Decoupling Generalization and Adaptation in Meta-Learning for Large Language Models

Anonymous ACL submission

Abstract

Fine-tuning large language models (LLMs) for downstream tasks remains expensive, even with parameter-efficient methods like Low-Rank Adaptation (LoRA). In this regard, meta-learning approaches such as Model-Agnostic Meta-Learning for LLMs (MAML-en-LLM) and Amortized Bayesian Meta-Learning for LoRA (ABMLL) have emerged as promising solutions for rapid downstream LLM adaptation. However, these methods fundamentally couple two distinct objectives: learning generalizable initializations and enabling efficient task adaptation. We argue that this coupling limits both the quality of learned representations and adaptation efficiency. In this paper, we introduce **DeGAML-LLM** (**De**coupled **G**eneralization and **A**daptation **M**eta-**L**earning for **L**arge **L**anguage **M**odels), a novel framework that explicitly separates these two objectives through dedicated parameter spaces. Specifically, we maintain a generalization module that learns task-agnostic representations across the task distribution, and an adaptation module that specializes in rapid task-specific adjustment. Extensive experiments on common-sense reasoning, mathematics, logic, social, medical and coding benchmarks across model scales demonstrate that DeGAML-LLM outperforms existing meta-learning and standard multi-task baselines.

1 Introduction

Large Language Models (LLMs) such as GPT-4 (Achiam et al., 2023), LLaMA (Grattafiori et al., 2024), and Qwen (Yang et al., 2024) have demonstrated remarkable capabilities across a wide range of natural language processing tasks. However, adapting these models to specific downstream tasks typically requires multi-step fine-tuning with substantial training data, incurring significant computational overhead. Although Parameter-Efficient Fine-Tuning (PEFT) methods (Hu et al., 2021; Houlsby et al., 2019) reduce the number of trainable parameters, fine-tuned LLMs often remain tightly coupled to the training domain, with performance failing to generalize or even degrading in new domains. Consequently, domain transfer frequently necessitates repeating the entire fine-tuning process (Kotha et al., 2024; Luo et al., 2025).

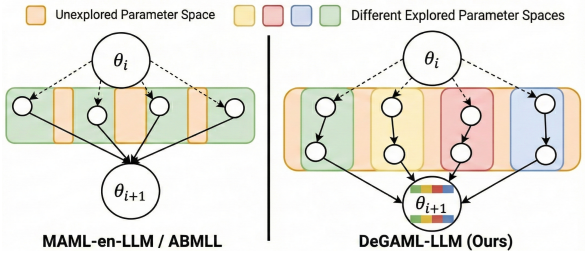


Figure 1: **Visual comparison of parameter exploration and update dynamics across meta-learning paradigms** - The figure illustrates a single update step from parameters at iteration $i : \theta_i$ to iteration $i+1 : \theta_{i+1}$. Dotted arrows denote task-specific adaptation steps, while solid arrows indicate the final parameter update. MAML-en-LLM / ABMLL explicitly adapt parameters for multiple tasks, exploring a broader shared parameter region and aggregating these adapted parameters via a coupled meta-update. DeGAML-LLM (ours) decouples generalization and adaptation: task-conditioned parameter generation explores diverse and disjoint regions of the parameter space, while task-specific adaptation proceeds independently before contributing to the final update. This separation enables richer exploration and avoids constraining adaptation to a single shared optimization trajectory (See Appendix A).

To address these limitations, meta-learning offers a promising solution: training models across diverse tasks to learn generalizable parameters that enable rapid adaptation to new tasks with only a few examples (Schmidhuber, 1987; Finn et al., 2017). Recently, meta-learning has been applied to LLM fine-tuning along two main directions. The first adapts the Model-Agnostic Meta-Learning (MAML) framework for efficient

and rapid fine-tuning of LLMs, as exemplified by MAML-en-LLM (Sinha et al., 2024). The second explores Bayesian parameter-generative approaches, where task-adaptive parameters are generated from a shared prior to facilitate fine-tuning, such as ABMLL (Zhang et al., 2025a).

Although these meta-learning methods[1] (Sinha et al., 2024; Zhang et al., 2025a; Min et al., 2022) have achieved promising results, their reliance on coupled optimization of general initialization and task-specific parameters introduces two key concerns. First, learning both parameter types is tightly intertwined despite *misaligned objectives*: one seeks cross-task generalization, while the other prioritizes rapid task-specific adaptation, leading to optimization conflicts and inevitable trade-offs. Second, both parameter types require *explicit gradient updates*, restricting adaptation to gradient-based procedures. This proves overly restrictive in settings with large task discrepancies, diverse metrics, or non-differentiable feedback, limiting flexibility across heterogeneous tasks.

Motivated by these concerns, we present a decoupled meta-learning paradigm that separates stable initialization from task adaptation, freeing the latter to employ more flexible, task-specific strategies. Specifically, in the initialization stage, we learn a task-conditioned parameter generator that captures shared structure across the task distribution and outputs an initial set of adapter parameters, without committing to any particular adaptation rule. Task adaptation is then handled by an independent, closed-loop mechanism that refines these parameters using task-level feedback, rather than gradients tied to the meta-training objective. This separation eliminates the need to meta-learn a fixed adaptation trajectory, allowing adaptation strategies to vary across tasks, metrics, and domains while preserving strong generalization.

Our main contributions are as follows:

1. We analyze existing meta-learning methods for LLMs and highlight the limitations arising from coupling cross-task generalization and task-specific adaptation within a single optimization process.
2. We propose DeGAML-LLM, a decoupled meta-learning framework that separates generalization and adaptation into distinct parameter modules, enabling more flexible task-specific adaptation.
3. We empirically demonstrate that DeGAML-LLM outperforms prior meta-learning and multi-task baselines on a diverse set of in-domain and out-of-domain benchmarks.

[1]Refer Appendix B for related works

2 Methodology

We now present our decoupled meta-learning framework for LLMs by using parameter generation and task adaptation. Our formulation separates the process of learning cross-task structural knowledge from per-task adaptation, enabling flexible adaptation via closed-loop reinforcement learning.

2.1 Preliminaries and Notation

Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ denote a distribution over N tasks, where each task $\mathcal{T}_i$ provides a dataset $D_i = (D_i^{\text{tr}}, D_i^{\text{val}})$ composed of training and validation splits. Each task is drawn from an unseen test distribution $\mathcal{T}_*$ at meta-test time. We consider a pretrained LLM parameterized by θ^0 (frozen) and a set of low-rank adapter parameters $\theta^a \in \mathbb{R}^d$ (trainable). Our goal[2] is to learn - 1. A generalization module $\mathcal{G}_\phi$, with parameters ϕ, that maps task data to a distribution over LoRA adapter parameters and 2. an adaptation policy, π_ψ with parameters ψ, that refines parameters for a novel task via closed-loop reinforcement learning.

2.2 Decoupled Objective

Classical meta-learning uses a coupled bi-level objective:

$$\min_{\phi} \sum_{i=1}^{N} \mathcal{L}_{\mathcal{T}_i}(\theta_i^a(\phi), D_i^{\text{val}}), \quad \text{s.t. } \theta_i^a(\phi) = \text{Adapt}(\phi, D_i^{\text{tr}}), \tag{1}$$

where $\text{Adapt}(\cdot)$ denotes gradient steps or Bayesian inference. In our case, we instead decouple generalization and adaptation into separate objectives:

Generalization: We learn a generator $\mathcal{G}_\phi$ that produces a distribution over adapter parameters conditional on a task support set:

$$\mathcal{G}_\phi : D_i^{\text{tr}} \mapsto q_\phi(\theta_i^a \mid D_i^{\text{tr}}), \tag{2}$$

such that the parameters drawn from $q_\phi(\cdot)$ maximize expected validation performance:

$$\min_{\phi} \sum_{i=1}^{N} \mathbb{E}_{\theta_i^a \sim q_\phi(\cdot|D_i^{\text{tr}})} \left[\mathcal{L}_{\mathcal{T}_i}(\theta_i^a, D_i^{\text{val}}) \right]. \tag{3}$$

[2]Refer Appendices C, E and F for details about training procedure as well generalization and adaptation modules

In-Domain Tasks

Dataset	ARC-c	ARC-e	HellaSwag	BoolQ	PIQA	WinoGrande	**Avg**
				Qwen2.5-1.5B-Instruct			
No Meta-Train LoRA	**74.5**	84.4	55.8	55.6	65.6	48.2	64.0
Union Train LoRA	63.2	73.9	48.9	55.1	47.8	**61.3**	58.3
ABMLL	69.9	83.2	51.1	**63.2**	54.3	52.9	62.4
MAML-en-LLM	66.0	84.3	**59.3**	58.7	68.1	56.8	65.5
DeGAML-LLM	73.7	**88.4**	57.2	58.8	**70.7**	57.3	**67.7**
Δ (vs MAML-en-LLM)	+7.7	+4.1	-2.1	+0.1	+2.6	+0.5	+2.2
Δ (vs ABMLL)	+3.8	+5.2	+6.1	-4.4	+16.4	+4.4	+5.3
Δ (vs No Meta-Train)	-0.8	+4.0	+1.4	+3.2	+5.1	+9.1	+3.7
Δ (vs Union Train)	+10.5	+14.5	+8.3	+3.7	+22.9	-4.5	+9.4
				Qwen2.5-0.5B-Instruct			
No Meta-Train LoRA	40.7	59.4	23.4	22.1	66.2	35.7	41.2
Union Train LoRA	39.7	47.4	26.3	14.7	51.1	50.5	38.3
ABMLL	37.6	54.4	26.5	**62.2**	37.6	34.5	42.1
MAML-en-LLM	47.7	63.7	36.3	46.2	**67.7**	50.1	51.9
DeGAML-LLM	**55.5**	**74.7**	**48.3**	58.7	60.1	**52.8**	**58.4**
Δ (vs MAML-en-LLM)	+7.8	+11.0	+12.3	+12.5	-7.6	+2.7	+6.5
Δ (vs ABMLL)	+17.9	+20.3	+21.8	-3.5	+22.5	+18.3	+16.3
Δ (vs No Meta-Train)	+14.8	+15.3	+24.9	+36.7	-6.1	+17.1	+17.2
Δ (vs Union Train)	+15.8	+27.3	+22.0	+44.1	+9.0	+2.3	+20.1

Out-of-Domain Tasks

Dataset	GSM-8K	MATH	DivLogicEval	SocialIQA	CodeMMLU	JAMA	**Avg**
				Qwen2.5-1.5B-Instruct			
Union Train LoRA	34.2	32.2	24.1	51.4	34.7	34.7	36.1
ABMLL	28.7	15.9	26.9	66.3	39.6	28.5	34.3
MAML-en-LLM	35.6	43.5	31.2	68.7	42.3	32.5	42.3
DeGAML-LLM	**51.4**	**46.9**	**31.4**	**69.5**	**44.6**	**41.5**	**47.5**
Δ (vs MAML-en-LLM)	+15.8	+3.4	+0.2	+0.8	+2.3	+9.0	+5.3
Δ (vs ABMLL)	+22.7	+31.0	+4.5	+3.2	+5.0	+13.0	+13.2
Δ (vs Union Train)	+17.2	+14.7	+7.3	+18.1	+9.9	+6.8	+11.4
				Qwen2.5-0.5B-Instruct			
Union Train LoRA	15.6	6.8	20.3	39.5	29.8	29.9	29.9
ABMLL	20.4	7.1	23.7	53.1	28.2	16.8	24.9
MAML-en-LLM	29.1	**26.3**	25.1	54.9	34.1	26.4	32.6
DeGAML-LLM	**30.3**	24.5	**28.7**	**55.1**	**35.6**	**31.2**	**34.2**
Δ (vs MAML-en-LLM)	+1.2	-1.8	+3.6	+0.2	+1.5	+4.8	+1.6
Δ (vs ABMLL)	+9.9	+17.4	+5.0	+2.0	+7.4	+14.4	+9.3
Δ (vs Union Train)	+14.7	+17.7	+8.4	+15.6	+5.8	+1.3	+4.3

Table 1: In-domain and Out-of-domain performance comparing DeGAML-LLM against baseline adaptation methods across general knowledge, common-sense, mathematical, logic, medical, social and coding reasoning benchmarks. Δ rows highlight absolute gains over MAML-en-LLM, ABMLL, No Meta-Train and Union Train LoRA baselines.

Adaptation: Given an unseen task $\mathcal{T}_*$ and a small support set D_*^{tr}, we initialize adapter parameters:

$$\theta_*^a \sim q_\phi(\cdot \mid D_*^{\text{tr}}), \tag{4}$$

and refine them via an RL policy π_ψ that proposes updates to θ^a based on feedback from validation performance. At each adaptation step t, the policy observes a state s_t representing the current performance indicators (e.g. prediction metrics like accuracy) and outputs an action a_t that perturbs the adapter parameters $\theta_{t+1}^a = \theta_t^a + a_t, \quad a_t \sim \pi_\psi(a_t \mid s_t)$.The agent receives a scalar reward binary reward r_t, based on the improvement in task performance $r_t = -\mathcal{L}_{\mathcal{T}_*}(\theta_{t+1}^a, D_*^{\text{val}})$, which is then used for ReST^{EM} training (Singh et al., 2024).

3 Results

We evaluate the proposed decoupled meta-learning framework (DeGAML-LLM) against the most recent LoRA-based LLM meta-learning baseline[3]

[3]In addition, we also compare with two standard multi-task baselines namely, “No Meta Train LoRA" that simply ignores the meta train set, and only fine-tune the LoRAs on the training set of each meta test task and “Union Train LoRA" which is a standard multi-task baseline that involves combining all meta training tasks into a big single union dataset, and then training the LoRAs on the union set. At meta test time it performs

In-Domain Tasks

Dataset	ARC-c	ARC-e	HellaSwag	BoolQ	PIQA	WinoGrande	**Avg**
				Qwen2.5-1.5B-Instruct			
Base Model	71.5	83.0	50.9	56.3	45.8	50.6	59.6
Generalization	73.0 (+2.1%)	83.7 (+0.8%)	56.2 (+10.4%)	55.2 (-2.0%)	56.4 (+23.1%)	50.2 (-0.8%)	62.5 (+4.9%)
Adaptation	73.7 (+1.0%)	88.4 (+5.6%)	57.2 (+1.8%)	58.8 (+6.5%)	70.7 (+25.4%)	57.3 (+14.1%)	67.7 (+8.3%)
				Qwen2.5-0.5B-Instruct			
Base Model	38.3	54.8	26.5	37.0	16.6	50.2	37.2
Generalization	42.7 (+11.5%)	63.2 (+15.3%)	25.9 (-2.3%)	44.9 (+21.4%)	47.6 (+186.7%)	50.0 (-0.4%)	45.7 (+22.9%)
Adaptation	55.5 (+30.0%)	74.7 (+18.2%)	48.3 (+86.5%)	58.7 (+30.7%)	60.1 (+26.3%)	52.8 (+5.6%)	58.4 (+27.8%)

Out-of-Domain Tasks

Dataset	GSM-8K	MATH	DivLogicEval	SocialIQA	CodeMMLU	JAMA	**Avg**
				Qwen2.5-1.5B-Instruct			
Base Model	51.8	30.3	28.3	65.9	42.6	38.9	42.9
Generalization	32.6 (-37.1%)	40.1 (+32.3%)	28.6 (+1.1%)	68.6 (+4.1%)	44.1 (+3.5%)	39.5 (+1.5%)	42.2 (-1.6%)
Adaptation	51.4 (+57.7%)	46.9 (+17.0%)	31.4 (+9.8%)	69.5 (+1.3%)	44.6 (+1.1%)	41.5 (+5.1%)	47.5 (+12.6%)
				Qwen2.5-0.5B-Instruct			
Base Model	15.2	2.8	22.4	50.8	32.4	23.8	24.5
Generalization	20.8 (+36.8%)	24.1 (+760.7%)	21.0 (-6.3%)	33.5 (-34.1%)	29.1 (-10.2%)	11.7 (-50.8%)	25.7 (+4.9%)
Adaptation	30.3 (+45.7%)	29.2 (+21.2%)	28.7 (+36.7%)	55.1 (+64.5%)	35.6 (+22.3%)	31.2 (+166.7%)	34.2 (+33.1%)

Table 2: Ablation study analyzing the contributions of generalization and adaptation components in DeGAML-LLM. **Base Model** denotes the frozen pretrained LLM without any LoRA adapters. **Generalization** evaluates performance using a single set of LoRA parameters generated by the task-conditioned generator $\mathcal{G}_\phi$, with no task-specific refinement and **adaptation** refers to applying RL-based ReST^{EM} refinement to LoRA parameters generated by $\mathcal{G}_\phi$.

(ABMLL (Zhang et al., 2025a)) as well as MAML-en-LLM (Sinha et al., 2024) on a suite of natural language understanding and reasoning benchmarks. Results (see Table 1) are reported for two model scales: Qwen2.5-1.5B-Instruct and Qwen2.5-0.5B-Instruct. We measure performance across a mix of in-domain and out-of-domain tasks[4]. For in-domain tasks, we consider ARC-challenge & ARC-easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019) and WinoGrande (Sakaguchi et al., 2021). For out-of-domain tasks, we evaluated on GSM-8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), DivLogicEval (Chung et al., 2025), SocialIQA (Sap et al., 2019), CodeMMLU (Manh et al., 2025) and JAMA Clinical (Chen et al., 2025). Table 2 illustrates each individual components effectiveness.

PEFT starting from the trained model as a initial model.

[4]We follow the setting of ABMLL and use 20 datapoints only of the target dataset for adaptation

4 Conclusion

We presented a decoupled meta-learning framework that separates cross-task generalization (via task-conditioned parameter generation) from task-specific adaptation (via closed-loop RL). Unlike traditional coupled meta-learning that integrates both processes through bi-level optimization, our approach enables flexible adaptation strategies selected based on observed performance. Empirical evaluations on diverse natural language understanding, reasoning and out-of-domain transfer tasks demonstrate that our approach consistently outperforms strong baselines across multiple model scales. The proposed formulation opens several promising research directions, including more expressive adaptation policies, learned rewards and scaling to ultra-large language models. We posit this represents an important step toward meta-learning that is both flexible and scalable, laying the groundwork for more adaptive and efficient language models in real-world applications.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, and 1 others. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. Preprint, arXiv:1911.11641.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. Benchmarking large language models on answering and explaining challenging medical questions. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 3563–3599, Albuquerque, New Mexico. Association for Computational Linguistics.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics.

Tsz Ting Chung, Lemao Liu, Mo Yu, and Dit-Yan Yeung. 2025. Divlogiceval: A framework for benchmarking logical reasoning evaluation in large language models. Preprint, arXiv:2509.15587.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. 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: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. Preprint, arXiv:1803.05457.

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. 2021. Training verifiers to solve math word problems. Preprint, arXiv:2110.14168.

Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matt Botvinick, Jane Wang, and Eric Schulz. 2023. Meta-in-context learning in large language models. Advances in Neural Information Processing Systems, 36:65189–65201.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning, pages 1126–1135.

Aaron Grattafiori and 1 others. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. Preprint, arXiv:2103.03874.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9):5149–5169.

Zejiang Hou, Julian Salazar, and George Polovets. 2022. Meta-learning the difference: preparing large language models for efficient adaptation. Transactions of the Association for Computational Linguistics, 10:1249–1265.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, and 1 others. 2019. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

Jinwu Hu, Zitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. 2025a. Test-time learning for large language models. In Forty-second International Conference on Machine Learning.

Yang Hu, Xingyu Zhang, Xueji Fang, Zhiyang Chen, Xiao Wang, Huatian Zhang, and Guojun Qi. 2025b. Slot: Sample-specific language model optimization at test-time. arXiv preprint arXiv:2505.12392.

Minjun Kim, Jeongwon Park, Seonghun Kang, and Bongkyoung Kwon. 2025. Reptilora: accelerating meta-initialization with low-rank adaptation. Issues in Information Systems, 26(3).

Minyoung Kim and Timothy Hospedales. 2025. Lift: Learning to fine-tune via bayesian parameter efficient meta fine-tuning. In The Thirteenth International Conference on Learning Representations.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2024. Understanding catastrophic forgetting in language models via implicit inference. In ICLR.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, and 1 others. 2025. Drag-and-drop llms: Zero-shot prompt-to-weights. arXiv preprint arXiv:2506.16406.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during

continual fine-tuning. IEEE Transactions on Audio, Speech and Language Processing.

Dung Nguyen Manh, Thang Phan Chau, Nam Le Hai, Thong T. Doan, Nam V. Nguyen, Quang Pham, and Nghi D. Q. Bui. 2025. Codemmlu: A multi-task benchmark for assessing code understanding & reasoning capabilities of codellms. Preprint, arXiv:2410.01999.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. Metaicl: Learning to learn in context. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. Commun. ACM, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. Preprint, arXiv:1904.09728.

Jürgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. Diploma thesis, Technische Universität München.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, and 22 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. Preprint, arXiv:2312.06585.

Sanchit Sinha, Yuguang Yue, Victor Soto, Mayank Kulkarni, Jianhua Lu, and Aidong Zhang. 2024. Maml-en-llm: Model agnostic meta-training of llms for improved in-context learning. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems.

Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. Learning to Learn, pages 3–17.

Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. Artificial Intelligence Review, 18(2):77–95.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, and 1 others. 2016. Matching networks for one shot learning. In Advances in Neural Information Processing Systems, pages 3630–3638.

Kai Wang and 1 others. 2025. Recurrent parameter generation. arXiv preprint.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024. Mixture-of-subspaces in low-rank adaptation. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 7880–7899.

An Yang, Baosong Yang, Binyuan Hui, and 1 others. 2024. Qwen2.5 technical report. arXiv preprint arXiv:2409.12186.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Liyi Zhang, Jake Snell, and Thomas L. Griffiths. 2025a. Amortized bayesian meta-learning for low-rank adaptation of large language models. Conference on Language Modeling (COLM).

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025b. A survey on test-time scaling in large language models: What, how, where, and how well? arXiv preprint arXiv:2503.24235.

A Visuals

A.1 Meta-Learning Paradigm Comparison

Figure 2 illustrates the fundamental difference between traditional coupled meta-learning and our proposed decoupled approach. In coupled frameworks (MAML, ABMLL), the meta-learner directly produces task-specific parameters through gradient-based or Bayesian updates that are back-propagated end-to-end during meta-training. This tight coupling means the adaptation procedure is fixed during meta-learning and cannot be changed at test time. In contrast, DeGAML-LLM decouples these processes: the generalization module learns only to generate good initializations by capturing cross-task structure, while the adaptation module independently learns to refine parameters through closed-loop feedback. This separation enables flexible test-time adaptation strategies (TTT, mixing, ensembling) not available in coupled approaches.

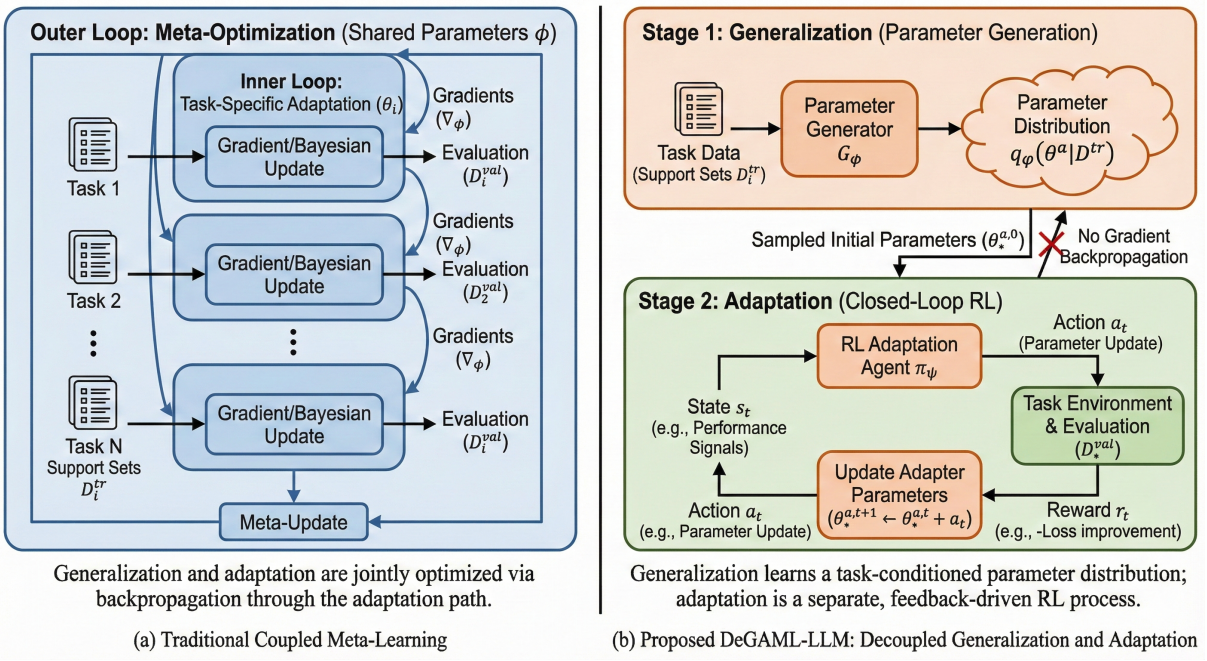


Figure 2: Comparison of coupled meta-learning (MAML, ABMLL) versus our proposed decoupled approach (DeGAML-LLM). Coupled methods learn parameters and adaptation jointly through bi-level optimization. DeGAML-LLM separates generalization (parameter generation from cross-task knowledge) from adaptation (flexible RL-based refinement).

A.2 Architecture Mechanisms

Figure 3 shows how different meta-learning approaches modify the internal transformer architecture:

- **MAML-en-LLM (a)**: Updates all LoRA weights via gradient descent from a meta-learned initialization. Adaptation is gradient-based and follows a fixed trajectory determined during meta-training.
- **ABMLL (b)**: Samples LoRA adapters from a learned Bayesian posterior $p(\theta^a \mid \text{meta-data}, \text{task-data})$. Coupling occurs through variational inference requiring gradients through the entire model.
- **DeGAML-LLM (c)**: Generator $\mathcal{G}_\phi$ produces initial weights from task prompts, then RL policy π_ψ refines them without backpropagation to the generator. The generator and policy are trained separately, enabling modular adaptation.

The key architectural insight is that DeGAML-LLM's policy can observe performance feedback and adaptively select refinement operations (e.g., test-time training on task data, mixing multiple generated adapters, ensemble voting) rather than being constrained to a single gradient-based update rule.

B Related Work

Meta-learning enables rapid task adaptation by leveraging shared structure across task distributions (Schmidhuber, 1987; Thrun and Pratt, 1998; Hospedales et al., 2021). Classical approaches like Matching Networks (Vinyals et al., 2016), Prototypical Networks (Snell et al., 2017), and MAML (Finn et al., 2017) established bi-level optimization paradigms (Vilalta and Drissi, 2002; Nichol et al., 2018), but their application to LLMs faces unique scalability challenges.

B.1 Gradient-Based Meta-Learning for LLMs

The Model-Agnostic Meta-Learning (MAML) family exemplifies gradient-based approaches that optimize for initial parameters from which tasks can be adapted with a few gradient steps. These methods have been effective on standard few-shot classification and reinforcement learning benchmarks. However, they inherently bind generalization and adaptation through a shared optimization trajectory, requiring second-order gradients or multiple inner-loop updates that are costly for large models. Variants such as MAML-en-LLM (Sinha et al., 2024),

MLtD (Hou et al., 2022), ReptiLoRA (Kim et al., 2025) adapt this framework to large pretrained language models, demonstrating improved unseen domain and adaptation performance via meta-training tailored for LLMs. Although promising, these approaches still adhere to coupled adaptation dynamics and necessitate gradient backpropagation through the LLM's entire parameter space, which remains expensive and often impractical at scale.

B.2 Bayesian and Amortized Meta-Learning for LLMs

Probabilistic meta-learning models introduce a generative perspective on task parameters, learning priors or distributions that can produce task-specific models. Hierarchical Bayesian frameworks such as LiFT (Kim and Hospedales, 2025)(Learning to Fine-Tune) treat PEFT adapter parameters of LLMs as random variables governed by a higher-level latent prior that captures shared information across tasks. By performing efficient sampling and inference, such models offer better generalization and can outperform traditional meta-learning and heuristic mixing approaches. More recently, Amortized Bayesian Meta-Learning for LoRA (ABMLL) (Zhang et al., 2025a) has been proposed specifically for large language models. ABMLL adapts amortized Bayesian meta-learning techniques to the LoRA parameterization of LLMs, reframing global and task-specific parameters to improve computational efficiency and generalization on multi-task benchmarks such as Unified-QA and CrossFit. The Bayesian formulation also yields improved uncertainty quantification for LLM adaptation. While these Bayesian approaches enable principled incorporation of uncertainty and task parameter distributions, they still rely on gradient-based inference or sampling within the meta-training loop, thereby retaining some of the limitations of coupled meta-learning, particularly for large pretrained models.

B.3 Adaptation Beyond Gradients

While the above methods primarily rely on gradients or approximate inference within the meta-training paradigm, emerging research suggests alternative adaptation strategies for large models that do not depend on gradient descent. For instance, in-context meta-learning (Coda-Forno et al., 2023; Chen et al., 2022) explores recursive in-context learning capabilities of LLMs to improve task performance without parameter updates.

Our framework diverges from these approaches by explicitly decoupling the generalization and adaptation processes, learning a generative model over task parameters in a task-agnostic manner, and adopting closed-loop reinforcement learning for adaptation that does not require gradients through the meta-learner or LLM parameters. This enables modular adaptation policies and addresses scalability and flexibility limitations inherent in prior gradient-coupled meta-learning approaches.

C Training

The meta-training procedure for DeGAML-LLM follows a **two-stage sequential process** that explicitly decouples generalization from adaptation. Crucially, **gradients from adaptation performance do not flow back to the generator**, ensuring true decoupling. All experiments were conducted used seed 999.

Stage 1: Generalization Module Training (Offline)

The first stage trains the parameter generator $\mathcal{G}_\phi$ to learn the cross-task parameter manifold by minimizing reconstruction error on collected checkpoints. This stage is completely independent of any adaptation mechanism.

Step 1.1: LoRA Checkpoint Collection. For each meta-training task $\mathcal{T}_i$, we collect a set of LoRA adapter checkpoints by:

1. Pretraining LoRA adapters on task $\mathcal{T}_i$ for 75 steps at learning rate 10^{-4} with batch size 32 using at max 5000 training samples
2. Fine-tuning for an additional 50 steps at learning rate 10^{-5}
3. Saving checkpoint parameters m_j at each step during both pretraining and fine-tuning

This process generates approximately 125 checkpoints per task, capturing the trajectory of task-specific adaptation from initialization to convergence. To create training data for the generator, we randomly pair prompt batches with checkpoints from the same task. Given task $\mathcal{T}_i$ with prompt batches $[p_1^i, \ldots, p_I^i]$ and checkpoints $[m_1^i, \ldots, m_J^i]$, we randomly sample pairs (p_k^i, m_j^i) to create approximately 5,000 training samples per task.

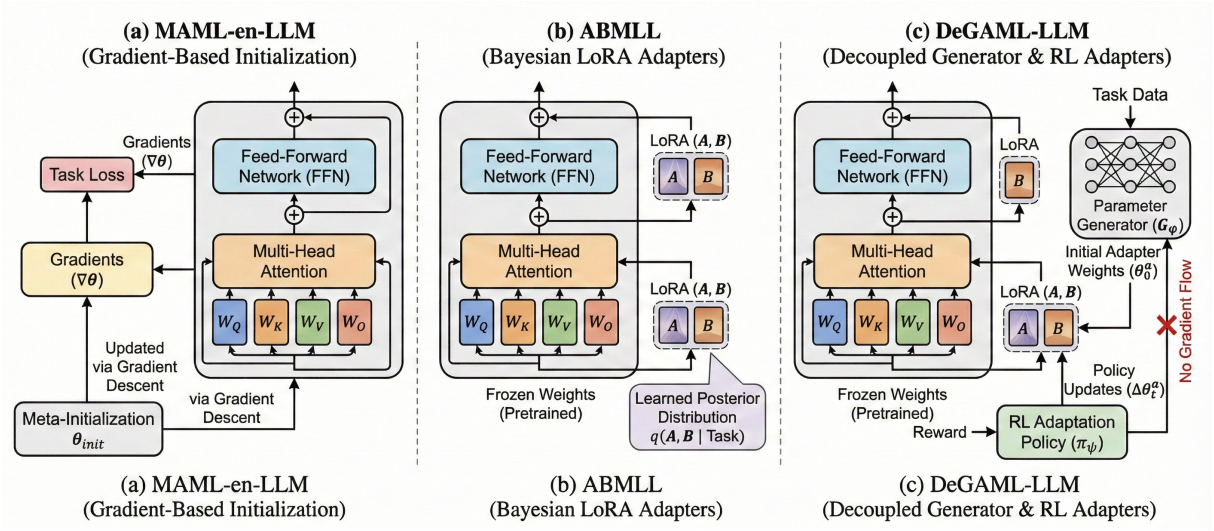


Figure 3: Internal Transformer Architecture Modifications. (a) MAML-en-LLM updates all weights via gradients from a meta-learned initialization (b) ABMLL samples low-rank LoRA adapters from a learned Bayesian posterior distribution (c) DeGAML-LLM uses a separate generator to predict initial adapter weights, which are then refined by a separate RL policy without gradient backpropagation to the generator.

Step 1.2: Generator Training via MSE Loss. The generator is trained to minimize mean squared error between generated parameters and checkpoint parameters:

$$\min_{\phi} \mathbb{E}_{(p_k, m_j)\sim\mathcal{D}_{\text{meta}}} \|\mathcal{G}_\phi(p_k) - m_j\|^2, \qquad (5)$$

where p_k denotes prompt batch embeddings extracted via Sentence-BERT and m_j denotes checkpoint parameters. This objective trains ϕ to capture the **cross-task parameter manifold**-the shared statistical structure of task-adapted parameters-**without encoding any specific adaptation dynamics**. The generator learns what good task-conditioned initializations look like, not how to adapt them. This realizes the generalization component of our decoupled framework.

Stage 2:Adaptation Module Training (Online)

After Stage 1 completes, the generator $\mathcal{G}_\phi$ is **frozen**, and we train the RL policy π_ψ to learn effective adaptation strategies. The key architectural principle is that **generator outputs are detached from the computation graph**-gradients from the RL objective cannot flow back to ϕ.

The training procedure alternates between policy rollouts and policy updates:

Policy Rollout. For each meta-training iteration:

1. Sample a batch of tasks $\{\mathcal{T}_i\}$ from the meta-training distribution
2. For each task $\mathcal{T}_i$:
   - Draw support set D_i^{tr} and validation set D_i^{val}
   - Initialize adapter parameters $\theta_{i,0}^a \sim q_\phi(\cdot \mid D_i^{\text{tr}})$ **(with gradient detachment)**
   - Execute T adaptation steps where the policy observes performance and proposes parameter updates
   - Compute cumulative reward R_i based on validation performance. In particular, these rewards are binary implying that an action gets a reward of 1 if the adapted parameters perform better than initialization else zero.

Policy Update. After collecting rollouts for all tasks in the batch, update the policy parameters ψ via policy gradients (ReST$^{\text{EM}}$ (Singh et al., 2024)) to maximize expected cumulative reward. Critically, **gradients are stopped at the generator outputs** and the policy learns to select effective adaptation strategies given the generator's outputs, but the generator itself is not updated based on adaptation performance.

This two-stage approach ensures that:

- The generalization module focuses solely on learning cross-task structure
- The adaptation module learns flexible, task-specific refinement strategies

Algorithm 1 Generalization Module Training (Offline) of DeGALM-LLM

Require: Meta-training tasks $\{\mathcal{T}_i\}$, pretrained LLM θ^0, parameter generator $\mathcal{G}_\phi$

// Goal: Learn task-conditioned parameter generator via MSE on checkpoints
Step 1.1: Collect LoRA Checkpoints
for each task $\mathcal{T}_i$ in meta-training set **do**
  Pretrain LoRA on $\mathcal{T}_i$ for 75 steps at lr=10^{-4}, batch=32
  Fine-tune LoRA on $\mathcal{T}_i$ for 50 steps at lr=10^{-5}
  Save checkpoints $\{m_j^i\}$ at each step
  Create prompt-checkpoint pairs (p_k^i, m_j^i) via random sampling
end for
Step 1.2: Train Parameter Generator
for each training iteration **do**
  Sample batch of prompt-checkpoint pairs $\{(p_k, m_j)\}$
  for each pair (p_k, m_j) in batch **do**
    Extract task embedding $c_k \leftarrow \text{Encoder}(p_k, \theta_{\text{enc}})$
    Generate parameters $\hat{m}_k \leftarrow \mathcal{G}_\phi(c_k)$
    Compute MSE loss $\mathcal{L}_k \leftarrow \|\hat{m}_k - m_j\|^2$
  end for
  Update $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \sum_k \mathcal{L}_k$
end for

Algorithm 2 Adaptation Module Training (Online) of DeGALM-LLM

Require: Parameter generator $\mathcal{G}_\phi$, adaptation policy π_ψ

// Goal: Learn RL policy to refine generated parameters
for each meta-training iteration **do**
  Sample batch of tasks $\{\mathcal{T}_i\}$
  for each task $\mathcal{T}_i$ in batch **do**
    Draw support D_i^{tr}, validation D_i^{val}
    Initialize $\theta_{i,0}^a \sim q_\phi(\cdot \mid D_i^{\text{tr}})$ **// Detach: stop gradients**
    for $t = 0 \dots T-1$ **do**
      Get state $s_{i,t}$ from current performance on D_i^{val}
      Sample action $a_{i,t} \sim \pi_\psi(a \mid s_{i,t})$
      Apply update $\theta_{i,t+1}^a \leftarrow \theta_{i,t}^a + a_{i,t}$
      Evaluate and receive reward $r_{i,t} = -\mathcal{L}(\theta_{i,t+1}^a, D_i^{\text{val}})$
    end for
    Store rewards $R_i \leftarrow \{r_{i,1}, \dots, r_{i,T}\}$
  end for

  Update ψ via policy gradient (ReST$^{\text{EM}}$) using R_i (Gradients do NOT reach ϕ)
end for
Output: Trained generator $\mathcal{G}_\phi$ and adaptation policy π_ψ

- No coupling exists between the two objectives during training

Hardware: All experiments were conducted on a HPC node running Ubuntu 22.04.1. The backend processor was EPYC 8434P, which had 48 physical cores (96 logical threads), 256 GB of system RAM and a maximum clock speed of 2.5 GHz. Four NVIDIA RTX A6000 GPUs, each with 48 GB of dedicated VRAM were utilized. Python version used was 3.12.11 and GPU-accelerated tasks were managed using CUDA version 12.4.

D Limitations and Future Work

While our decoupled meta-learning framework (DeGAML-LLM) demonstrates significant empirical improvements over coupled baselines such as ABMLL and MAML-en-LLM as well as standard multi-task baselines across a range of in-domain and out-of-domain tasks, several limitations and avenues for further research remain.

Adaptation Efficiency and Latency- Although DeGAML-LLM's adaptation mechanism avoids backpropagating through the entire meta-learner, the reinforcement adaptation policy may require multiple environment interactions[5] before converging to optimal task parameters. This can introduce latency relative to traditional gradient updates, especially on tasks where evaluation is expensive. Future investigations could focus on sample-efficient RL algorithms, model-based adaptation strategies, or hybrid gradient + policy updates to improve adaptation speed without sacrificing flexibility.

Robustness and Safety- Closed-loop reinforcement adaptation introduces new dynamics that could be sensitive to adversarial or noisy feedback. Ensuring robust adaptation in the presence of unreliable rewards and aligning adaptation behavior with safety and fairness constraints are important practical challenges for deployment in real-world systems.

[5] Current experiments have consistently used 20 episodes

E Generalization Module

In our decoupled meta-learning framework, the **generalization module** learns task-agnostic structural knowledge across the meta-training distribution. Implemented as a task-conditioned parameter generator $\mathcal{G}_\phi$, this module models distributions over LoRA adapter parameters that capture shared cross-task structure **without encoding any specific adaptation trajectory**. This independence from adaptation mechanisms is the key distinction from coupled meta-learning approaches.

E.1 Task Embedding Extraction

Given a task support set D_i^{tr} containing unlabeled task examples, we extract a task-level embedding using a pretrained text encoder:

$$c_i = \text{Encoder}(p_i, \theta_{\text{enc}}), \tag{6}$$

where p_i denotes the task prompt batch. We use Sentence-BERT (all-MiniLM-L6-v2), producing embeddings $c_i \in \mathbb{R}^{L\times C}$ with dimension $[B, N, L, C]$ where B is batch size, N is number of prompts (set to 128 in this work), L is sequence length (set to 512) and C is hidden dimension (set to 384).

E.2 Parameter Tokenization

We employ a parameter tokenization strategy following (Wang et al., 2025), which transforms LoRA adapter weights into a sequence of uniform tokens suitable for processing by the generalization model. It involves,

Layer-wise splitting & normalization- Given complete LoRA adapter parameters W spanning all layers, first parameters are segregated by layer index and then layer-wise normalization is applied to reduce distribution shifts across layers:

$$\begin{aligned} W &\xrightarrow{\text{split by layer}} [w[1], \ldots, w[I]] \\ &\xrightarrow{\text{normalize}} [\hat{w}[1], \ldots, \hat{w}[I]] \end{aligned} \tag{7}$$

Uniform tokenization- Each normalized layer $\hat{w}[i]$ is then partitioned into contiguous, non-overlapping chunks of uniform size k (with padding applied to the final chunk if necessary):

$$\hat{w}[i] \xrightarrow{\text{tokenize}} K[i] = \left[k_i^1, k_i^2, \ldots, \text{pad}\left(k_i^{J_i}\right)\right], \tag{8}$$

where J_i denotes the number of tokens for layer i, and $\text{pad}(\cdot)$ indicates zero-padding to achieve uniform token length k. Each checkpoint W is then assigned a unique permutation state S encoded as a one-hot vector. Each token is further augmented with 2D sinusoidal position embeddings. For the j-th token in layer i, $e_i^j = \text{PE}_{2D}(i, j)$, is computed, where the first dimension encodes layer index i and the second dimension encodes in-layer token position j.

For Qwen2.5-0.5B-Instruct with LoRA rank $r = 8$, each layer's LoRA matrices have dimensions 8×896. With token size $k = 1024$, we obtain 7 tokens of size 8×128 per layer, with the final token padded to 10×130. For Qwen2.5-1.5B-Instruct with $r = 16$, matrices of size 16×1536 decompose into 6 tokens of 16×256, padded to 18×258. These tokenization schemes balance information density with computational tractability.

E.3 Convolutional Decoder Architecture

The generator employs multi-layer 2D convolutions (inspired by (Liang et al., 2025)) in three categories: width convolution (Conv_W on (C, L)), height convolution (Conv_H on (L, N)), and layer-wise convolution (Conv_L on (N, L)). Each decoder layer contains two Conv_W, two Conv_H, and one Conv_L:

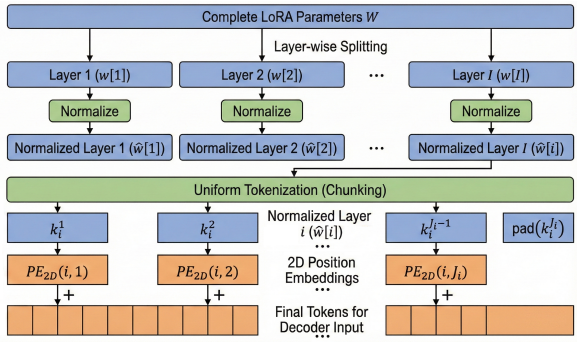
$$\begin{aligned} c_W^\ell &= \text{Conv}_H^1(\text{Conv}_W^1(c^{\ell-1})) \\ c_H^\ell &= \text{Conv}_W^2(\text{Conv}_H^2(c^{\ell-1})), \\ c^\ell &= \text{Conv}_L\left(\frac{c_W^\ell + c_H^\ell + b}{3}\right) \end{aligned} \tag{9}$$

In this work, the entire flow of dimensions is (128,384,384) → (128,200,300) → (128,100,256) → (256,50,200) → (512,50,200) → (1024,25,200) → (1024,10,200) → (2048,10,200) → (4296,8,128) / (4508, 18, 258) with the former for Qwen2.5-0.5B-Instruct and latter for Qwen2.5-1.5B-Instruct

E.4 Training Objective

The generator is trained to minimize mean squared error (MSE) between generated parameters and collected checkpoints. Checkpoints are obtained by: (1) pretraining LoRA adapters on each task for 75 steps at learning rate 10^{-4} with batch size 32 using at max only 5000 training samples, then (2) finetuning for 50 additional steps at learning rate 10^{-5}. Random prompt-checkpoint pairing creates the required training data. The training objective is:

$$\min_\phi \mathbb{E}_{(p_i, m_j)\sim\mathcal{D}_{\text{meta}}} \|\mathcal{G}_\phi(p_i) - m_j\|^2, \tag{10}$$



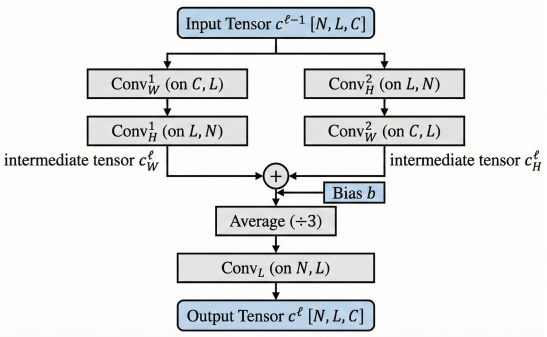


Figure 4: (a) Tokenization of LoRA parameters *(left)* and (b) Generalization Module Architecture *(right)*

where m_j denotes checkpoint parameters and p_i denotes prompt batch embeddings.

E.5 Decoupling via Independent Optimization

Critically, this training procedure optimizes ϕ to capture the **cross-task parameter manifold**-the shared statistical structure of task-adapted parameters across the meta-training distribution-**without encoding adaptation dynamics**. The generator learns what good task-conditioned initializations look like, not how to adapt them. This realizes the generalization component of our decoupled framework, with adaptation handled independently by the RL policy (Appendix F).

F Adaptation Module

The **adaptation module** in DeGAML-LLM is an duplicate instance of the base model which employs a closed-loop RL policy π_ψ that refines generated parameters through task-specific feedback. Unlike coupled meta-learning approaches that commit to a fixed adaptation procedure (e.g., gradient descent) during meta-training, our decoupled framework enables the policy to **flexibly select among diverse adaptation strategies** at test time based on observed task characteristics and performance feedback.

This flexibility arises directly from separating generalization (parameter generation) from adaptation (RL-based refinement): since the generator $\mathcal{G}_\phi$ is optimized independently of any specific adaptation mechanism, the policy π_ψ is free to employ any strategy that improves validation performance-including gradient-based updates, parameter mixing, ensemble voting, or latent-space modifications-without being constrained by the generator's training objective.

F.1 Strategy Design Space

We identify four primary families of adaptation strategies, each addressing different adaptation requirements. These are not exhaustive but represent a design space enabled by our decoupling:

Test-Time Training (TTT)- Minimizes input perplexity on unlabeled task data via gradient steps (Hu et al., 2025a). The model outputs a JSON object with fields: `{ttl_steps, learning_rate, batch_size, shuffle_data}`.

LoRA Subspace Mixing (LoRA)- Blends decomposed LoRA subspaces post-generation. Two-subspace (TS) mixing interpolates magnitude and direction components (Wu et al., 2024), enabling controlled parameter interpolation. The model outputs a JSON object with fields: `{lambda}` for mixing ratio.

Latent Space (LS) Modification- Adjusts hidden activations (representation after the final layer specifically) rather than modifying parameters directly (Hu et al., 2025b). Effective for abstract reasoning tasks where internal token-generation dynamics matter more than surface parameter values. The model outputs a JSON object with fields: `{times, learning_rate}`.

Test-Time Scaling (TTS)- Generates multiple adapters from different prompt batches and aggregates predictions via ensemble voting (max confidence, majority vote, sum logprobs) or selects a single adapter via routing (closest prompt embedding) (Zhang et al., 2025b). Useful for adversarial or high-variance tasks where ensemble robustness improves performance. In the router approach (see Figure 5), we basically sample multiple prompt batches and choose that batch

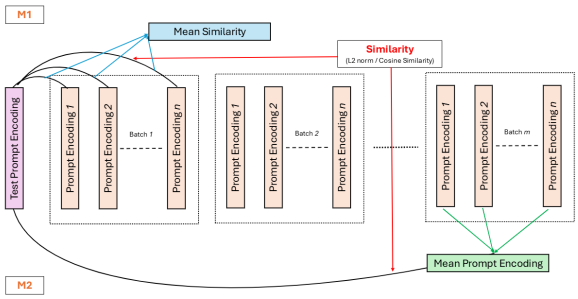


Figure 5: Router Approach for TTS

whose average of similarity scores[6] of individual prompts (M1) or averaged prompt embedding (M2), is closest to that of the question at test time. The corresponding JSON object has fields `num_prompt_batches` (indicating the number of prompt batches to be sampled from the test split of unseen dataset) and `method` which can take one of five values - `avg_sim_score`, `avg_prompt_embed`, `max_confidence`, `majority_vote` or (summing log probabilities) i.e., `sum_logprobs` (former two belong to router approach and the latter three constitute the ensemble approach).

F.2 Strategy Selection via RL

During meta-training, the RL policy π_ψ learns to predict which strategy family and configuration will maximize validation performance for a given task. The policy observes task characteristics through validation metrics and adapts its strategy selection accordingly. Importantly, **strategies are not meta-learned as part of the generator's objective**; rather, they constitute the action space over which the policy is trained via reinforcement learning.

At test time, the policy LLM proposes a strategy configuration based on initial validation performance with the generated parameters. The strategy is then executed (e.g., TTT performs gradient updates, TTS generates multiple adapters), and performance feedback guides further refinement when adaptation is iterative.

[6]Cosine similarity and Euclidean distance were tested and the latter was found to perform better empirically. Thus, `avg_sim_score` and `avg_prompt_embed`. use euclidean distance by default. Alternatively, measure of similarity can also be included as a new field but hasn't been explored in the current work.

F.3 Dataset Descriptions

For in-domain tasks[7], we used ARC-challenge & ARC-easy(Clark et al., 2018) which contains grade-school level, multiple-choice science questions. HellaSwag (Zellers et al., 2019) instructs models to select from choices that best finish the sentence among ground truth and an adversarial set of machine-generated wrong answers. BoolQ (Clark et al., 2019) is a question answering dataset for yes/no questions containing various factual problems. PIQA (Bisk et al., 2019) focuses on everyday situations with a preference for a typical solutions. WinoGrande (Sakaguchi et al., 2021) features a fill-in-a-blank task with binary options for commonsense reasoning questions.

For out-of-domain tasks, we evaluated on GSM-8K (Cobbe et al., 2021)which contains diverse grade-school math word problems requiring multi-step arithmetic reasoning and MATH (Hendrycks et al., 2021) which includes challenging competition-level mathematics problems. DivLogicEval (Chung et al., 2025) tests LLMs' ability to answer counterintuitive natural-language logic questions isolating pure logical inference. SocialIQA (Sap et al., 2019) probes social and emotional understanding of LLMs in everyday interactions. CodeMMLU (Manh et al., 2025) is a multitask benchmark assessing code understanding, including code analysis, defect detection, and software engineering principles across diverse programming tasks. JAMA Clinical Challenge (Chen et al., 2025) features complex real-world medical case questions with expert-written explanations to evaluate clinical reasoning.

F.4 Empirical Strategy Alignments

In our experiments, the adaptation policy learned task-appropriate strategy selections that align intuitively with task characteristics. For instance, here we mention the adaptation families identified for the datasets evaluated

- **ARC-e, JAMA Clinical, PIQA**: TTT (`ttl_steps:25, lr:1e-5`)-perplexity minimization aligns model distributions to domain-specific language
- **ARC-c, SocialIQA**: LS (`times:5, lr:0.1`)-latent steering for abstract reasoning and social intent modeling

[7]By in-domain, we mean that these are the task which has been used during meta-training phase by leave-one out method.

- **BoolQ, GSM-8K, MATH**: LoRA mixing (`lambda:0.5`)-subspace blending for binary questions and mathematical reasoning
- **HellaSwag, WinoGrande, DivLogicEval, CodeMMLU**: TTS (`batches:20`, `method:max_confidence`)-ensemble robustness for adversarial narrative completion and logical inference

These alignments emerge from RL training rather than manual assignment, demonstrating that the policy learns meaningful correlations between task properties and effective adaptation strategies. For instance,

- ARC-e contains multiple-choice science questions where adapting on unlabeled in-domain inputs (minimizing perplexity or self-supervision) helps tune local decision boundaries quickly. TTT can correct surface token probabilites without heavy structural changes.
- JAMA Clinical is jargon-heavy and benefits from adjusting to the specific distribution of clinical language at test time; TTT using unlabeled examples can quickly align token distributions.
- ARC-c contains harder, more abstract science problems where success depends on internal reasoning strategies (abstraction, multi-hop reasoning). Changing hidden activations or decoding/sampling behavior (LS family) can adjust internal token-generation dynamics that drive such abstraction, making LS assignment plausible.
- Social reasoning involves latent constructs like intent, emotion, social norms which are not surface tokens but patterns in activations. Latent-space steering is a natural fit making it a good call by the policy LLM for SocialIQA.
- HellaSwag is about picking the most plausible narrative continuation among adversarial choices. Using multiple adapters (test-time scaling / ensemble) and picking the max-confidence or majority vote-based answer is a good fit because different adapters capture different narrative priors and thus ensemble voting stabilizes adversarial options.

This validates the core premise of decoupled meta-learning: by separating generalization from adaptation, we enable flexible, task-driven strategy selection unavailable in coupled approaches.

G Ablation Studies

To understand the effectiveness of each component in our decoupled framework, we conduct ablation studies comparing: (1) Base LLM with no adaptation, (2) Generalization module only (generator-produced parameters without RL refinement), (3) Full DeGAML-LLM (generator + RL adaptation). Results in Tables 2 demonstrate that both modules contribute to performance:

- **Generalization module alone** provides substantial gains over the base model (avg +22.9 points in-domain, +4.9 out-of-domain for 0.5B), confirming that task-conditioned parameter generation captures useful cross-task structure.
- **Adding RL adaptation** further improves performance (additional +8.3 in-domain, +27.8 out-of-domain for 0.5B and +12.6 in-domain, +33.1 out-of-domain for 1.5B), validating the benefit of task-specific refinement through closed-loop feedback.
- **Decoupling is essential**: The two-stage approach outperforms end-to-end coupled training because the generator focuses solely on cross-task structure while the policy handles task-specific dynamics.

The larger gains from adaptation on out-of-domain tasks (GSM-8K, MATH, Medical) suggest that RL refinement is particularly valuable when the task distribution differs significantly from meta-training, as the policy can discover task-appropriate strategies (e.g., TTT for distribution shift, ensemble for adversarial robustness) beyond what the generator alone provides.