

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, rajmajes}@andrew.cmu.edu
10-423/623 Generative AI Course Project

December 11, 2025

1 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.

20 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 ℓ 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.

74 3 Related Work

75 **LoRA** [Hu et al., 2022] represents weight updates as
76 $\Delta W = BA$ with $r \ll \min(d, k)$, reducing parameters
77 by $10,000\times$ while matching full fine-tuning.

78 **Hypernetworks** [Ha et al., 2017] generate weights
79 for other networks. HyperFormer [Mahabadi et al.,
80 2021] extends this to transformers, generating adapter
81 parameters from task embeddings with 0.29% addi-
82 tional parameters per task.

83 **Meta-Learning.** MAML [Finn et al., 2017] en-
84 ables rapid adaptation via bi-level optimization but re-
85 quires gradient updates during adaptation; hypernet-
86 works generate parameters in one forward pass.

87 **Text-to-LoRA** [Charakorn et al., 2025] generates
88 complete LoRA adapters from task descriptions,
89 achieving 98% of specialized adapter performance.
90 HyperDecoders [Ivison and Peters, 2022] treat param-
91 eter generation as autoregressive sequence modeling.

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

95 4 Methods

96 4.1 Baseline: Trained LoRA Adapters

97 **LoRA (Individual):** Separate adapters trained inde-
98 pendently for each task. This represents the upper
99 bound of task-specific optimization but requires N
100 training runs for N tasks.

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

104 Both baselines use identical LoRA configuration:
105 rank $r = 2$, $\alpha = 8$, applied to query and value pro-
106 jections.

107 4.2 TaskWeaver Architecture

108 The architecture overview can be seen in Figure 1.

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

113 **Conditioning.** Following Charakorn et al. [2025],
114 we add learned embeddings to the semantic embed-
115 ding:

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

116 where e_{layer} , e_{module} , e_{matrix} are learned embeddings for
117 layer index, module type (query projection, value pro-
118 jection, 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 Ψ 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.

139 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 mod-
elling loss, (4) update only hypernetwork parameters.
Output heads are zero-initialized for small initial adap-
tations, and this is crucial for stable training (also men-
tioned by Charakorn et al. [2025]).

149 5 Experiments

150 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/Gem-
ma/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.

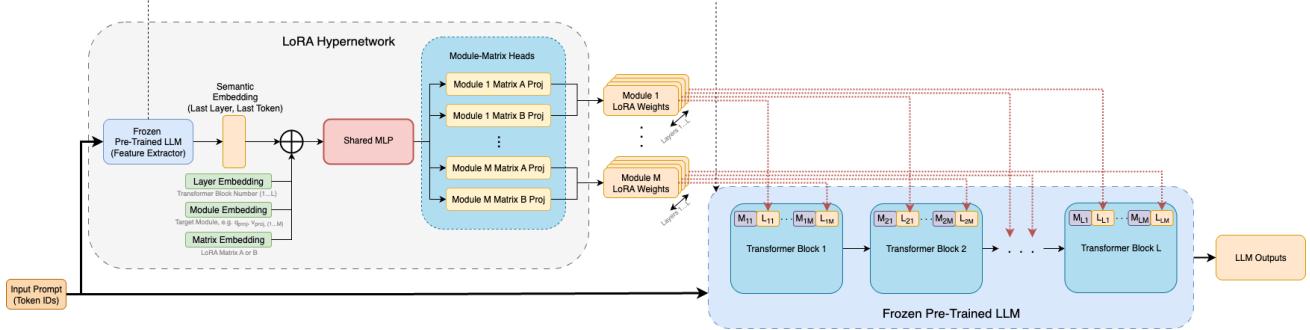


Figure 1: Overview of the TaskWeaver architecture. The hypernetwork (left) takes an input prompt and processes it through a frozen pre-trained LLM to extract a semantic embedding of the prompt (taken as the last layer representation of the last token). These embeddings are added to learned layer, module 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.

159 5.2 Main Results

160 Table 1 presents comprehensive results.

161 **Pythia-70M:** TaskWeaver achieves best performance on 6/11 benchmarks including both zero-shot 162 tasks, being consistently better than the base model, 163 while being a close second in cases where it didn’t have 164 the best performance. 165

166 Note how most performance numbers increase dras- 167 tically. This is because the base Pythia model is unable 168 to generate responses aligned with the question on its 169 own, and hence fails to even answer the prompt. Fine- 170 tuning helps align the model’s outputs to match each 171 task’s expectations, and leads to the improvement. An 172 example is provided in Appendix D Table. 2. The 173 performance numbers themselves are close to that of ran- 174 dom predictions, which is expected from such a small 175 model.

176 **Gemma3-270M-IT:** Competitive performance with 177 best GSM8K (5.84%) and strong zero-shot results. The 178 instruction-tuned base provides a higher starting point.

179 **Qwen3-0.6B:** Most interesting patterns. While 180 LoRA (Individual) wins on in-distribution tasks, 181 TaskWeaver excels on GSM8K (**52.5%** vs 34.5% 182 Mixed) and SVAMP (**74.7%** vs 52.7%). This demon- 183 strates robustness to training data skew. Further, we 184 notice that in some cases, LoRA finetuning hinders 185 the base model’s performance (GSM8K 50% → 35%, 186 SVAMP 73% → 53.7%), but TaskWeaver always im- 187 proves upon the base model, preserving its original ca- 188 pabilities.

189 **Key Findings:** TaskWeaver consistently outper- 190 forms the base model, or is close enough to it, indic- 191 ating that it builds on top of, without hindering, the 192 base model’s capabilities. We also see that TaskWeaver

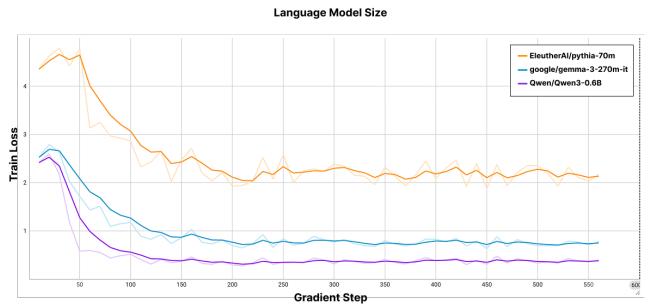


Figure 2: Train Loss vs Model Size

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).

200 5.3 Scaling Analysis

201 We analyze hyperparameter effects on training:

202 **Model Size:** Larger models show smoother training, 203 faster convergence, lower final loss. Qwen converges to 204 ∼0.5 loss vs Pythia’s ∼2.0 (Figure 2). We also observe 205 that larger models are more invariant to hyperparameter 206 changes.

207 **Hidden Dimension:** Increasing the hidden dimen- 208 sion of the hypernetwork, we see training loss decrease 209 monotonically.

210 **LoRA Rank:** Increasing the rank of the predicted 211 adapters, we see training loss decrease monotonically.

212 **LoRA Alpha:** Increasing the alpha of the predicted 213 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%	25.4%	22.4%	34.9%	49.7%	-	-	-
	LoRA (Mixed)	24.1%	22.3%	53.9%	1.97%	24.6%	26.6%	31.0%	44.3%	3.0%	18.3%	18.7%
	TaskWeaver (Ours)	24.9%	23.9%	54.5%	1.67%	24.8%	28.6%	29.5%	48.4%	1.0%	19.7%	25.1%
google/gemma-3-270m-it	Base	18.9%	23.0%	43.5%	4.55%	24.7%	24.6%	34.3%	50.6%	19.3%	18.4%	20.8%
	LoRA (Individual)	25.6%	25.0%	56.4%	3.64%	24.2%	26.4%	42.9%	53.8%	-	-	-
	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%	5.84%	24.5%	27.6%	34.7%	47.8%	16.3%	21.5%	23.0%
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%	71.8%	78.8%	35.9%	50.7%	62.2%	84.2%	48.7%	-	-	-
	LoRA (Mixed)	54.4%	68.8%	68.6%	34.5%	33.0%	53.4%	76.9%	48.5%	52.7%	50.9%	64.8%
	TaskWeaver (Ours)	35.2%	48.5%	64.1%	52.5%	29.0%	38.8%	49.2%	52.1%	74.7%	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.

214 The plots for these can be found in Appendix B

215 5.4 Generated Weight Analysis

216 To understand what the hypernetwork learns, we vi-
217 sualize the generated LoRA weights using t-SNE. For
218 each model, we extract generated weights for 20 sam-
219 ples per dataset and project to 2D (more details and
220 analyses in Appendix. C). The T-SNE for Qwen3 0.6B
221 is in Fig. 3. The T-SNE visualization of generated
222 weights reveals:

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

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

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

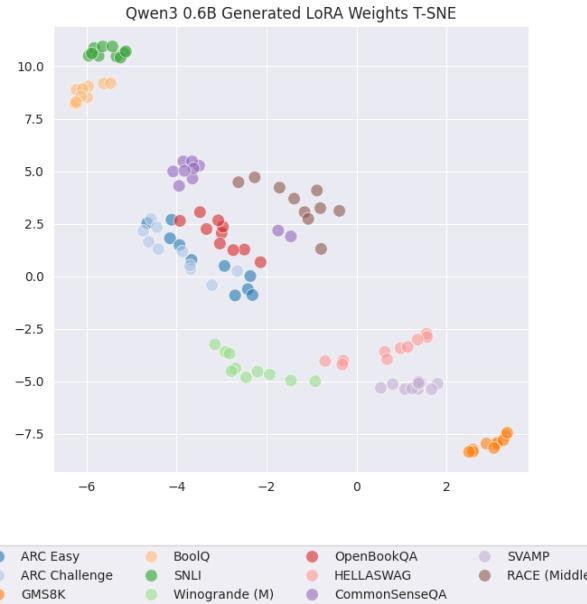


Figure 3: T-SNE visualization for Qwen3-0.6B, highlighting clustering

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

255 Custom Collator

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

265 Dataset Configurations

6 Code Overview

232 ~3,000 lines across modules for the hypernetwork,
233 finetuning on LoRA and evaluating models on selected
234 datasets (GitHub: [dkapur17/taskweaver](https://github.com/dkapur17/taskweaver)):

235 Core Hypernetwork

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

243 Dynamic LoRA

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

268 `src/dsconf/dataset_configs.py`: Implements
 269 a registry for datasets we use in this work,
 270 unifying them under the `DatasetConfig` API.
 271 Each dataset is configured with a chat and non-chat
 272 processor function that converts the dataset’s columns
 273 into prompt and completion columns for use with
 274 SFTTrainer. A central registry to ensure all training
 275 and evaluation scripts use the same data format. Lines
 276 596-747 implement `DatasetMixer`, used to mix
 277 multiple datasets together while still exposing the
 278 same API as individual datasets.

279 LoRA Finetuner

280 `src/lora/lora_finetuner.py`: Implements a
 281 high-level LoRA finetuner class that leverages the
 282 `DatasetConfig` API.

283 Evaluator

284 `src/evaluate.py`: Parses input configurations in-
 285 cluding base, LoRA and TaskWeaver, locates corre-
 286 sponding model weights, and runs evaluations over in-
 287 put datasets. Saves all outputs along with their parsed
 288 and expected predictions and final results.

289 `src/reevaluate.py`: Helps re-evaluate outputs of
 290 `evaluate.py` when evaluation metrics are updated.

291 `src/eval/`: Implements the actual interface with
 292 TaskWeaver, obtains dataset hooks, generates outputs
 293 and evaluates them with various metrics.

294 Training Scripts

295 `src/train_hypernet.py` and
 296 `src/train_lora.py` implement flexible train-
 297 ing with for TaskWeaver and baseline LoRA on
 298 multiple datasets simultaneously.

8 Research Log

Gradient Explosion: Initial training diverged, often
 301 resulting in very large gradients and nan loss. To ad-
 302 dress this, we use Bias-HyperInit Beck et al. [2022]
 303 which initializes the linear output head of the hyper-
 304 network such that the weights are all zeros and the bias
 305 matches the initialization of the underlying layers. In
 306 our work, this corresponds to the output bias of the hy-
 307 pernetwork being initialized to $U\left(-\frac{1}{\sqrt{2d}}, \frac{1}{\sqrt{2d}}\right)$ for the
 308 A head and all zero for the B head to match the ini-
 309 tialization of tradition LoRA, similar to the M model
 310 in Charakorn et al. [2025]. This helps curb large gra-
 311 dients compared to fully random initialization of the
 312 weights and greatly stabilizes training, being extremely
 313 effective against the problem of exploding gradients.
 314 However, since the weight prediction requires several
 315 passes through the shared output heads, it is natural to
 316 see relatively high gradient norms during training. But
 317 if the number of gradient accumulation steps are too
 318 high, the resulting gradient norm could diverge to infin-
 319 ity and training will fail. As a result, we were conser-
 320 vative with the number of gradient accumulation steps
 321 for training to be successful.

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

Self-Referential Design: We originally planned to
 331 use a dedicated encoder model such as GTE-Large [Li
 332 et al., 2023] or ModernBERT [Warner et al., 2024],
 333 since extracting a semantic embedding from the input
 334 prompt is well suited for encoder-only models. How-
 335 ever, this adds an additional layer of complexity, due
 336 to the fact that we would have to work with two dif-
 337 ferent tokenizers, and this would make it exception-
 338 ally tricky to train the system using Hugging Face’s
 339 SFTTrainer.

Evaluation Complexity: Since evaluation is a key
 341 part of our project, we require a robust evaluation suite
 342 to work with. We initially relied on Eleuther AI’s LM
 343 Evaluation Harness [Gao et al., 2024] for standard-
 344 ized evaluation. However, its tightly coupled abstrac-
 345 tions and limited transparency made it difficult to di-
 346 agnose errors related to Qwen chat template format-
 347 ting and prompt construction. In particular, the inter-
 348 action between system prompts, role tokens, and gen-
 349 eration boundaries was opaque, complicating debug-
 350 ging and ablation. We then tried using the `DeepEval`
 351 from Confident AI since they supported all of our tar-

299 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
Total hours	~114
Total hours (per person avg.)	~38

353 get datasets out of the box with quite a transparent interface. However, this framework is reliant on G-Eval,
354 which is an LLM as a judge evaluation method requiring API keys to a powerful model such as ChatGPT
355 or Claude. We finally decided to implement our own lightweight custom evaluation framework that provides
356 explicit control over prompt assembly, tokenization, and generation parameters, enabling reproducible eval-
357 uation across the different model types, and a far easier time debugging.
362

363 **Scope Adjustment:** We exclude code generation
364 benchmarks such as Chen et al. [2021]HumanEval and
365 Austin et al. [2021]MBPP from our evaluation. Be-
366 yond their computational cost, these benchmarks re-
367 quire executing model-generated code within a robust
368 security sandbox to ensure safe and correct evaluation.
369 Building and validating such an execution environment
370 introduces significant engineering overhead and com-
371 plexity, which is orthogonal to the goals of this work.
372 We therefore focus on non-executable language tasks
373 that more directly evaluate the semantic and instruc-
374 tional capabilities of the generated LoRA adapters.

375 9 Conclusion

376 We presented TaskWeaver, a hypernetwork architec-
377 ture for instance-level test-time adaptation of language
378 models. By generating unique LoRA weights for each
379 input based solely on semantics, TaskWeaver elimi-
380 nates the need for task labels or descriptions while
381 achieving competitive performance with task-specific
382 fine-tuning.

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

388 **Limitations:** We see performance gaps on larger
389 models, perhaps due to the limited expressiveness of
390 the simple MLP backbone. Training data is skewed
391 towards language understanding and knowledge re-
392 trieval, and there are far fewer math, logic and reason-
393 ing tasks.

394 **Future Work:** Modify the hypernetwork to use a
395 more sophisticated backbone architecture, such as a
396 transformer. We also see an application of this work in
397 context-heavy tasks such as few-shot prompting or Re-
398 trieval Augmented Generation, where instead of flood-
399 ing the model’s context with this information, we in-
400 stead pass the content to TaskWeaver and have it gen-
401 erate LoRA weights that encode the content in them,
402 reducing the pressure on the language model’s context
403 window. We would also like to experiment with differ-
404 ent model types, seeing TaskWeaver’s influence on re-

soning model, Mixture-of-Expert models and perhaps
405 the newer flavor of diffusion language models.
406

10 Thought-Experiment on Compute

407 **Actual Usage:** ~50 GPU-hours total. 20h on A100
408 (~\$40), 30h RTX 3090 (~\$15), plus local 3070/M4
409 Pro. Estimated cost: \$50-60.
410

411 **With Additional \$450:** (1) Scale to Qwen 1.7B/7B
412 (\$250), (2) hyperparameter sweeps (\$100), (3) addi-
413 tional zero-shot benchmarks (\$50). This would vali-
414 date scaling behavior and strengthen conclusions, (4)
415 purchase LLM API credits for use as an evaluation
416 judge to evaluate free-form output tasks (\$50).

References

417 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten
418 Bosma, Henryk Michalewski, David Dohan, Ellen
419 Jiang, Carrie Cai, Michael Terry, Quoc Le, and
420 Charles Sutton. Program synthesis with large lan-
421 guage models, 2021. URL <https://arxiv.org/abs/2108.07732>.
422

424 Jacob Beck, Matthew Thomas Jackson, Risto Vuorio,
425 and Shimon Whiteson. Hypernetworks in meta-
426 reinforcement learning, 2022. URL <https://arxiv.org/abs/2210.11348>.
427

428 Stella Biderman, Hailey Schoelkopf, Quentin Gregory
429 Anthony, Herbie Bradley, Kyle O’Brien, Eric Hal-
430 lahan, Mohammad Aflah Khan, Shivanshu Purohit,
431 USVSN Sai Prashanth, Edward Raff, et al. Pythia:
432 A suite for analyzing large language models across
433 training and scaling. In *International Conference on
434 Machine Learning*, pages 2397–2430. PMLR, 2023.
435

436 Samuel R. Bowman, Gabor Angeli, Christopher Potts,
437 and Christopher D. Manning. A large annotated
438 corpus for learning natural language inference. In
439 Lluís Màrquez, Chris Callison-Burch, and Jian Su,
440 editors, *Proceedings of the 2015 Conference on
441 Empirical Methods in Natural Language Process-
442 ing*, pages 632–642, Lisbon, Portugal, September
443 2015. Association for Computational Linguistics.
444 doi: 10.18653/v1/D15-1075. URL <https://aclanthology.org/D15-1075>.
445

446 Rujikorn Charakorn, Edoardo Cetin, Yujin Tang,
447 and Robert Tjarko Lange. Text-to-LoRA: In-
448 stant transformer adaption. In *Forty-second
449 International Conference on Machine Learning*,
450 2025. URL <https://openreview.net/forum?id=zWskCdu3QA>.
451

- 451 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming
452 Yuan, Henrique Ponde de Oliveira Pinto, Jared Ka-
453 plan, Harri Edwards, Yuri Burda, Nicholas Joseph,
454 Greg Brockman, Alex Ray, Raul Puri, Gretchen
455 Krueger, Michael Petrov, Heidy Khlaaf, Girish Sas-
456 try, Pamela Mishkin, Brooke Chan, Scott Gray,
457 Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz
458 Kaiser, Mohammad Bavarian, Clemens Winter,
459 Philippe Tillet, Felipe Petroski Such, Dave Cum-
460 mings, Matthias Plappert, Fotios Chantzis, Eliza-
461 beth Barnes, Ariel Herbert-Voss, William Hebgen
462 Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie
463 Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain,
464 William Saunders, Christopher Hesse, Andrew N.
465 Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan
466 Morikawa, Alec Radford, Matthew Knight, Miles
467 Brundage, Mira Murati, Katie Mayer, Peter Welin-
468 der, Bob McGrew, Dario Amodei, Sam McCandlish,
469 Ilya Sutskever, and Wojciech Zaremba. Evaluating
470 large language models trained on code, 2021. URL
471 <https://arxiv.org/abs/2107.03374>. 502
- 472 Christopher Clark, Kenton Lee, Ming-Wei Chang, and
473 Tom Kwiatkowski. Boolq: Exploring the surprising
474 difficulty of natural yes/no questions. In *Proceed-
475 ings of the 2019 Conference of the North American
476 Chapter of the Association for Computational Lin-
477 guistics*, 2019. 503
- 478 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot,
479 Ashish Sabharwal, Carissa Schoenick, and Oyvind
480 Tafjord. Think you have solved question answering?
481 try arc, the ai2 reasoning challenge. In *Proceed-
482 ings of the 2018 Conference on Empirical Methods
483 in Natural Language Processing*, 2018. 504
- 484 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian,
485 Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias
486 Plappert, Jerry Tworek, Jacob Hilton, Reiichiro
487 Nakano, Christopher Hesse, and John Schulman.
488 Training verifiers to solve math word problems.
489 *arXiv preprint arXiv:2110.14168*, 2021. 505
- 490 Chelsea Finn, Pieter Abbeel, and Sergey Levine.
491 Model-agnostic meta-learning for fast adaptation of
492 deep networks. In *Proceedings of the 34th Inter-
493 national Conference on Machine Learning*, volume 70
494 of *Proceedings of Machine Learning Re-
495 search*, pages 1126–1135. PMLR, 2017. 506
- 496 Leo Gao, Jonathan Tow, Baber Abbasi, Stella Bider-
497 man, Sid Black, Anthony DiPofi, Charles Foster,
498 Laurence Golding, Jeffrey Hsu, Alain Le Noac'h,
499 Haonan Li, Kyle McDonell, Niklas Muennighoff,
500 Chris Ociepa, Jason Phang, Laria Reynolds, Hai-
ley Schoelkopf, Aviya Skowron, Lintang Sutawika,
501 Eric Tang, Anish Thite, Ben Wang, Kevin Wang,
502 and Andy Zou. The language model evaluation har-
503 ness, 07 2024. URL <https://zenodo.org/records/12608602>. 504
- 506 David Ha, Andrew M Dai, and Quoc V Le. Hyper-
507 networks. In *International Conference on Learn-
508 ing Representations*, 2017. URL <https://openreview.net/forum?id=rkpACe1lx>. 509
- 510 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan
511 Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
512 and Weizhu Chen. LoRA: Low-rank adapta-
513 tion of large language models. In *Inter-
514 national Conference on Learning Representations*,
515 2022. URL <https://openreview.net/forum?id=nZeVKeefYf9>. 516
- 517 Hamish Ivison and Matthew E. Peters. Hyperde-
518 coders: Instance-specific decoders for multi-task
519 nlp, 2022. URL <https://arxiv.org/abs/2203.08304>. 520
- 521 Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang,
522 and Eduard Hovy. RACE: Large-scale ReAd-
523 ing comprehension dataset from examinations. In
524 *Proceedings of the 2017 Conference on Empir-
525 ical Methods in Natural Language Processing*,
526 pages 785–794, Copenhagen, Denmark, Septem-
527 ber 2017. Association for Computational Linguis-
528 tics. doi: 10.18653/v1/D17-1082. URL <https://aclanthology.org/D17-1082>. 529
- 530 Chenglin Li, Qianglong Chen, Liangyue Li, Caiyu
531 Wang, Yicheng Li, Zulong Chen, and Yin Zhang.
532 Mixed distillation helps smaller language model bet-
533 ter reasoning, 2024. URL <https://arxiv.org/abs/2312.10730>. 534
- 535 Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long,
536 Pengjun Xie, and Meishan Zhang. Towards general
537 text embeddings with multi-stage contrastive learn-
538 ing. *arXiv preprint arXiv:2308.03281*, 2023. 539
- 540 Rabeeh Mahabadi, Sebastian Ruder, Mostafa De-
541 ghani, and James Henderson. Parameter-efficient
542 multi-task fine-tuning for transformers via shared
543 hypernetworks. In *Proceedings of the 59th Annual
544 Meeting of the Association for Computational Lin-
545 guistics and the 11th International Joint Conference
546 on Natural Language Processing (Volume 1: Long
547 Papers)*, pages 565–576. Association for Computa-
548 tional Linguistics, 2021. 549
- 548 Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish
549 Sabharwal. Can a suit of armor conduct electricity?
550 a new dataset for open book question answering. In
550

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/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/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

577

```

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

634

635

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:

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

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

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

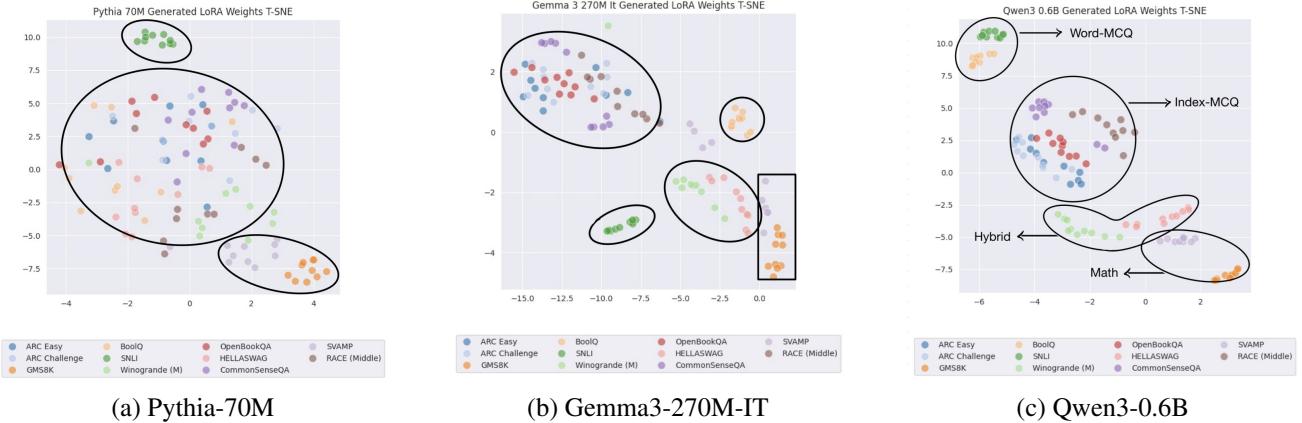


Figure 4: T-SNE visualizations for the 3 backbones of TaskWeaver, with Pythia-70M having an amorphous blob of predicted LORAs, while Qwen3-0.6B has clear-cut distinctions in the clustering as annotated in Fig. 4c

661 D Qualitative Examples

Task / Model	Prompt & Outputs
Instruction Following <i>Pythia-70M</i>	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... Base : Incoherent repetition TaskWeaver : B (Correct format)
Math Reasoning <i>Qwen3-0.6B</i>	Prompt : Johnny has 500 legos, another box with $3 \times$ more, and another with $1/4$ the number. Total? Base : Computes third box as $\frac{1}{4} \times 1500 = 375$ (wrong reference) TaskWeaver : Correctly computes $\frac{1}{4} \times 500 = 125$, total = 2125 (Correct)

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
GSM8K (<i>Math</i>)	Grade-school math word problems requiring 2–8 step arithmetic reasoning. Tests decomposition and numerical computation. 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 3/4 of the building is occupied. What's the total number of unoccupied units In the building?</i> Gemma3 (Base) : Let the total number of floors in the apartment building be F . Let the number of floors in the building be F . The number of units in each floor is 8. So, the total number of units in the building is $F \times 8 = 8F$. The number of units occupied is $\frac{3}{4}$ of the total number of units. The number of units occupied is $\frac{3}{4} \times 8F = \frac{3 \times 8F}{4} = \frac{24F}{4} = 6F$. The number of unoccupied units is the total number of units minus the number of units occupied. Unoccupied units = $F - \text{occupied units} = F - 6F = F - 6F = -5F$. Since the number of unoccupied units is negative, we need to find a positive value for F . The number of unoccupied units is $-5F$. The final answer is $-5F$. Therefore, the total number of unoccupied units in the building is $-5F$ 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 $15 \times 8 = 120$ units. The number of unoccupied units is the total number of units minus the number of occupied units, which is $120 - 90 = 30$ units
ARC-Easy (<i>Science</i>)	Grade-school science multiple-choice questions requiring reasoning beyond direct fact retrieval. 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> Gemma3 (Base) : A Gemma3 (TaskWeaver) : B

Continued on next page

Table 3 continued from previous page

Dataset	Description & Example
ARC-Challenge (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. Respond 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?</i> 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.</p> <p>Gemma3 (Base) : A Gemma3 (TaskWeaver) : C</p>
BoolQ (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: Passage: <i>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>
SNLI (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: Premise: <i>Many people standing outside of a place talking to each other in front of a building that has a sign that says "HI-POINTE."</i> Hypothesis: <i>The people are having a chat before going into the work building.</i></p> <p>Gemma3 (Base) : entailment Gemma3 (TaskWeaver) : neutral</p>
OpenBookQA (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:</i> A. a marsh B. a tundra C. the plains D. a desert</p> <p>Gemma3 (Base) : C Gemma3 (TaskWeaver) : A</p>
HellaSwag (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
Winogrande (Coreference)	Pronoun resolution requiring world knowledge to disambiguate references. 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> Gemma3 (Base) : 2. Derrick Gemma3 (TaskWeaver) : 1

Continued on next page

Table 3 continued from previous page

Dataset	Description & Example
<i>Zero-Shot Evaluation (not seen during training)</i>	
SVAMP* <i>(Math)</i>	<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 d be the number of marbles Ed had initially. Let y be the number of marbles Doug had initially. We are given that Ed had 10 more marbles than Doug, so $d = y + 10$. We are also given that Doug lost 11 marbles, so $y - 11 = 0$. Therefore, $y = 11$. Now we can find the number of marbles Ed had: $d = y + 10 = 11 + 10 = 21$. So Ed had 21 marbles. We want to find how many more marbles Ed had than Doug, which is $d - y$. $d - y = 21 - 11 = 10$.</p> <p>Gemma3 (TaskWeaver) : Let d be the number of marbles Ed had. Let g be the number of marbles Doug had. We are given that Ed had 10 more marbles than Doug, so $d = g + 10$. We are also given that Doug lost 11 marbles, so $g - d = 11$. We have the system of equations: $d = g + 10$ $g - d = 11$ From the first equation, we have $g = d + 10$. Substituting this into the second equation, we get: $g - (d + 10) = 11$ $g - d - 10 = 11$ $g - d = 21$ Now we have $d = g + 10$ and $g - d = 21$. We can substitute the first equation into the second equation: $g - (g + 10) = 21$ $g - g - 10 = 21 - 10 = 21$</p>
Commonsense- 5-way multiple choice questions requiring commonsense knowledge about concepts.	
QA* <i>(Reasoning)</i>	<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
RACE-Middle* (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 ..</p> <ul style="list-style-type: none"> 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>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.