$, $L_{\text{HN}} = 4$, fact embedding dim $= 512$). The 52-fact run uses $n_h = 512$ due to the hypernetwork context window limit.

N_{facts}	n_h	N_h (w/ embed)	N_h (w/o embed)	FLOPs/example
2	22.8	703,641,512	625,989,032	1.446×10^{11}
4	45.6	703,641,512	625,989,032	2.302×10^{11}
8	91.2	703,641,512	625,989,032	4.014×10^{11}
16	182.4	703,641,512	625,989,032	7.440×10^{11}
32	364.8	703,641,512	625,989,032	1.429×10^{12}
52	512	703,641,512	625,989,032	1.982×10^{12}

B Domain Classification

In this section, we provide further details on the domain classification procedure, which is utilized in the construction of our dataset, `MegaWikiQA`. Each example in `MegaWikiQA` is derived from a triplet (s, r, o) in `Wikidata5M`, and we assign each triplet to one of 39 semantic domains (e.g., geography, political science, music) using a two-stage classification pipeline described below. We assign each example to a semantic domain to enable controlled evaluation of generalization across domains. Domain labels allow us to construct OOD splits by holding out entire domains during training, providing a stricter test of whether injected knowledge generalizes beyond the distribution of observed facts.

Domain taxonomy. We adopt a fixed set of 39 domains, some were extracted from prior work on knowledge editing benchmarks (e.g., `UniEdit` [Chen et al., 2025]), ensuring consistency with existing datasets and facilitating comparability. To align our data with this taxonomy, we train a domain classifier on triplet-domain pairs derived from the `UniEdit` dataset and GPT-4.1.

Classification pipeline. Given a triplet (s, r, o), we assign a domain label using a two-stage pipeline. In the first stage, we prompt GPT-4.1 to predict a domain label from the predefined set of 39 domains based on the semantics of the triplet. In the second stage, we apply a fine-tuned LLM with a classifier head to validate the prediction. For training data, we first collected around 100K high quality labelled samples using multiple calls of GPT models & voting ensemble. We then performed supervised finetuning of classifier with class-wise cross-entropy loss. We perform rejection filtering by discarding examples for which the classifier assigns low confidence, retaining only high-confidence domain labels.

Quality control. This process yields domain labels with over 95% manually verified classification accuracy on a held-out validation set, and a 92% agreement rate with GPT-4.1 predictions. Filtering low-confidence predictions reduces label noise and ensures reliable domain assignments for downstream evaluation.

Application to multi-hop examples. Domain labels are first assigned to single-hop triplets and then propagated to multi-hop examples based on their underlying triplet sequences. Using these labels, we construct stratified splits that are balanced in both domain coverage and reasoning complexity (i.e., number of hops), ensuring that scaling experiments are not confounded by distributional imbalances.

TABLE 5: FLOPs per example for target model scaling ($d_{\text{model}} = 256$, $L_{\text{HN}} = 4$, $n_h = 45.6$).

Target	N_h (w/ embed)	N_h (w/o embed)	FLOPs/example
0.5B	309,772,752	300,066,192	9.820×10^{10}
1.5B	635,580,904	625,874,344	2.302×10^{11}
3B	909,808,128	900,101,568	3.665×10^{11}
7B	2,100,000,000	2,090,293,440	8.795×10^{11}
14B	2,800,000,000	2,790,293,440	1.292×10^{12}

Domain-level base model accuracy. Figure 8 shows the base model accuracy across all 39 domains. The three held-out OOD domains (*philosophy*, *linguistics*, and *civil engineering*) exhibit notably higher base model accuracy than the in-distribution domains, lying approximately 4.8% above the in-distribution mean. This has an important implication for the interpretation of our OOD results: because the frozen target model already has stronger prior knowledge of these domains, the OOD non-rephrased loss is close to the ID validation loss across all scaling experiments, as observed in the scaling law figures in Section 5. The OOD rephrased and OOD MCQ splits, which test robustness to linguistic variation and format shift respectively, provide a stricter evaluation that is less confounded by base model familiarity.

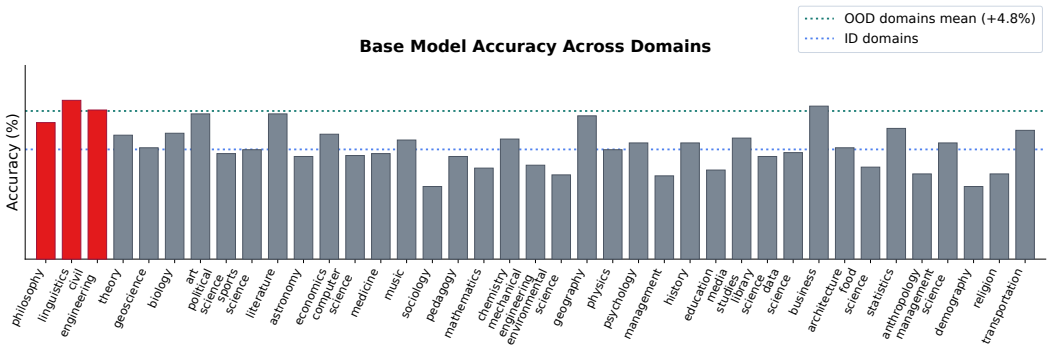

FIGURE 8: Base model accuracy across all 39 knowledge domains. The three held-out OOD domains (*philosophy*, *linguistics*, *civil engineering*) exhibit high base model accuracy, lying approximately 4.8% above the mean of the in-distribution domains.

C Dataset Statistics

In this section, we provide full statistics for the `MegaWikiQA` dataset. Table 6 summarizes all splits used in our experiments. All evaluation splits are disjoint from the training set at the triplet level, and the three OOD domains (*philosophy*, *linguistics*, *civil engineering*) are fully withheld during training.

TABLE 6: Dataset statistics. All splits have zero entity and triplet overlap with the training set. OOD domains are fully held out during training.

Split	Type	Number of hops	Examples
Training	In-distribution	1, 2, 3, 4	1,250,000
ID Eval	In-distribution	1, 2, 3, 4	10,000
OOD Non-rephrased	Held-out domains	1, 2, 3, 4	10,000
OOD Rephrased	Held-out domains	1, 2, 3, 4	10,000
OOD MCQ	Held-out domains	1, 2, 3, 4	10,000

D Hypernetwork Architecture Details

This section covers full architectural details of the hypernetwork g_ϕ introduced in Section 4.2. This includes the input encoding scheme, the transformer encoder design, the LoRA weight generation procedure, the target layer selection strategy, and the hyperparameter tuning approach used across all scaling experiments.

D.1 Input Encoding

Each fact $f_i \in \mathcal{F}$ is serialized as a natural language sentence and tokenized using the same tokenizer as the target language model, with a maximum sequence length of 128 tokens. The fact set $\mathcal{F}$ is concatenated and processed by the hypernetwork transformer encoder. To obtain a single fixed-size representation of $\mathcal{F}$, we apply masked mean pooling over all non-padding token positions. This pooling strategy is parameter-free and treats all token positions symmetrically, avoiding the inductive bias of token-specific aggregation schemes such as `[CLS]` pooling.

D.2 Encoder Architecture

The hypernetwork transformer encoder uses rotary positional embeddings (RoPE) Su et al. [2024] applied to the query and key projections of each attention layer, providing relative positional awareness without absolute position bias. Each transformer block follows a post-norm architecture, where layer normalization is applied after the residual connection rather than before, which we find improves training stability across the range of hypernetwork depths explored in our scaling experiments.

D.3 LoRA Weight Generation

For each target weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ in the selected layers of the target model, the hypernetwork maintains two dedicated linear projection heads that map the pooled representation $\mathbf{h}$ to the corresponding low-rank factors:

$$A = (\tanh(W_A \mathbf{h}))^\top \in \mathbb{R}^{r \times d_{\text{in}}}, \qquad B = (\tanh(W_B \mathbf{h}))^\top \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{2}$$

where r is the LoRA rank, $W_A \in \mathbb{R}^{(r \cdot d_{\text{in}}) \times d_{\text{model}}}$ and $W_B \in \mathbb{R}^{(d_{\text{out}} \cdot r) \times d_{\text{model}}}$ are learned projection matrices, and the outputs are reshaped from vectors to matrices of the indicated dimensions. The weight adaptation applied to the target model is then $\Delta W = \alpha c BA/r$ where r is the LoRA rank, α is the scaling factor, and $c = 0.01$ is a fixed scalar that controls the magnitude of the generated adaptations. We use $r = 4$ and $\alpha = 8$ across all experiments. LoRA adaptations are applied to all linear layers within the selected transformer blocks of the target model, covering both attention projection matrices and feed-forward layers, to maximize the expressivity of the generated adaptations.

D.4 Target Layer Selection

We apply hypernetwork-generated LoRA adaptations to the later $\lfloor L/2 \rfloor$ layers of the target model, where L denotes the total number of transformer layers. This later-layer adaptation strategy concentrates the injection signal into the higher-level semantic representations of the target model, which prior work has shown to be the primary locus of factual knowledge [Meng et al., 2022]. The proportion $\lfloor L/2 \rfloor$ is held fixed across all target model sizes, ensuring that the fraction of adapted parameters scales consistently with model depth.

D.5 Training Objective

The hypernetwork is trained end-to-end by minimizing cross-entropy loss over the answer tokens under the adapted target model. Gradients flow through the target model and back into the hypernetwork parameters ϕ via the LoRA adaptation matrices. The target model parameters θ receive no gradient updates and remain frozen throughout.

D.6 Hyperparameter Tuning

We use the AdamW [Loshchilov and Hutter, 2019] optimizer with weight decay $\lambda = 10^{-5}$ across all experiments. To identify the best scheduler and learning rate range efficiently, we first conducted a

broad sweep across schedulers and learning rates on a small subset of 10K–80K training examples, before scaling up to the full 1.25M-example training set. Figure 9 shows an example of such a sweep for the largest width configuration ($d_{\text{model}} = 1024$), illustrating how cosine annealing with AdamW consistently outperformed polynomial decay and the Muon optimizer across all learning rates tested. Based on these small-scale experiments, we adopted cosine annealing with AdamW as the scheduler for all scaling experiments.

Rather than performing an exhaustive hyperparameter search for each configuration at full scale, we adopt an efficient *anchor-and-interpolate* strategy for learning rate selection. For each scaling variable (e.g., hypernetwork depth, width, etc), we first run experiments at the lowest and highest values in the experimental range, followed by a third experiment at the midpoint (arithmetic mean). We perform hyperparameter optimization at these three anchor points, then fit a power-law of the form $\eta^*(x) = a \cdot x^b + c$, where x is the scaling variable and the coefficients a, b, c are estimated via least-squares regression in log scale. Learning rates for all intermediate configurations are obtained by evaluating this fitted curve. The optimal learning rate ranges observed across scaling axes were as follows: for width scaling, η^* decreased from 5×10^{-4} to 6×10^{-5} as d_{model} increased from 64 to 1024; for depth scaling, the range was narrower (2×10^{-4} to 9×10^{-5}) owing to the similar parameter counts across depth configurations; for target model scaling, η^* ranged from 6×10^{-4} to 4×10^{-5}; and for fact count scaling, from 2×10^{-4} to 7×10^{-5}.

The number of training epochs is selected per experiment based on convergence behavior. We continue training until the rate of improvement becomes negligible and then continue training for a small number of additional epochs to confirm convergence. In practice, the number of epochs ranged from 5 to 8 across all experiments. For configurations with lower FLOPs per example (e.g., small hypernetwork widths or few injected facts), training is extended for additional epochs to ensure that these runs reach a comparable level of convergence to their higher-compute counterparts, enabling fair comparisons across the scaling axis. Batch size is selected independently for each experiment based on training loss stability.

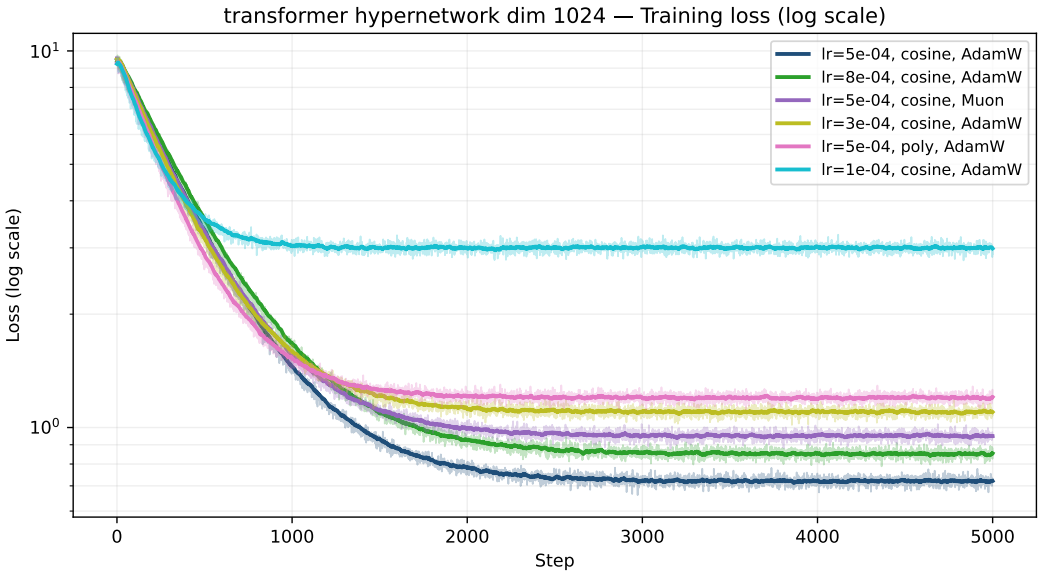

FIGURE 9: Hyperparameter sweep results for the largest width configuration ($d_{\text{model}} = 1024$), showing training loss on a log scale across different learning rates, schedulers, and optimizers on a small-scale subset. Cosine annealing with AdamW with learning rate 5×10^{-4} achieves the lowest final training loss, motivating its selection for all scaling experiments.

D.7 Target Model Scaling

To characterize how knowledge injection performance scales with the capacity of the frozen target model, we conduct experiments across five model sizes from the `Qwen2.5` family Yang et al. [2024]: **0.5B**, **1.5B**, **3B**, **7B**, and **14B** parameters. The hypernetwork architecture is held fixed across all target

model sizes, and LoRA adaptations are applied to the later $\lfloor L/2 \rfloor$ layers in each case, where L varies with target model depth. This design isolates the effect of target model capacity from hypernetwork capacity, enabling a clean comparison of the effect of increasing training compute in the target model versus the hypernetwork.

E LoRA Rank Scaling

To complement the target-model-scale comparison in Section 5.5, we also examine how LoRA finetuning scales with adapter capacity. We train LoRA finetuning baselines across ranks $r \in \{2, 4, 8, 16, 32, 64\}$ with scaling factor $\alpha = 2r$, holding all other hyperparameters fixed and using the same target model as the hypernetwork experiments. Figure 10 shows the final epoch loss against r.

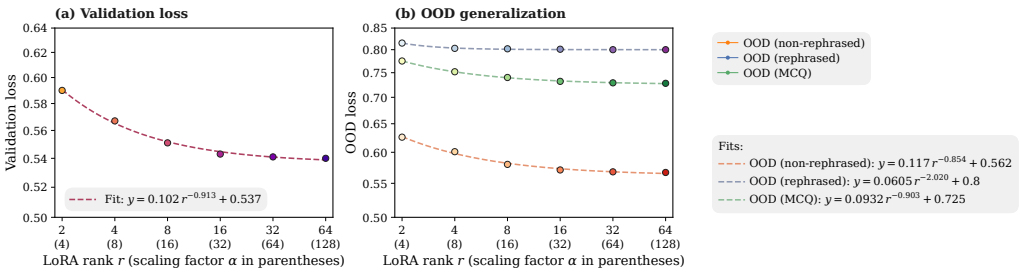

FIGURE 10: LoRA rank scaling results, showing final epoch loss vs. LoRA rank r. Unlike other scaling axes, LoRA rank scaling exhibits a clear saturating regime and is well-fit by a power-law with a constant offset: $\mathcal{L}(r) = a \cdot r^b + c$. Fitted parameters: validation $0.102 \cdot r^{-0.913} + 0.537$, OOD non-rephrased $0.117 \cdot r^{-0.854} + 0.562$, OOD rephrased $0.0605 \cdot r^{-2.020} + 0.800$, and OOD MCQ $0.0932 \cdot r^{-0.903} + 0.725$.

Unlike the other scaling axes, LoRA rank scaling exhibits a clear saturating regime and cannot be captured by a pure power-law. We therefore fit a power-law with an additive constant, $\mathcal{L}(r) = a \cdot r^b + c$, where c represents the asymptotic loss floor. The fitted floor c is substantial in every metric (0.537 for validation, 0.562, 0.800, and 0.725 for OOD non-rephrased, rephrased, and MCQ respectively), meaning that increasing adapter capacity beyond a modest rank produces diminishing returns and cannot close the OOD gap to the hypernetwork. This saturating behavior contrasts with the target-model-scale results in Section 5.5, where LoRA continues to improve without a visible plateau, indicating that target model capacity is the dominant scaling axis for LoRA-based knowledge injection.

F Loss Trajectories During Training

In this section, we provide the full loss trajectories for all scaling experiments, plotting validation loss and OOD non-rephrased loss against cumulative training FLOPs. Using FLOPs as the x-axis rather than steps or epochs enables direct comparison across configurations that differ in per-example compute cost (see Appendix A). In all plots, darker colors correspond to larger configurations (i.e., wider, deeper, larger target model, or more facts). The monotonic ordering of the loss values by configuration is established early during training and largely maintained throughout, confirming that the scaling trends reported in Section 5 are stable and not artifacts of training length.

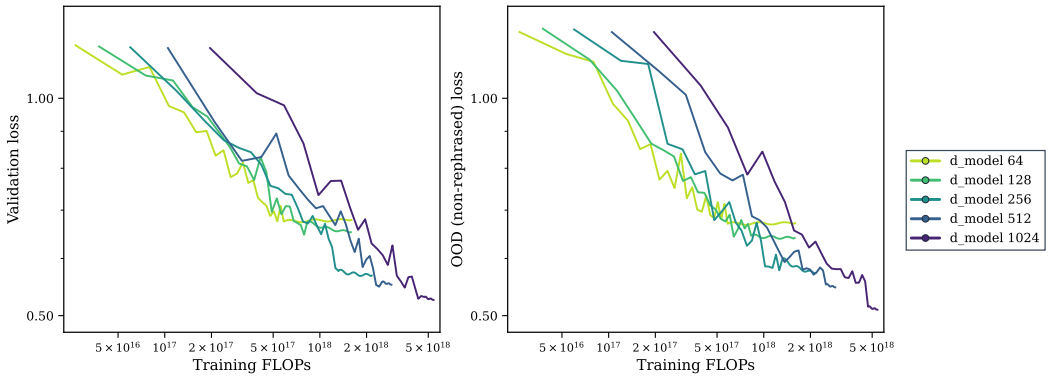

FIGURE 11: Loss trajectories for hypernetwork width scaling, showing validation loss (left) and OOD non-rephrased loss (right) vs. cumulative training FLOPs for $d_{\text{model}} \in \{64, 128, 256, 512, 1024\}$.

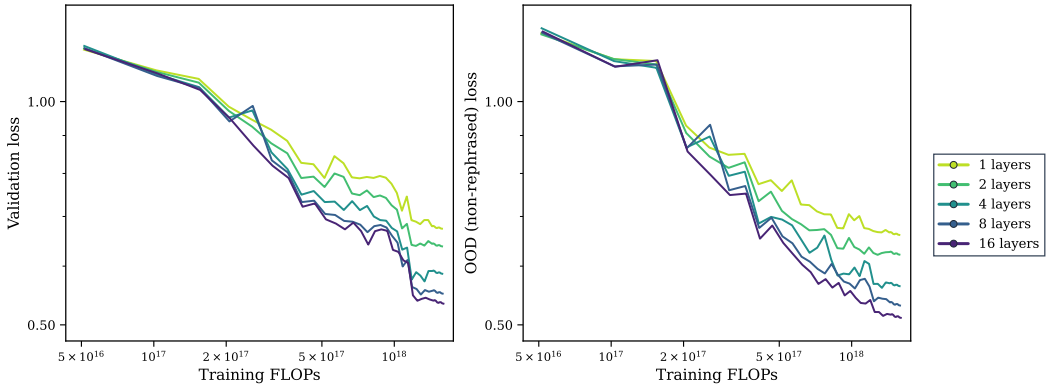

FIGURE 12: Loss trajectories for hypernetwork depth scaling, showing validation loss (left) and OOD non-rephrased loss (right) vs. cumulative training FLOPs for $L_{\text{HN}} \in \{1, 2, 4, 8, 16\}$. The narrow FLOPs range reflects the near iso-compute nature of depth scaling (see Table 3).

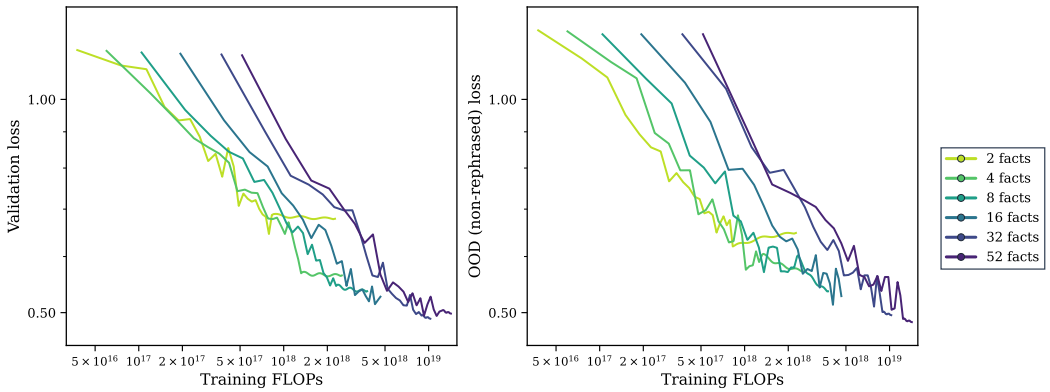

FIGURE 13: Loss trajectories for fact count scaling, showing validation loss (left) and OOD non-rephrased loss (right) vs. cumulative training FLOPs for $N_{\text{facts}} \in \{2, 4, 8, 16, 32, 52\}$. Configurations with more facts reach higher total FLOPs due to the larger n_h (see Table 4).

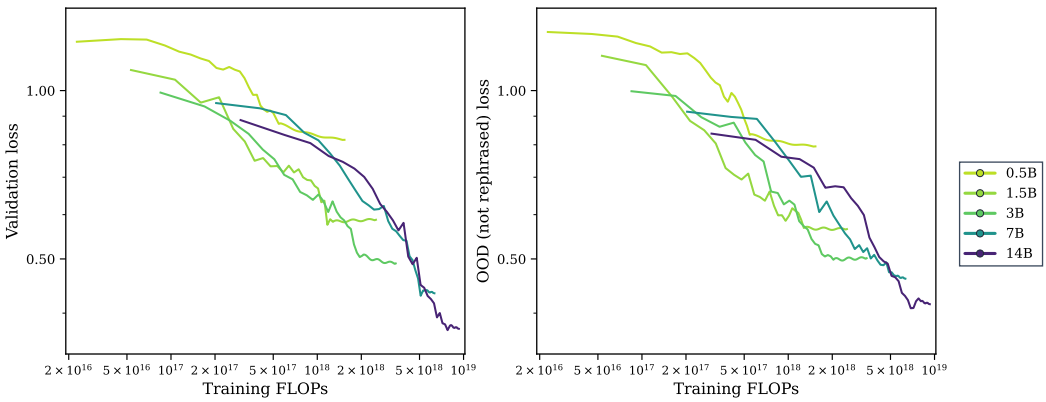

FIGURE 14: Loss trajectories for target model scaling, showing validation loss (left) and OOD non-rephrased loss (right) vs. cumulative training FLOPs for target model sizes 0.5B, 1.5B, 3B, 7B, and 14B. Larger target models incur higher per-example FLOPs and therefore reach higher total FLOPs at the same number of epochs.