

# TaskWeaver: Instance-Level Test-Time Adaptation for LLMs via Hypernetwork-Generated LoRA Weights

## Final Executive Summary

Dhruv Kapur & Andrews George Varghese & Raj Maheshwari  
{dkapur, ageorgev, rajmahes}@andrew.cmu.edu  
10-423/623 Generative AI Course Project

December 11, 2025

### Abstract

We present TaskWeaver, a hypernetwork architecture that generates instance-level LoRA weights for language models during inference. Unlike prior work that produces task-level adapters from natural language descriptions, TaskWeaver generates unique LoRA parameters for each individual input based solely on its semantic content, no task labels or descriptions required. Our self-referential design uses the frozen base model as both semantic encoder and adaptation target. We evaluate across three model scales (70M, 270M and 600M parameters) and eleven benchmarks. Results show TaskWeaver achieves competitive performance with task-specific LoRA, particularly excelling on mathematical reasoning (52.5% vs 34.5% on GSM8K for Qwen3-0.6B), while also showing impressive generalization capabilities. Analysis also reveals a well structured semantic space of generated LoRA weights through T-SNE analysis.

### 1 Introduction

Large language models achieve remarkable success across tasks, yet adapting them typically requires expensive fine-tuning. Low-Rank Adaptation (LoRA) [Hu et al., 2022] reduces this burden by constraining updates to low-rank decompositions, but still requires training separate adapters per task in advance.

Recent work on Text-to-LoRA [Charakorn et al., 2025] demonstrated that hypernetworks can generate task-specific LoRA adapters from natural language descriptions. However, this operates at *task-level* granularity, producing one set of weights per task description, and requires carefully crafted descriptions for well-performing LoRA adapters.

We want to side-step the requirement for crafting high-quality task descriptions for each input by extending to **instance-level adaptation**: generating

unique LoRA parameters for each input during inference based solely on its semantic content. This offers: (1) no task identifiers (labels, descriptions, etc.) required, (2) finer-grained per-input optimization, (3) seamless multi-task handling, and (4) better generalization to novel inputs.

**Contributions:** (1) A hypernetwork using self-referential encoding where the base model serves as both encoder and target; (2) `DynamicLoraLinear`, enabling batched instance-level LoRA weight injection at runtime; (3) comprehensive evaluation across 3 models and 11 benchmarks including zero-shot testing; (4) analysis showing generated weights cluster by task type with no priors other than input semantics.

### 2 Dataset, Task, and Model Selection

**Task.** Given input  $x$ , TaskWeaver generates LoRA matrices  $\{A_{\ell,m}, B_{\ell,m}\}$  for each layer  $\ell$  and module  $m$ , injected into a frozen base model. Training minimizes language modeling loss end-to-end.

**Training Data (8 benchmarks):** GSM8K [Cobbe et al., 2021] (math), ARC-Easy/Challenge [Clark et al., 2018] (science), BoolQ [Clark et al., 2019] (reading comprehension), SNLI [Bowman et al., 2015] (natural language inference), OpenBookQA [Mihaylov et al., 2018] (multi-hop reasoning), HellaSwag [Zellers et al., 2019] (commonsense), Winogrande [Sakaguchi et al., 2020] (pronoun resolution). More details about the datasets are available in Appendix. D.

**Zero-Shot Evaluation (3 benchmarks):** SVAMP (math) [Li et al., 2024], CommonsenseQA [Talmor et al., 2019], RACE-Middle [Lai et al., 2017] (reading). These datasets are held out during training and only used for evaluation.

**Models:** Pythia-70M [Biderman et al., 2023] (base), Gemma3-270M-IT (instruction-tuned), Qwen3-0.6B (instruction-tuned). This spans base vs. instruction-tuned with varying capacities.

### 3 Related Work

**LoRA** [Hu et al., 2022] represents weight updates as  $\Delta W = BA$  with  $r \ll \min(d, k)$ , reducing parameters by  $10,000\times$  while matching full fine-tuning.

**Hypernetworks** [Ha et al., 2017] generate weights for other networks. HyperFormer [Mahabadi et al., 2021] extends this to transformers, generating adapter parameters from task embeddings with 0.29% additional parameters per task.

**Meta-Learning.** MAML [Finn et al., 2017] enables rapid adaptation via bi-level optimization but requires gradient updates during adaptation; hypernetworks generate parameters in one forward pass.

**Text-to-LoRA** [Charakorn et al., 2025] generates complete LoRA adapters from task descriptions, achieving 98% of specialized adapter performance. HyperDecoders [Iverson and Peters, 2022] treat parameter generation as autoregressive sequence modeling.

**Our Contribution:** TaskWeaver extends to instance-level granularity, generating unique weights per input without task labels.

## 4 Methods

### 4.1 Baseline: Trained LoRA Adapters

**LoRA (Individual):** Separate adapters trained independently for each task. This represents the upper bound of task-specific optimization but requires  $N$  training runs for  $N$  tasks.

**LoRA (Mixed):** A single adapter trained on all tasks combined. This matches TaskWeaver’s training data and represents a naive multi-task baseline.

Both baselines use identical LoRA configuration: rank  $r = 2$ ,  $\alpha = 8$ , applied to query and value projections.

### 4.2 TaskWeaver Architecture

The architecture overview can be seen in Figure 1.

**Semantic Encoding.** We use the frozen base LM itself as encoder (self-referential design). Given input tokens, we extract the hidden state from the last prompt token at the final layer as embedding  $h \in \mathbb{R}^d$ .

**Conditioning.** Following Charakorn et al. [2025], we add learned embeddings to the semantic embedding:

$$c_{\ell,m,k} = \text{proj}(h) + e_{\text{layer}}^{(\ell)} + e_{\text{module}}^{(m)} + e_{\text{matrix}}^{(k)} \quad (1)$$

where  $e_{\text{layer}}$ ,  $e_{\text{module}}$ ,  $e_{\text{matrix}}$  are learned embeddings for layer index, module type (query projection, value projection, etc.), and matrix type (LoRA A and LoRA B).

Different layers and modules serve different functions in a transformer. Early layers often handle syntactic patterns while later layers capture semantics. Query projections attend to relevant context while value projections determine what information to extract. The conditioning embeddings allow the hypernetwork to specialize its predictions for each location in the LLM backbone.

**Weight Generation.** A shared 2-layer MLP  $\Psi$  with GELU and LayerNorm processes conditioned representations. Separate output heads per module-matrix pair handle varying dimensions:

$$A_{\ell,m} = W_{m,A} \cdot \Psi(c_{\ell,m,A}) + b_{m,A} \quad (2)$$

$$B_{\ell,m} = W_{m,B} \cdot \Psi(c_{\ell,m,B}) + b_{m,B} \quad (3)$$

**Dynamic Injection.** Our DynamicLoraLinear accepts batch-specific weights, computing:

$$\text{out}_i = Wx_i + \frac{\alpha}{r} B_i A_i x_i + b \quad (4)$$

where  $A_i, B_i$  are unique per batch element  $i$ , requiring  $A \in \mathbb{R}^{b \times r \times d_{\text{in}}}$  and  $B \in \mathbb{R}^{b \times d_{\text{out}} \times r}$  tensors.  $W$  and  $b$  are copied over from the corresponding module in the pretrained base LM.

An example of a PyTorch instantiation of TaskWeaver is present in Appendix A.

### 4.3 Training

We train end-to-end through the frozen base model: (1) predict LoRA weights from input, masking the completion tokens from the hypernetwork, (2) inject predicted weights into the frozen model through DynamicLoraLinear, (3) compute language modelling loss, (4) update only hypernetwork parameters. Output heads are zero-initialized for small initial adaptations, and this is crucial for stable training (also mentioned by Charakorn et al. [2025]).

## 5 Experiments

### 5.1 Setup

**Hardware:** Access to NVIDIA A100, NVIDIA L40S, RTX 3090, RTX 3070, Apple M4 Pro across team members for varying amounts of time.

**Hyperparameters:** Hidden dim 1024, LoRA rank 2,  $\alpha = 8$ . Batch sizes: 16/8/4 for Pythia/Gemma/Qwen. Learning rate  $10^{-6}$ , AdamW, for 3 epochs. Training times: 3:06 / 16:30 / 47:25 (mm:ss). More details are available in Appendix. E.

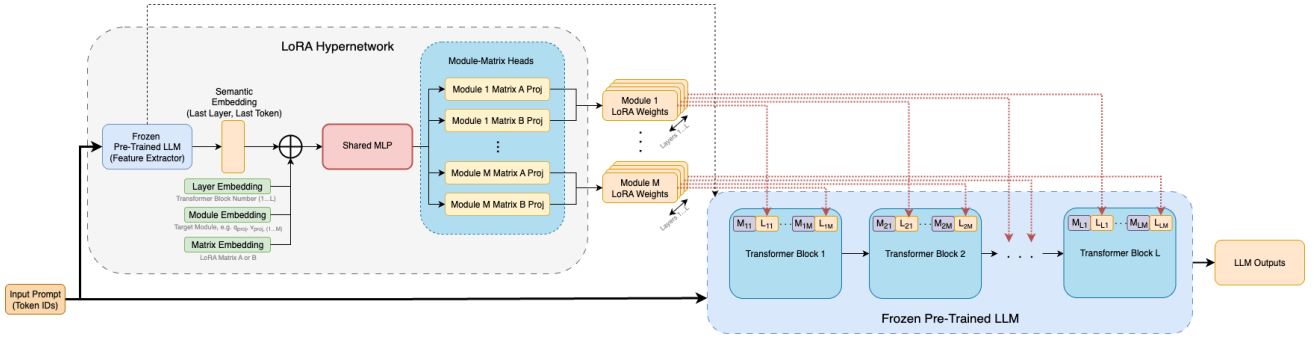


Figure 1: **Overview of the TaskWeaver architecture.** The hypernetwork (left) takes an input prompt and processes it through a frozen pre-trained LLM to extract a semantic embedding of the prompt (taken as the last layer representation of the last token). These embeddings are added to learned layer, module and matrix embeddings (A and B), then passed through a shared MLP backbone. Module-specific prediction heads generate LoRA weight matrices (A and B) for different modules (e.g., `q_proj`, `v_proj`, etc.) across all transformer layers. The generated LoRA weights are dynamically injected into the frozen base LLM (right) at inference time, adapting the model’s behavior on a per-input basis without requiring gradient-based optimization.

## 5.2 Main Results

Table 1 presents comprehensive results.

**Pythia-70M:** TaskWeaver achieves best performance on 6/11 benchmarks including both zero-shot tasks, being consistently better than the base model, while being a close second in cases where it didn’t have the best performance.

Note how most performance numbers increase drastically. This is because the base Pythia model is unable to generate responses aligned with the question on its own, and hence fails to even answer the prompt. Fine-tuning helps align the model’s outputs to match each task’s expectations, and leads to the improvement. An example is provided in Appendix. D Table. 2. The performance numbers themselves are close to that of random predictions, which is expected from such a small model.

**Gemma3-270M-IT:** Competitive performance with best GSM8K (5.84%) and strong zero-shot results. The instruction-tuned base provides a higher starting point.

**Qwen3-0.6B:** Most interesting patterns. While LoRA (Individual) wins on in-distribution tasks, TaskWeaver excels on GSM8K (52.5% vs 34.5% Mixed) and SVAMP (74.7% vs 52.7%). This demonstrates robustness to training data skew. Further, we notice that in some cases, LoRA finetuning hinders the base model’s performance (GSM8K 50%  $\rightarrow$  35%, SVAMP 73%  $\rightarrow$  53.7%), but TaskWeaver always improves upon the base model, preserving its original capabilities.

**Key Findings:** TaskWeaver consistently outperforms the base model, or is close enough to it, indicating that it builds on top of, without hindering, the base model’s capabilities. We also see that TaskWeaver

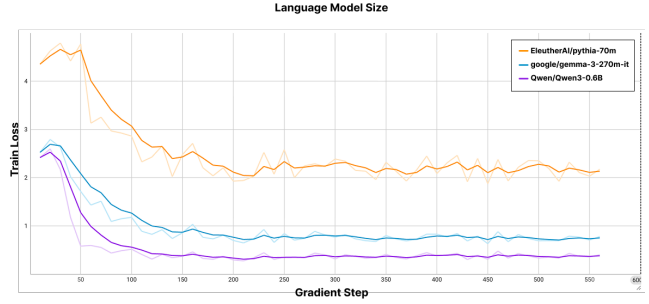


Figure 2: Train Loss vs Model Size

is generally comparable, or sometimes even outperforms task-specific LoRA adapters. Finally, we see the biggest gains from using TaskWeaver on the smallest model (Pythia-70M), but we hypothesize that this may be due to the limited expressiveness of the hypernetwork in our experiments (a simple MLP with 2 hidden layers).

## 5.3 Scaling Analysis

We analyze hyperparameter effects on training:

**Model Size:** Larger models show smoother training, faster convergence, lower final loss. Qwen converges to  $\sim 0.5$  loss vs Pythia’s  $\sim 2.0$  (Figure 2). We also observe that larger models are more invariant to hyperparameter changes.

**Hidden Dimension:** Increasing the hidden dimension of the hypernetwork, we see training loss decrease monotonically.

**LoRA Rank:** Increasing the rank of the predicted adapters, we see training loss decrease monotonically.

**LoRA Alpha:** Increasing the alpha of the predicted adapters, we see training loss decrease monotonically.

Model	Mode	ARC-C	ARC-E	BoolQ	GMS8K	HSWAG	OBQA	SNLI	WG	SVAMP*	CSQA*	RACE*
EleutherAI/pythia-70m	Base	9.13%	7.07%	14.2%	1.29%	4.94%	8.2%	0.519%	6.08%	2.33%	3.44%	8.64%
	LoRA (Individual)	22.8%	23.7%	53.5%	1.14%	<b>25.4%</b>	22.4%	<b>34.9%</b>	<b>49.7%</b>	-	-	-
	LoRA (Mixed)	24.1%	22.3%	53.9%	<b>1.97%</b>	24.6%	26.6%	31.0%	44.3%	<b>3.0%</b>	18.3%	18.7%
	TaskWeaver (Ours)	<b>24.9%</b>	<b>23.9%</b>	<b>54.5%</b>	1.67%	24.8%	<b>28.6%</b>	29.5%	48.4%	1.0%	<b>19.7%</b>	<b>25.1%</b>
google/gemma-3-270m-it	Base	18.9%	23.0%	43.5%	4.55%	<b>24.7%</b>	24.6%	34.3%	50.6%	<b>19.3%</b>	18.4%	20.8%
	LoRA (Individual)	<b>25.6%</b>	<b>25.0%</b>	<b>56.4%</b>	3.64%	24.2%	26.4%	<b>42.9%</b>	<b>53.8%</b>	-	-	-
	LoRA (Mixed)	25.3%	21.3%	52.0%	3.41%	23.5%	25.2%	33.3%	50.9%	4.67%	20.6%	22.8%
	TaskWeaver (Ours)	21.1%	23.7%	45.7%	<b>5.84%</b>	<b>24.5%</b>	<b>27.6%</b>	34.7%	47.8%	16.3%	<b>21.5%</b>	<b>23.0%</b>
Qwen/Qwen3-0.6B	Base	21.7%	31.7%	63.6%	50.0%	25.9%	34.2%	42.4%	50.4%	73.0%	38.2%	34.4%
	LoRA (Individual)	50.7%	<b>71.8%</b>	<b>78.8%</b>	35.9%	<b>50.7%</b>	<b>62.2%</b>	<b>84.2%</b>	48.7%	-	-	-
	LoRA (Mixed)	<b>54.4%</b>	68.8%	68.6%	34.5%	33.0%	53.4%	76.9%	48.5%	52.7%	<b>50.9%</b>	<b>64.8%</b>
	TaskWeaver (Ours)	35.2%	48.5%	64.1%	<b>52.5%</b>	29.0%	38.8%	49.2%	<b>52.1%</b>	<b>74.7%</b>	45.9%	44.2%

\* Zero-shot evaluation (not seen during training). All values are accuracy (%). Bold = best per model.

ARC-C/E: ARC-Challenge/Easy, HSWAG: HellaSwag, OBQA: OpenBookQA, WG: Winogrande, CSQA: CommonsenseQA.

Table 1: Performance comparison across models and benchmarks. TaskWeaver excels on mathematical reasoning and zero-shot generalization, particularly for smaller models.

The plots for these can be found in Appendix B

## 5.4 Generated Weight Analysis

To understand what the hypernetwork learns, we visualize the generated LoRA weights using t-SNE. For each model, we extract generated weights for 20 samples per dataset and project to 2D (more details and analyses in Appendix. C). The T-SNE for Qwen3 0.6B is in Fig. 3. The T-SNE visualization of generated weights reveals:

**Semantic Clustering:** Weights cluster by task type—math (GSM8K, SVAMP), word-MCQ (SNLI, BoolQ), index-MCQ (ARC, OpenBookQA).

**Model Size Effect:** Larger models produce cleaner cluster separation.

**Interesting Pattern:** Winogrande and HellaSwag cluster between multiple choice tasks and math tasks. This is understandable since these are multiple choice tasks, but with numerical index choices.

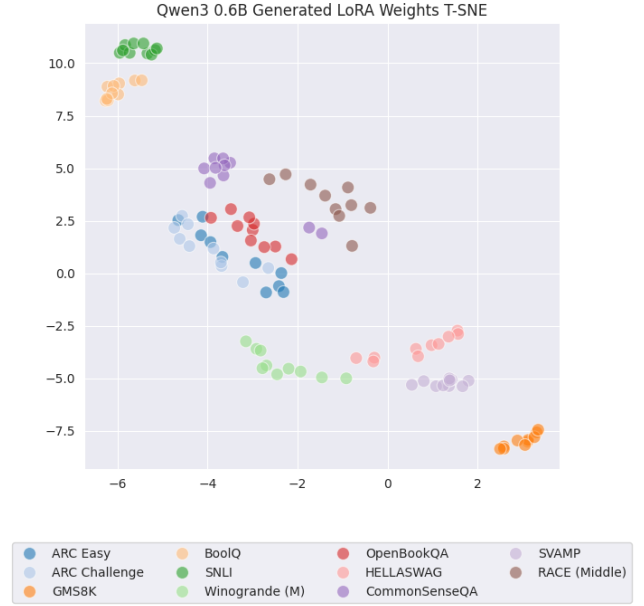


Figure 3: T-SNE visualization for Qwen3-0.6B, highlighting clustering

## 6 Code Overview

~3,000 lines across modules for the hypernetwork, finetuning on LoRA and evaluating models on selected datasets (GitHub: [dkapur17/taskweaver](https://github.com/dkapur17/taskweaver)):

### Core Hypernetwork

`src/hypernet/taskweaver.py`: TaskWeaver class with semantic projection, learnable embeddings, shared MLP and output heads. `replace_linears` dynamically swaps target linear layers with instances of `DynamicLoraLinear`, while preserving the learned weights. The module’s API is compliant with Hugging Face’s CausalLM API, allowing for direct use with `SFTTrainer`.

### Dynamic LoRA

`src/hypernet/dynamic_lora.py`: Implements `DynamicLoraLinear`, which allows injection of batch-specific LoRA weights at runtime through the `set_lora_parameters` method. The forward

method applies the injected instance-level LoRA matrices to each batch element using the einsum expression “ $bri, bor, bsi \rightarrow bso$ ”, where  $b$  is the batch size,  $r$  is the rank,  $i$  and  $o$  are the input and output dimensions of the linear layer respectively, and  $s$  is the sequence length.

### Custom Collator

`src/hypernet/collator.py`: Implements `DataCollatorWithPromptLengths`, which also returns prompt lengths (length of the sequence without the completion tokens), in addition to other inputs. The prompt length is used by TaskWeaver to mask out completion tokens for the hypernetwork during training, since the LoRA weights should only depend on the prompt tokens. During inference, only the prompt is provided, so the entire input is passed directly to the hypernetwork.

### Dataset Configurations

## 8 Research Log

**Gradient Explosion:** Initial training diverged, often resulting in very large gradients and nan loss. To address this, we use Bias-HyperInit Beck et al. [2022] which initializes the linear output head of the hypernetwork such that the weights are all zeros and the bias matches the initialization of the underlying layers. In our work, this corresponds to the output bias of the hypernetwork being initialized to  $U\left(-\frac{1}{\sqrt{2d}}, \frac{1}{\sqrt{2d}}\right)$  for the  $A$  head and all zero for the  $B$  head to match the initialization of tradition LoRA, similar to the  $M$  model in Charakorn et al. [2025]. This helps curb large gradients compared to fully random initialization of the weights and greatly stabilizes training, being extremely effective against the problem of exploding gradients. However, since the weight prediction requires several passes through the shared output heads, it is natural to see relatively high gradient norms during training. But if the number of gradient accumulation steps are too high, the resulting gradient norm could diverge to infinity and training will fail. As a result, we were conservative with the number of gradient accumulation steps for training to be successful.

**Batch-Wise LoRA:** Critical challenge — Text-to-LoRA uses uniform task descriptions per batch, but instance-level needs different weights per element. Solution: predict and apply  $b \times r \times d$  and  $b \times d \times r$  LoRA matrices through `DynamicLoraLinear`, instead of the traditional  $r \times d$  and  $d \times r$  matrices. For simplicity and clarity, we use einops to perform the batch-aware LoRA matrix multiplication.

**Self-Referential Design:** We originally planned to use a dedicated encoder model such as GTE-Large [Li et al., 2023] or ModernBERT [Warner et al., 2024], since extracting a semantic embedding from the input prompt is well suited for encoder-only models. However, this adds an additional layer of complexity, due to the fact that we would have to work with two different tokenizers, and this would make it exceptionally tricky to train the system using Hugging Face’s `SFTTrainer`.

**Evaluation Complexity:** Since evaluation is a key part of our project, we require a robust evaluation suite to work with. We initially relied on Eleuther AI’s LM Evaluation Harness [Gao et al., 2024] for standardized evaluation. However, its tightly coupled abstractions and limited transparency made it difficult to diagnose errors related to Qwen chat template formatting and prompt construction. In particular, the interaction between system prompts, role tokens, and generation boundaries was opaque, complicating debugging and ablation. We then tried using the [DeepEval from Confident AI](#) since they supported all of our tar-

## 7 Timeline and Effort

Activity	Hours
Literature review	15
Environment setup	2
DynamicLoraLinear development	2
Hypernetwork implementation	8
LoRA Finetuner implementation	2
Dataset Configs implementation	15
Evaluation framework	12
Baseline training (LoRA)	6
TaskWeaver training	8
Evaluation runs	20
Analysis (t-SNE, scaling)	4
Writing (reports, poster)	20
<b>Total hours</b>	<b>~114</b>
<b>Total hours (per person avg.)</b>	<b>~38</b>

get datasets out of the box with quite a transparent interface. However, this framework is reliant on G-Eval, which is an LLM as a judge evaluation method requiring API keys to a powerful model such as ChatGPT or Claude. We finally decided to implement our own lightweight custom evaluation framework that provides explicit control over prompt assembly, tokenization, and generation parameters, enabling reproducible evaluation across the different model types, and a far easier time debugging.

**Scope Adjustment:** We exclude code generation benchmarks such as Chen et al. [2021]HumanEval and Austin et al. [2021]MBPP from our evaluation. Beyond their computational cost, these benchmarks require executing model-generated code within a robust security sandbox to ensure safe and correct evaluation. Building and validating such an execution environment introduces significant engineering overhead and complexity, which is orthogonal to the goals of this work. We therefore focus on non-executable language tasks that more directly evaluate the semantic and instructional capabilities of the generated LoRA adapters.

## 9 Conclusion

We presented TaskWeaver, a hypernetwork architecture for instance-level test-time adaptation of language models. By generating unique LoRA weights for each input based solely on semantics, TaskWeaver eliminates the need for task labels or descriptions while achieving competitive performance with task-specific fine-tuning.

**Key findings:** (1) competitive with task-specific LoRA (2) superior resiliency to task skew in training data and zero-shot generalization compared to mixed-task LoRA; (3) generated weights form semantically meaningful clusters, even for zero-shot tasks.

**Limitations:** We see performance gaps on larger models, perhaps due to the limited expressiveness of the simple MLP backbone. Training data is skewed towards language understanding and knowledge retrieval, and there are far fewer math, logic and reasoning tasks.

**Future Work:** Modify the hypernetwork to use a more sophisticated backbone architecture, such as a transformer. We also see an application of this work in context-heavy tasks such as few-shot prompting or Retrieval Augmented Generation, where instead of flooding the model’s context with this information, we instead pass the content to TaskWeaver and have it generate LoRA weights that encode the content in them, reducing the pressure on the language model’s context window. We would also like to experiment with different model types, seeing TaskWeaver’s influence on rea-

soning model, Mixture-of-Expert models and perhaps the newer flavor of diffusion language models.

## 10 Thought-Experiment on Compute

**Actual Usage:** ~50 GPU-hours total. 20h on A100 (~\$40), 30h RTX 3090 (~\$15), plus local 3070/M4 Pro. Estimated cost: \$50-60.

**With Additional \$450:** (1) Scale to Qwen 1.7B/7B (\$250), (2) hyperparameter sweeps (\$100), (3) additional zero-shot benchmarks (\$50). This would validate scaling behavior and strengthen conclusions, (4) purchase LLM API credits for use as an evaluation judge to evaluate free-form output tasks (\$50).

## References

- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL <https://arxiv.org/abs/2108.07732>.
- Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning, 2022. URL <https://arxiv.org/abs/2210.11348>.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL <https://aclanthology.org/D15-1075>.
- Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=zWskCdu3QA>.



- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL <https://arxiv.org/abs/2107.03374>.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, and Tom Kwiatkowski. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- David Ha, Andrew M Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=rkpACellx>.
- 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>.
- Hamish Ivison and Matthew E. Peters. Hyperdecoders: Instance-specific decoders for multi-task nlp, 2022. URL <https://arxiv.org/abs/2203.08304>.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL <https://aclanthology.org/D17-1082>.
- Chenglin Li, Qianglong Chen, Liangyue Li, Caiyu Wang, Yicheng Li, Zulong Chen, and Yin Zhang. Mixed distillation helps smaller language model better reasoning, 2024. URL <https://arxiv.org/abs/2312.10730>.
- Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.
- Rabeeh 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)*, pages 565–576. Association for Computational Linguistics, 2021.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In

551 *Proceedings of the 2018 Conference on Empirical*  
552 *Methods in Natural Language Processing*, 2018.

553 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhaga-  
554 vatula, and Yejin Choi. Winogrande: An adversarial  
555 winograd schema challenge at scale. In *Proceedings*  
556 *of the AAAI Conference on Artificial Intelligence*,  
557 volume 34, pages 8732–8740, 2020.

558 Alon Talmor, Jonathan Herzig, Nicholas Lourie, and  
559 Jonathan Berant. Commonsenseqa: A question an-  
560 swering challenge targeting commonsense knowl-  
561 edge, 2019. URL [https://arxiv.org/abs/](https://arxiv.org/abs/1811.00937)  
562 [1811.00937](https://arxiv.org/abs/1811.00937).

563 Benjamin Warner, Antoine Chaffin, Benjamin Clavié,  
564 Orion Weller, Oskar Hallström, Said Taghadouini,  
565 Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom  
566 Aarsen, Nathan Cooper, Griffin Adams, Jeremy  
567 Howard, and Iacopo Poli. Smarter, better, faster,  
568 longer: A modern bidirectional encoder for fast,  
569 memory efficient, and long context finetuning and  
570 inference, 2024. URL [https://arxiv.org/](https://arxiv.org/abs/2412.13663)  
571 [abs/2412.13663](https://arxiv.org/abs/2412.13663).

572 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali  
573 Farhadi, and Yejin Choi. Hellaswag: Can a machine  
574 really finish your sentence? In *Proceedings of the*  
575 *57th Annual Meeting of the Association for Compu-*  
576 *tational Linguistics*, 2019.



## A The TaskWeaver PyTorch Module

```
TaskWeaver(  
  (lm): Qwen3ForCausalLM(  
    (model): Qwen3Model(  
      (embed_tokens): Embedding(151936, 1024)  
      (layers): ModuleList(  
        (0-27): 28 x Qwen3DecoderLayer(  
          (self_attn): Qwen3Attention(  
            (q_proj): DynamicLoraLinear(in_features=1024, out_features=2048, bias  
=False, lora_rank=2, lora_scaling=4.0, lora_dropout=0.01)  
            (k_proj): Linear(in_features=1024, out_features=1024, bias=False)  
            (v_proj): DynamicLoraLinear(in_features=1024, out_features=1024, bias  
=False, lora_rank=2, lora_scaling=4.0, lora_dropout=0.01)  
            (o_proj): Linear(in_features=2048, out_features=1024, bias=False)  
            (q_norm): Qwen3RMSNorm((128,), eps=1e-06)  
            (k_norm): Qwen3RMSNorm((128,), eps=1e-06)  
          )  
          (mlp): Qwen3MLP(  
            (gate_proj): Linear(in_features=1024, out_features=3072, bias=False)  
            (up_proj): Linear(in_features=1024, out_features=3072, bias=False)  
            (down_proj): Linear(in_features=3072, out_features=1024, bias=False)  
            (act_fn): SiLUActivation()  
          )  
          (input_layernorm): Qwen3RMSNorm((1024,), eps=1e-06)  
          (post_attention_layernorm): Qwen3RMSNorm((1024,), eps=1e-06)  
        )  
      )  
      (norm): Qwen3RMSNorm((1024,), eps=1e-06)  
      (rotary_emb): Qwen3RotaryEmbedding()  
    )  
    (lm_head): Linear(in_features=1024, out_features=151936, bias=False)  
  )  
  (semantic_proj): Linear(in_features=1024, out_features=1024, bias=True)  
  (module_embedding): Embedding(2, 1024)  
  (matrix_embedding): Embedding(2, 1024)  
  (layer_embedding): Embedding(28, 1024)  
  (mlp): Sequential(  
    (0): Linear(in_features=1024, out_features=1024, bias=True)  
    (1): GELU(approximate='none')  
    (2): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)  
    (3): Linear(in_features=1024, out_features=1024, bias=True)  
    (4): GELU(approximate='none')  
    (5): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)  
  )  
  (heads): ModuleDict(  
    (q_proj): ModuleDict(  
      (A): Linear(in_features=1024, out_features=2048, bias=True)  
      (B): Linear(in_features=1024, out_features=4096, bias=True)  
    )  
    (v_proj): ModuleDict(  
      (A): Linear(in_features=1024, out_features=2048, bias=True)  
      (B): Linear(in_features=1024, out_features=2048, bias=True)  
    )  
  )  
)
```

Listing 1: An example of the TaskWeaver Python Module

Listing 1 shows the PyTorch module of TaskWeaver with Qwen/Qwen3-0.6B. TaskWeaver.lm is the reference to the base language model. The target modules inside the base language model, in

636 this case `TaskWeaver.lm.model.layers.*.self_attn.{q_proj, v_proj}`, are replaced with a  
 637 `DynamicLoraLinear` instance. Note that `k_proj` and `o_proj` are still `Linear` layers because they aren't  
 638 part of the target modules list.

639 We can also see the `semantic_proj`, `module_embedding` (2 embeddings, `q_proj` and `v_proj`),  
 640 `matrix_embedding` (2 embeddings, `A` and `B`) and `layer_embedding` (28 embeddings, since  
 641 Qwen3-0.6B has 28 transformer blocks, shared mlp and the module-matrix specific heads.

## 642 B Scaling Analysis Details

643 Training dynamics across configurations show consistent patterns:

- 644 • **Hidden Dimension (128–2048):** Larger dimensions reduce training loss across all models. Qwen3-0.6B  
 645 shows less sensitivity than Pythia-70M, suggesting larger models are more robust to hypernetwork capacity  
 646 choices.
- 647 • **LoRA Rank (1–32):** Higher ranks improve training loss with diminishing returns beyond  $r = 8$ . Larger  
 648 models benefit more from increased rank.
- 649 • **LoRA Alpha ( $0.5\times$ – $16\times$  rank):** Higher alpha values reduce loss. Larger models show smoother response  
 650 curves.
- 651 • **Training Curves:** Qwen3 converges to loss  $\approx 0.5$  within 200 steps; Pythia plateaus at  $\approx 2.0$  after 400 steps.

## 652 C T-SNE Visualization

653 We extract generated LoRA weights for 20 samples per dataset and apply t-SNE:

654 **Cluster Groups Identified:**

- 655 • **Math:** GSM8K, SVAMP (numerical outputs)
- 656 • **Word-MCQ:** SNLI, BoolQ (word-based answers: “entailment”, “yes/no”)
- 657 • **Index-MCQ:** ARC, OpenBookQA, CommonsenseQA (letter indices: A/B/C/D)
- 658 • **Hybrid:** Winogrande, HellaSwag cluster near math (numeric indices: 1/2)

659 **Model Size Effect:** Pythia-70M shows overlapping clusters; Gemma-270M shows emerging separation;  
 660 Qwen3-0.6B produces distinct, well-separated clusters.

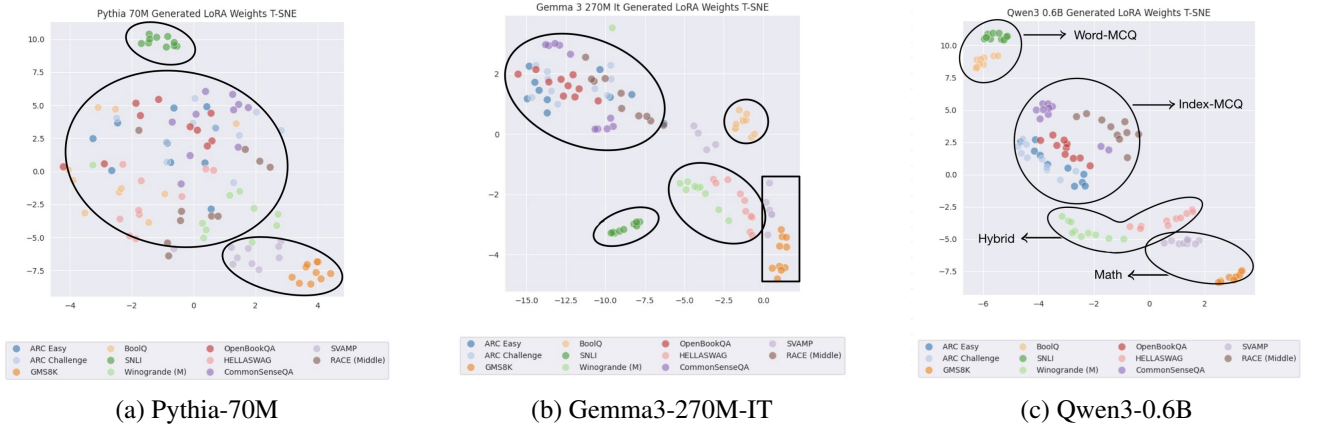


Figure 4: T-SNE visualizations for the 3 backbones of TaskWeaver, with Pythia-70M having an amorphous blob of predicted LoRAs, while Qwen3-0.6B has clear-cut distinctions in the clustering as annotated in Fig. 4c

## 661 D Qualitative Examples

Task / Model	Prompt & Outputs
<b>Instruction Following</b> <i>Pythia-70M</i>	<p>Prompt: Choose the most reasonable answer. Question: What suggested the presence of a planet outside our solar system? A. radio wave emissions B. a wobble...</p> <p>Base: Incoherent repetition</p> <p>TaskWeaver: B (Correct format)</p>
<b>Math Reasoning</b> <i>Qwen3-0.6B</i>	<p>Prompt: Johnny has 500 legos, another box with <math>3\times</math> more, and another with <math>1/4</math> the number. Total?</p> <p>Base: Computes third box as <math>\frac{1}{4} \times 1500 = 375</math> (wrong reference)</p> <p>TaskWeaver: Correctly computes <math>\frac{1}{4} \times 500 = 125</math>, total = 2125 (Correct)</p>

Table 2: **Qualitative examples.** TaskWeaver improves instruction following for small models (Pythia) and mathematical reasoning for larger models (Qwen).

Table 3: **Benchmark datasets and example outputs.** Training benchmarks (top) cover diverse reasoning types. Zero-shot benchmarks (bottom, marked with \*) test generalization to unseen tasks. The base Gemma model failed on each of these examples, while the TaskWeaver produced correct outputs

Dataset	Description & Example
<b>GSM8K</b> ( <i>Math</i> )	<p>Grade-school math word problems requiring 2–8 step arithmetic reasoning. Tests decomposition and numerical computation.</p> <p>Prompt: Analyze the given math problem, reason through it step by step, and provide the final answer in a new line starting with ####, for example: #### 72: <i>Richard lives in an apartment building with 15 floors. Each floor contains 8 units, and <math>3/4</math> of the building is occupied. What’s the total number of unoccupied units In the building?</i></p> <p>Gemma3 (Base) : Let the total number of floors in the apartment building be <math>F</math>. Let the number of floors in the building be <math>F</math>. The number of units in each floor is 8. So, the total number of units in the building is <math>F \times 8 = 8F</math>. The number of units occupied is <math>\frac{3}{4}</math> of the total number of units. The number of units occupied is <math>\frac{3}{4} \times 8F = \frac{3 \times 8F}{4} = \frac{24F}{4} = 6F</math>. The number of unoccupied units is the total number of units minus the number of units occupied. Unoccupied units = <math>F - \text{occupied units} = F - 6F = F - 6F = -5F</math>. Since the number of unoccupied units is negative, we need to find a positive value for <math>F</math>. The number of unoccupied units is <math>-5F</math>. The final answer is <math>-5F</math>. Therefore, the total number of unoccupied units in the building is <math>-5F</math></p> <p>Gemma3 (TaskWeaver) : The total number of units in the building is 15 floors * 8 units/floor = 120 units. The number of units in the building is 15 * 8 = 120 units. The number of unoccupied units is the total number of units minus the number of occupied units, which is 120 - 90 = 30 units</p>
<b>ARC-Easy</b> ( <i>Science</i> )	<p>Grade-school science multiple-choice questions requiring reasoning beyond direct fact retrieval.</p> <p>Prompt: Choose the most reasonable answer for the question from the given options. Respond only with A, B, C or D: <i>Which piece of safety equipment is used to keep mold spores from entering the respiratory system? A. safety goggles B. breathing mask C. rubber gloves D. lead apron</i></p> <p>Gemma3 (Base) : A    Gemma3 (TaskWeaver) : B</p>

Continued on next page

Table 3 continued from previous page

Dataset	Description & Example
<b>ARC-Challenge</b> (Science)	<p>Harder subset of ARC requiring more complex scientific reasoning and world knowledge.</p> <p>Prompt: Choose the most reasonable answer for the question from the given options. Repond only with A, B, C or D: <i>Petrified palm trees are found in sedimentary rock near glaciers. The presence of the petrified palm trees most likely provides evidence for which statement?A.There was once more water in the area.B.The area was once grassland.C.The climate in the area was once tropical. D.There are active faults in the area.</i></p> <p>Gemma3 (Base) : A    Gemma3 (TaskWeaver) : C</p>
<b>BoolQ</b> (Reading)	<p>Yes/no questions from real Google searches paired with Wikipedia passages. Tests reading comprehension.</p> <p>Prompt: Use the content in the passage to answer the question with either true or false only: <i>Passage: Bloodline was announced in October 2014 as part of a partnership between Netflix and Sony Pictures Television, representing Netflix’s first major deal with a major film studio for a television series. The series was created and executive produced by Todd A. Kessler, Glenn Kessler, and Daniel Zelman, who previously created the FX series Damages. According to its official synopsis released by Netflix, Bloodline “centers on a close-knit family of four adult siblings whose secrets and scars are revealed when their black sheep brother returns home.”</i></p> <p>Gemma3 (Base) : True    Gemma3 (TaskWeaver) : False</p>
<b>SNLI</b> (NLI)	<p>Sentence pairs labeled entailment, contradiction, or neutral. Tests logical relationship understanding.</p> <p>Prompt: Given a premise and a hypothesis, determine the relationship between them. Respond with 'entailment' if the hypothesis follows from the premise, 'contradiction' if the hypothesis contradicts the premise, or 'neutral' if the relationship is undetermined: <i>Premise: Many people standing outside of a place talking to each other in front of a building that has a sign that says "HI-POINTE." Hypothesis: The people are having a chat before going into the work building.</i></p> <p>Gemma3 (Base) : entailment    Gemma3 (TaskWeaver) : neutral</p>
<b>OpenBookQA</b> (Reasoning)	<p>Multi-hop reasoning questions requiring combining background knowledge with given facts.</p> <p>Prompt: Choose the most reasonable answer for the question from the given options. Respond only with A, B, C or D: <i>There is most likely going to be fog around: A. a marsh B. a tundra C. the plains D. a desert</i></p> <p>Gemma3 (Base) : C    Gemma3 (TaskWeaver) : A</p>
<b>HellaSwag</b> (Commonsense)	<p>Sentence completion requiring commonsense reasoning about everyday situations.</p> <p>Prompt: Choose the most reasonable continuation from the given options. Respond only with 0, 1, 2 or 3: <i>High jump: A boy is running down a track. the boy 0. runs into a car. 1. gets in a mat. 2. lifts his body above the height of a pole. 3. stands on his hands and springs.</i></p> <p>Gemma3 (Base) : 3    Gemma3 (TaskWeaver) : 2</p>

Continued on next page

Table 3 continued from previous page

Dataset	Description & Example
<b>Winogrande</b> (Coreference)	<p>Pronoun resolution requiring world knowledge to disambiguate references.</p> <p>Prompt: Fill in the blank with the correct option. Respond only with 1 or 2:  <i>All the clutter in the house excited Leslie but not Derrick because cleaning energized - very much. 1. Leslie 2. Derrick</i></p> <p>Gemma3 (Base) : 2. Derrick    Gemma3 (TaskWeaver) : 1</p>

*Continued on next page*

Table 3 continued from previous page

Dataset	Description & Example
<i>Zero-Shot Evaluation (not seen during training)</i>	
<b>SVAMP*</b> (Math)	<p>Math word problems testing generalization from GSM8K to different problem structures.</p> <p>Prompt: Solve the math word problem step by step and provide the final numerical answer: <i>Ed had 10 more marbles than Doug. Doug lost 11 of his marbles at the playground. If Ed had 45 marbles How many more marbles did Ed have than Doug then?</i></p> <p>Gemma3 (Base) : Let <math>d</math> be the number of marbles Ed had initially. Let <math>y</math> be the number of marbles Doug had initially. We are given that Ed had 10 more marbles than Doug, so <math>d = y + 10</math>. We are also given that Doug lost 11 marbles, so <math>y - 11 = 0</math>. Therefore, <math>y = 11</math>. Now we can find the number of marbles Ed had: <math>d = y + 10 = 11 + 10 = 21</math>. So Ed had 21 marbles. We want to find how many more marbles Ed had than Doug, which is <math>d - y</math>. <math>d - y = 21 - 11 = 10</math>.</p> <p>Gemma3 (TaskWeaver) : Let <math>d</math> be the number of marbles Ed had. Let <math>g</math> be the number of marbles Doug had. We are given that Ed had 10 more marbles than Doug, so <math>d = g + 10</math>. We are also given that Doug lost 11 marbles, so <math>g - d = 11</math>. We have the system of equations: <math>d = g + 10</math> <math>g - d = 11</math> From the first equation, we have <math>g = d + 10</math>. Substituting this into the second equation, we get: <math>g - (d + 10) = 11</math> <math>g - d - 10 = 11</math> <math>g - d - 10 = 11</math> <math>g - d = 21</math> Now we have <math>d = g + 10</math> and <math>g - d = 21</math>. We can substitute the first equation into the second equation: <math>g - (g + 10) = 21</math> <math>g - g - 10 = 21 - 10 = 21</math></p>
<b>Commonsense- QA*</b> (Reasoning)	<p>5-way multiple choice questions requiring commonsense knowledge about concepts.</p> <p>Prompt: Answer the commonsense reasoning question by choosing the most appropriate option. Respond only with A, B, C, D, or E: <i>What island country is ferret popular? A. own home B. north carolina C. great britain D. hutch E. outdoors</i></p> <p>Gemma3 (Base) : A    Gemma3 (TaskWeaver) : C</p>

Continued on next page

Table 3 continued from previous page

Dataset	Description & Example
<b>RACE-Middle*</b> (Reading)	<p>Reading comprehension from middle school English exams. Tests passage understanding.</p> <p>Prompt: Read the article and answer the question by choosing the most appropriate option. Respond only with A, B, C, or D: Article: <i>When my wife left this world, I chose to travel in Antigua looking for a peaceful place to rest my old body. Not quite old and weak, I felt I wanted something more than the usual hotel room with 24-hour room service. I decided this year to try something completely new and booked myself a private holiday home in Antigua. This was the best decision I had ever made, as there was plenty to do, plenty to see and lots of lovely restaurants to visit. There was a private swimming pool, and a cool, wide yard where I ate my breakfast most mornings. Antigua has to be one of the loveliest places on earth to spend a holiday. The bright blue sea and the endless blue around the beach areas proved to be an excellent place for me to spend the long afternoons. I had to hurry to do what I wanted to do before the holiday came to an end. I managed to visit the Sugar Mill and Shirley Heights on my last two days and yet found myself wondering whether I could extend for a few more days. I rented a boat and came home after a day’s sailing, refreshed, looking forward to dinner. Everything is so pleasant on these beautiful islands, swept by the trade winds and warmed by the sun for so many summer months. The food just tasted better to me, perhaps because I was having such a great holiday. There was always someone to have a drink with—that’s what I liked most.</i></p> <p>Question: The most important reason for the writer to travel in Antigua was . . . A. there was bright blue sea B. there was endless blue sky C. there were lots of lovely restaurants D. there were many people to drink with</p> <p>Gemma3 (Base) : A    Gemma3 (TaskWeaver) : D</p>

## E Hyperparameter Configuration

662

Parameter	Pythia 70M	Gemma3 270M	Qwen3 0.6B
Hidden dimension	1024	1024	1024
LoRA rank	2	2	2
LoRA alpha	8	8	8
Batch size	16	8	4
Learning rate	$10^{-6}$	$10^{-6}$	$10^{-6}$
Gradient accum.	2	2	2
Training steps	~1680	~1680	~1680
Trainable parameters %	8.8%	2.9%	2.2%

Table 4: Training configuration across model scales.