MARD: Module-Aware Reasoning Distillation for Language Models with Adaptive Supervision

Wenqi Yang[1], Jianjun Li[1]*, Zhibo Zhang[1], Mingqiang Ding[1], Yushen Fang[1]
[1]School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, China
{yangwenqi, jianjunli, zhibo_zhang, mingqianding, yushen_fang}@hust.edu.cn

Abstract

Multi-step reasoning remains challenging for language models with limited capacity. While recent reasoning distillation approaches transfer chain-of-thought supervision from large teacher models, they typically apply uniform supervision across all Transformer components, overlooking the fact that different modules contribute unequally to reasoning. We propose Module-Aware Reasoning Distillation, a parameter-efficient framework that explicitly targets key Transformer components for effective reasoning transfer. Through systematic analysis, we identify the feed-forward network projections and the output projection of self-attention as primary bottlenecks for reasoning. Based on these findings, we introduce lightweight adapter modules at these components while freezing the backbone parameters, enabling focused and efficient distillation. Our approach adopts an offline distillation setting, where a strong teacher model provides reasoning trajectories in advance, and incorporates an adaptive supervision strategy that adjusts the strength of reasoning-related losses according to problem difficulty. Experiments on mathematical reasoning benchmarks demonstrate consistent improvements over strong baselines, and ablation studies confirm the importance of both module-aware placement and adaptive supervision. Our code is available at https://github.com/wqyang24/MARD.

1 Introduction

Large language models (LLMs) have demonstrated remarkable reasoning capabilities across diverse tasks(Huang and Chang, 2023; Li et al., 2024b; Ning et al., 2024; Cheng et al., 2025; Liu et al., 2025a), including arithmetic problem solving, multi-hop question answering, and symbolic reasoning (Zhou et al., 2025; Hu et al., 2025; Dong et al., 2025; Liu et al., 2025b; Fang et al., 2020; Ranaldi et al., 2025). However, these strong performances are typically achieved by models with billions of parameters, which hinders their deployment in resource-constrained environments(Fu et al., 2023; Li et al., 2025b; Bi et al., 2025; Wang et al., 2025). A key research challenge, therefore, is to effectively transfer the reasoning abilities of large teacher models to significantly smaller, more efficient student models without substantial performance loss (Kang et al., 2023; Yang et al., 2024; Feng et al., 2024; Tian et al., 2025).

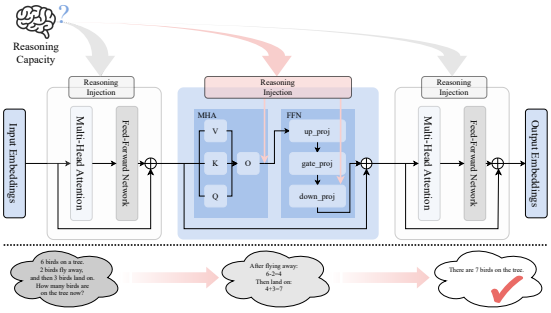

Figure 1: Reasoning capability in Transformer-based models is not uniformly distributed. Rather than being evenly contributed by all components, reasoning in large Transformer models appears to concentrate in a subset of layers and modules, which motivates locating reasoning-critical modules as a prerequisite for effective reasoning enhancement.

Conventional knowledge distillation methods primarily rely on aligning the final output distributions of the teacher and student models (Chen et al., 2025; He et al., 2022; Shridhar et al., 2023). While effective for general language modeling, such output-level alignment often fails to capture the *internal reasoning mechanisms* of LLMs. Crucially, a growing body of evidence suggests that reasoning is not uniformly encoded across the network; rather, it is localized within specific layers and modules (Shao and Wu, 2025; Li et al., 2025a).

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 37680–37695
July 2-7, 2026 ©2026 Association for Computational Linguistics

This non-uniformity implies that indiscriminately distilling knowledge from all components is inefficient, as it forces the student to mimic teacher behaviors irrelevant to reasoning, thereby introducing unnecessary computational overhead and potentially obscuring the core reasoning signals (Xu et al., 2025b; Sharma and Chopra, 2025; Qian et al., 2025; Chatziveroglou et al., 2025).

To address this fundamental limitation, we introduce **MARD**, a distillation framework designed to **explicitly transfer localized reasoning capabilities**. Our approach is built on two core insights derived from an analysis of LLM internals. First, multi-step reasoning is concentrated in a critical subset of Transformer layers and functional components. Second, problems of varying complexity exert heterogeneous demands on these components. Based on these insights, MARD incorporates: (1) **explicit supervision of selected internal representations** at structurally critical points in the student model via lightweight adapters, enabling precise injection of reasoning knowledge without modifying the backbone architecture; and (2) a **difficulty-adaptive meta-optimization strategy** that dynamically modulates the strength of module-level supervision based on problem complexity, ensuring adaptive knowledge transfer across different reasoning regimes. By integrating output-level alignment with targeted module-level representation matching and adaptive weighting, MARD effectively distills reasoning capabilities from large teachers into compact models, even in the sub-10B parameter regime. Our contributions are summarized as follows:

- We systematically analyze and characterize the localization of multi-step reasoning in large language models, identifying the critical Transformer layers and components where reasoning computations predominantly reside. Leveraging this finding, we propose a novel reasoning injection mechanism that directly supervises these key modules in the student model.

- We observe that reasoning complexity heterogeneously affects different model components. To exploit this, we introduce a difficulty-adaptive distillation strategy that meta-optimizes and dynamically adjusts module-level supervision strength according to problem complexity, enabling more nuanced and effective knowledge transfer.

- Through extensive experiments on diverse reasoning benchmarks, we demonstrate that MARD consistently and effectively transfers multi-step reasoning capabilities from large teachers to compact students. Our analyses validate the complementary benefits of explicit module-aware supervision and difficulty-adaptive optimization.

Despite increasing evidence for the non-uniform encoding of reasoning in LLMs, a systematic understanding of where multi-step reasoning occurs within the Transformer architecture remains underdeveloped. Such an understanding is essential for designing principled distillation methods that selectively transfer reasoning-relevant knowledge, rather than applying indiscriminate supervision. To bridge this gap and establish a foundation for our method, we first conduct an internal analysis to pinpoint the layers and modules that dominate reasoning processes. The following section details our investigation into where reasoning happens in large language models.

2 Where Does Reasoning Happen in LLMs?

To pinpoint which components of Transformer-based language models are most critical for reasoning (Havrilla et al., 2024; Hao et al., 2024; Li et al., 2025a; Xu et al., 2025a), we analyze parameter discrepancies between each base Qwen model and its counterpart distilled using DeepSeek-R1 (Guo et al., 2025) across four model scales: 1.5B, 7B, 14B, and 32B. Specifically, we compute the layer-wise ℓ_2-norm differences of all projection matrices, visualized in the heatmaps in Figure 2.

A consistent pattern emerges: the most significant parameter differences occur in the feed-forward network (FFN) layers and the self-attention output projection (*o_proj*). In contrast, projections related to query, key, and value transformations show substantially smaller variation. This suggests that distillation primarily reshapes how intermediate representations are transformed and aggregated, rather than how local token dependencies are initially captured.

Notably, the relative importance of FFN and *o_proj* shifts with model scale. In smaller models, the largest discrepancies concentrate in FFN

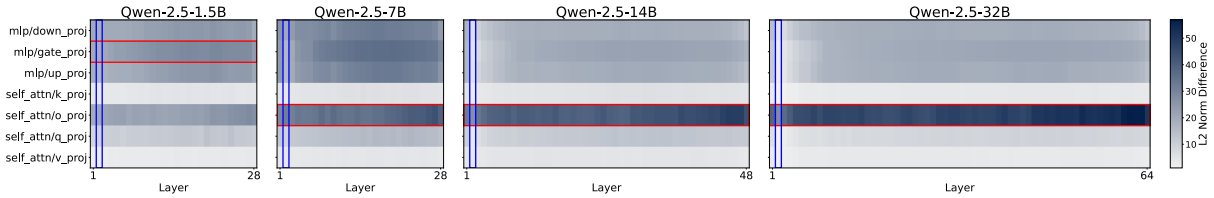

Figure 2: Layer-wise ℓ_2-norm difference heatmaps between base Qwen models and their DeepSeek-R1 distilled counterparts across four scales (1.5B, 7B, 14B, and 32B). Discrepancies are most pronounced in the feed-forward network (FFN) layers and the self-attention output projection (*o_proj*); red boxes indicate modules with the largest mean differences, while blue boxes denote those with the smallest.

projections. Given the limited representational capacity of attention modules at this scale, FFN layers appear to act as a compensatory mechanism, enabling nonlinear restructuring of hidden states to approximate the teachers reasoning behavior more closely. As model size increases, however, the dominant discrepancies gradually shift toward *o_proj*. Larger models possess more expressive attention subspaces, allowing distillation to more directly modify how multi-head attention outputs are combineda crucial operation for coordinating multi-step reasoning.

Across all scales, we also observe a clear depth-dependent pattern. Shallow layers exhibit relatively minor differences between base and distilled models, while deeper layers show substantially larger discrepancies. This aligns with the known functional hierarchy of Transformer architectures: lower layers predominantly encode lexical and syntactic information, which remains largely consistent across different training objectives, whereas deeper layers are responsible for integrating information over longer contexts and refining intermediate reasoning states. Consequently, distillation-induced adjustments naturally concentrate in these deeper, reasoning-active layers.

In summary, our analysis reveals that reasoning capability is not uniformly distributed throughout the model architecture. Instead, it is governed predominantly by a subset of modulesspecifically, FFN layers and attention output projections in deeper Transformer blocks. Moreover, the relative contribution of these modules is scale-dependent: FFN layers play a more prominent role in smaller models, while *o_proj* becomes increasingly critical as model capacity grows. These findings provide a principled motivation for selectively adapting a small set of reasoning-critical modules, laying the groundwork for our module-aware distillation approach using parameter-efficient adapters.

3 Method

Building upon the finding that reasoning is unevenly distributed across Transformer components (Section 2), we introduce MARD, a Module-Aware Reasoning Distillation framework designed for interpretable and efficient knowledge transfer. We first formalize our objectives and then detail its three core components: (1) selective adapter injection guided by teacher saliency, (2) a composite distillation objective, and (3) a meta-learned controller for difficulty-adaptive supervision.

3.1 Problem Formulation and Interpretability Objective

Given a large teacher model T and a compact student model S, our goal is to transfer T's multi-step reasoning capability to S. The teacher, given an input x, produces an output $p^{(T)}(y|x)$, potentially with a chain-of-thought trace. Beyond performance, we prioritize *interpretability*, aiming to mechanistically answer the questions raised in the Introduction:

- **Where** is reasoning computation localized within the Transformer?
- **How** do module contributions vary with reasoning difficulty?
- **How can** such structured signals be selectively transferred to a smaller model?

Formally, our goal is to endow the student with improved reasoning ability *without modifying its backbone*, relying instead on module-aware supervision, selective parameter augmentation, and difficulty-adaptive allocation of learning signals.

3.2 Module-Aware Adapter Injection

We augment a standard Transformer backbone with lightweight bottleneck adapters inserted at

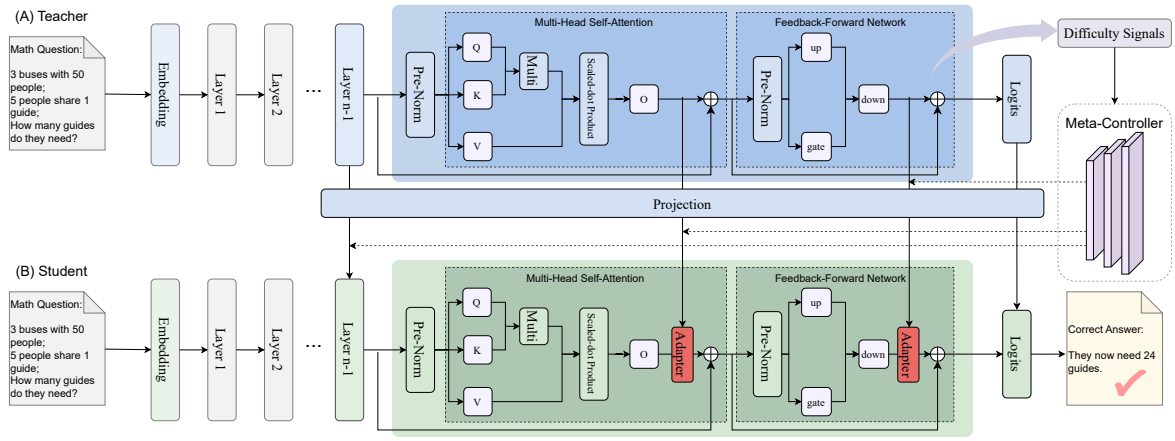

Figure 3: Overview of the proposed interpretable module-aware reasoning distillation framework. Teacher-derived reasoning saliency signals identify layers and modules where reasoning-related computation is concentrated. Lightweight adapters are selectively injected into attention and FFN residual write-back points of the student, and supervised via token-level and module-level distillation. A meta-learned controller dynamically reweights attention and FFN supervision based on input difficulty, enabling capacity-aware and interpretable transfer of multi-step reasoning ability without modifying the student backbone architecture.

residual write-back points, positions where transformed information is integrated back into the residual stream. Adapters are placed after two key modules: (i) the attention output projection (`o_proj`), and (ii) the FFN down-projection (`down_proj`).

Formally, let $\mathbf{h}^{(l)}$ denote the residual stream input to layer l. The attention and FFN sublayers produce intermediate representations $\mathbf{z}_{\text{attn}}^{(l)}$ and $\mathbf{z}_{\text{ffn}}^{(l)}$, respectively. The output of layer l is then computed as:

$$\tilde{\mathbf{h}}^{(l)} = \mathbf{h}^{(l)} + \mathbf{A}_{\text{attn}}^{(l)}(\mathbf{z}_{\text{attn}}^{(l)}) + \mathbf{A}_{\text{ffn}}^{(l)}(\mathbf{z}_{\text{ffn}}^{(l)}), \quad (1)$$

where $\mathbf{A}_{\text{attn}}^{(l)}$ and $\mathbf{A}_{\text{ffn}}^{(l)}$ are low-rank bottleneck adapters.

Adapters are selectively injected, not applied to all layers. We target layers identified as reasoning-relevant according to teacher-derived signals, based on the hypothesis that reasoning processes are localized rather than uniformly distributed across the network.

3.3 Teacher-Derived Reasoning Saliency Signals

To locate where reasoning computations occur within the teacher model, we extract internal representations from both the attention and FFN submodules at each layer:

$$\mathbf{h}_{\text{attn},l}^{(T)}(x), \quad \mathbf{h}_{\text{ffn},l}^{(T)}(x). \quad (2)$$

Rather than manually assigning fixed weights to different modules, we define a scale-normalized, module-aware reasoning saliency score that accounts for the intrinsic magnitude differences between attention and FFN activations. Specifically, we first compute the expected activation norms over the training set:

$$\begin{aligned} \mu_{\text{attn}} &= \mathbb{E}_{x,l}\left[\left\|\mathbf{h}_{\text{attn},l}^{(T)}(x)\right\|_2\right], \\ \mu_{\text{ffn}} &= \mathbb{E}_{x,l}\left[\left\|\mathbf{h}_{\text{ffn},l}^{(T)}(x)\right\|_2\right]. \end{aligned} \quad (3)$$

Using these statistics, we define the reasoning saliency at layer l for input x as:

$$s_l(x) = \frac{\left\|\mathbf{h}_{\text{attn},l}^{(T)}(x)\right\|_2}{\mu_{\text{attn}}} + \frac{\left\|\mathbf{h}_{\text{ffn},l}^{(T)}(x)\right\|_2}{\mu_{\text{ffn}}}. \quad (4)$$

This formulation adaptively normalizes module contributions based on teacher-side activation statistics, ensuring that the saliency measure reflects relative computational engagement rather than absolute scale differences. Notably, we emphasize that activation magnitude alone does not fully characterize reasoning. Rather, this saliency score provides a simple, scalable, and teacher-agnostic proxy for computational engagement under reasoning supervision. Empirically, we observe that FFN activations in higher layers dominate the score, supporting the hypothesis that deeper nonlinear transformations underlie multi-step reasoning.

3.4 Interpretable Layer Selection for Student Supervision

Teacher-derived reasoning saliency offers a principled signal for identifying layers where reasoning-related computation is concentrated. We first compute the expected saliency for each teacher layer by averaging over the training data:

$$\bar{s}_l = \mathbb{E}_{x \sim \mathcal{D}_{\text{train}}}[s_l(x)], \tag{5}$$

yielding a depth-wise saliency profile that reflects the uneven distribution of reasoning activity across layers.

Guided by this profile, we select a subset of teacher layers with consistently high saliency values. Empirically, these layers often form a contiguous band in the middle-to-upper region of the teacher model, rather than being confined to the final blocks, supporting the hypothesis that multi-step reasoning is localized, not uniformly distributed along depth.

Saliency-Mass Preserving Layer Alignment. To transfer this structural pattern to the student model, we align teacher and student layers by preserving cumulative reasoning saliency mass, instead of relying on simple depth matching. This approach explicitly accounts for the non-uniform allocation of reasoning across layers.

We first define a normalized cumulative saliency function over teacher layers:

$$F_T(l) = \frac{\sum_{i=1}^{l} \bar{s}_i}{\sum_{j=1}^{L_T} \bar{s}_j}, \tag{6}$$

where L_T denotes the total number of layers in the teacher model, and $F_T(l) \in (0, 1]$ represents the fraction of total teacher-side reasoning computation accumulated up to layer l.

For the student model with L_S layers, we define a depth-based cumulative distribution as a structural prior:

$$F_S(k) = \frac{k}{L_S}, \qquad k \in \{1, \ldots, L_S\}. \tag{7}$$

Each selected teacher layer l_T is then mapped to a student layer l_S by matching cumulative distributions:

$$l_S = \arg \min_{k \in \{1,\ldots,L_S\}} \left| F_S(k) - F_T(l_T) \right|. \tag{8}$$

This mapping preserves the relative allocation of reasoning computation across depth while accommodating architectural differences between teacher and student models.

The resulting set of student layers is denoted as $\mathcal{L}_S$. Module-aware adapters are inserted, and module-level distillation losses are applied exclusively to layers in $\mathcal{L}_S$. All remaining layers are trained only via standard token-level supervision, avoiding unnecessary parameter overhead and supervision noise in low-saliency regions.

Notably, this design decouples *where* reasoning supervision is applied from *how strongly* it is applied. Layer selection is determined by teacher-derived saliency as a stable, input-agnostic structural signal, while the strength of module-level supervision is dynamically modulated by the difficulty-adaptive controller introduced in the following subsection.

3.5 Student Distillation Objectives

The student model is supervised through a combination of behavioral and mechanistic distillation objectives.

Token-Level Distillation. The students token-level output distribution $p^{(S)}(y_t|x)$ is aligned with the teachers distribution $p^{(T)}(y_t|x)$ via the KL divergence:

$$\mathcal{L}_{\text{token}}(x) = \sum_{t=1}^{T} \text{KL}(p^{(T)}(y_t|x) \parallel p^{(S)}(y_t|x)), \tag{9}$$

where T denotes the sequence length.

Module-Level Distillation. For each supervised student layer $s \in \mathcal{L}_S$, we align internal representations with the teachers corresponding layer outputs:

$$\mathcal{L}_{\text{attn}}^{(s)}(x) = \left\| \mathbf{h}_{\text{attn},s}^{(S)}(x) - \text{P}(\mathbf{h}_{\text{attn},l_T(s)}^{(T)}(x)) \right\|_2^2, \tag{10}$$

$$\mathcal{L}_{\text{ffn}}^{(s)}(x) = \left\| \mathbf{h}_{\text{ffn},s}^{(S)}(x) - \text{P}(\mathbf{h}_{\text{ffn},l_T(s)}^{(T)}(x)) \right\|_2^2, \tag{11}$$

where $\text{P}(\cdot)$ is a linear projection applied when the teacher and student feature dimensions differ.

Overall Objective. The complete student loss is a weighted combination:

$$\begin{aligned} \mathcal{L}_{\text{student}}(x) &= \lambda_{\text{token}} \mathcal{L}_{\text{token}}(x) \\ &+ \sum_{s \in \mathcal{L}_S} (\lambda_{\text{attn}}^{(s)} \mathcal{L}_{\text{attn}}^{(s)}(x) + \lambda_{\text{ffn}}^{(s)} \mathcal{L}_{\text{ffn}}^{(s)}(x)), \end{aligned} \tag{12}$$

where λ_{token}, $\lambda_{\text{attn}}^{(s)}$, and $\lambda_{\text{ffn}}^{(s)}$ are balancing coefficients that regulate the strength of each supervision signal.

3.6 Meta-Learned Controller as an Explanatory Model

Different inputs impose different reasoning demands. To dynamically allocate supervision strength, we introduce a meta-learned controller $\mathcal{M}_\psi$ that predicts module-wise loss weights conditioned on the teacher's intermediate activations:

$$\lambda(x) = \mathcal{M}_\psi(\{\mathbf{h}^{(T)}_{\text{attn},l}(x), \mathbf{h}^{(T)}_{\text{ffn},l}(x)\}_{l=0}^{L_T-1}). \quad (13)$$

Beyond optimization, the controller also provides an interpretable decomposition of reasoning difficulty into module-specific learning demands, offering a transparent window into what makes certain examples more challenging.

Controller Architecture. $\mathcal{M}_\psi$ is implemented as a lightweight neural network that produces normalized supervision weights via a softmax:

$$\begin{aligned}&\lambda^{(s)}_{\text{attn}}(x), \lambda^{(s)}_{\text{ffn}}(x) \\ &= \text{softmax}\big(f_\psi(\mathbf{h}^{(T)}_{\text{attn},l_T(s)}, \mathbf{h}^{(T)}_{\text{ffn},l_T(s)})\big).\end{aligned} \quad (14)$$

where $l_T(s)$ denotes the teacher layer aligned with student layer s.

Bi-level Optimization. The controller is trained using a bi-level optimization scheme that separates student parameter updates from controller parameter updates:

$$\theta_S \leftarrow \theta_S - \eta \nabla_{\theta_S} \mathcal{L}_{\text{student}}(x; \psi), \quad (15)$$

$$\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_{\text{val}}\big(\theta_S(\psi)\big), \quad (16)$$

where $\mathcal{L}_{\text{val}}$ denotes the student validation loss. This formulation enables the controller to learn which modules are most critical for different inputs, transforming loss reweighting into a transparent mechanism for analyzing reasoning complexity.

3.7 Overall Training Procedure

Our proposed training pipeline integrates three key components: module-aware adapter injection, teacher-guided reasoning saliency estimation, and a meta-learned controller for difficulty-adaptive supervision. The full procedure consists of the following three stages: 1) **Saliency Extraction** computing teacher-layer reasoning saliency from training data; 2) **Layer & Adapter Initialization** selecting student layers and injecting lightweight adapters based on saliency alignment; and 3) **Joint Bilevel Optimization** cotraining the student and the meta-controller with adaptive supervision weights. Due to space constraints, the complete pseudocode is provided in Appendix A.

4 Experiments

In this section, we conduct a systematic experimental evaluation of our Interpretable Difficulty-Adaptive Module-Aware Reasoning Distillation framework. We assess both overall reasoning performance and the mechanisms by which reasoning capability is transferred within Transformer models, including selective supervision of reasoning-relevant layers, module-level adapter injection, and difficulty-adaptive reallocation of distillation signals. Experiments are conducted across diverse mathematical, logical, and multistep reasoning benchmarks, comparing our approach against standard knowledge distillation, chainofthought distillation, and parameterefficient finetuning baselines. We further provide ablation and interpretability analyses to characterize the resulting layerwise, modulewise, and inputdependent behaviors.

4.1 Experimental Setup

Datasets. We evaluate our method on a set of benchmarks that require multi-step mathematical and logical reasoning, including GSM8K, SVAMP, LogiQA, and StrategyQA. All datasets follow their official train–validation–test splits, and performance is measured using exact-match or classification accuracy, depending on the task. Detailed descriptions of the datasets, including task characteristics, dataset statistics, and evaluation protocols, are provided in Appendix B.

Models. We adopt **Qwen-2.5-Math-1.5B** and **Qwen-2.5-Math-7B** as student models, representing capacity-constrained Transformer architectures with strong mathematical pretraining. As teachers, we use **DeepSeek-R1-Distill-Qwen-14B** and **DeepSeek-R1-Distill-Qwen-32B**, whose architectures are closely aligned with the students, enabling meaningful layer-wise and module-wise alignment during distillation. We intentionally avoid substantially larger or architecturally dissimilar teacher models, as such choices would introduce representation mismatch and confound the analysis of where and how reasoning-related computation is transferred.

Baselines. We compare against a diverse set of recent baselines in chainofthought prompting, selfrefinement, and reasoningoriented distillation:

Method	# Params	Distillation Teachers	Mathematical		Commonsense	Logical
			GSM8K	SVAMP	StrategyQA	LogiQA
Direct LLM Evaluation						
GPT-3.5-Turbo	175B	–	78.01	82.30	70.92	40.55
Qwen-2.5-14B	14B	–	65.87	71.23	63.45	42.18
Qwen-2.5-32B	32B	–	72.14	78.01	68.32	44.56
Llama-3.1-70B	70B	–	76.98	81.45	70.87	46.32
Distilled Student Model Evaluation						
T5-XXL+CoT (Magister et al., 2023)	11B	PaLM, GPT-3	21.99	25.37	63.77	30.28
GPT-J+Self-Reflection (Wang et al., 2023)	6B	ChatGPT	33.10	55.21	65.87	36.94
ORCA2-7B (Mitra et al., 2023)	7B	ChatGPT, GPT-4	47.23	53.18	50.12	35.02
CodeT5-Large+PaD (Zhu et al., 2024)	770M	GPT-3.5-Turbo	44.90	51.23	50.47	34.89
Llama-7B+NCE (Li et al., 2024a)	7B	GPT-3.5-Turbo, GPT-4	41.93	51.52	52.11	33.67
Llama2-7B+ReversalMath (Guo et al., 2024)	7B	GPT-4	52.10	59.27	55.18	38.14
Qwen2.1.5B+SIKeD (Adarsh et al., 2025)	1.5B	Llama3-70B	64.97	75.43	60.21	40.18
Llama3.1-8B+ReDistill (Jia et al., 2025)	8B	DeepSeek-R1	75.66	82.04	68.33	45.29
The Proposed MARD						
Qwen-2.5-Math-1.5B	1.5B	DeepSeek-Distill-Qwen-14B	66.12	72.05	63.89	42.90
Qwen-2.5-Math-1.5B	1.5B	DeepSeek-Distill-Qwen-32B	68.34	74.12	65.33	44.27
Qwen-2.5-Math-7B	7B	DeepSeek-Distill-Qwen-14B	71.50	76.33	66.87	44.90
Qwen-2.5-Math-7B	7B	DeepSeek-Distill-Qwen-32B	73.88	78.55	68.91	46.22

Table 1: Performance comparison of direct LLM evaluation, distilled student models, and our student–teacher distillation experiments across multiple reasoning benchmarks.

- **T5XXL+CoT** (Magister et al., 2023) and **GPTJ+SelfReflection** (Wang et al., 2023) leverage explicit reasoning traces and iterative refinement.
- **ORCA27B** (Mitra et al., 2023) uses explanationrich supervision for reasoning generalization.
- **CodeT5Large+PaD** (Zhu et al., 2024) incorporates programaided decomposition for mathematical problems.
- **Llama7B+NCE** (Li et al., 2024a) and **Llama27B+ReversalMath** (Guo et al., 2024) employ contrastive or curriculumstyle objectives tailored for math reasoning.
- **Qwen2.1.5B+SIKeD** (Adarsh et al., 2025) and **Llama3.18B+ReDistill** (Jia et al., 2025) propose reasoningaware distillation that selectively transfers teacher signals.

Note unlike these approaches, which typically apply uniform or instancelevel supervision, our method explicitly targets reasoningcritical layers and modules, and adaptively modulates supervision strength based on problem difficulty. Full baseline descriptions are provided in Appendix C. For completeness and fair comparison, we further report additional results with parameter-matched uniform adapter and LoRA baselines under a frozen-backbone setup in Appendix D.

Training and Evaluation. Student backbone parameters remain frozen throughout training. All methods are trained under identical optimization settings to ensure fair comparison. Detailed hyperparameters and configurations are given in Appendix E. In addition, we report a detailed efficiency analysis of the proposed method, including adapter parameter counts, incremental FLOPs, and measured training and inference overhead. The corresponding statistics and measurements are presented in Appendix E.3.

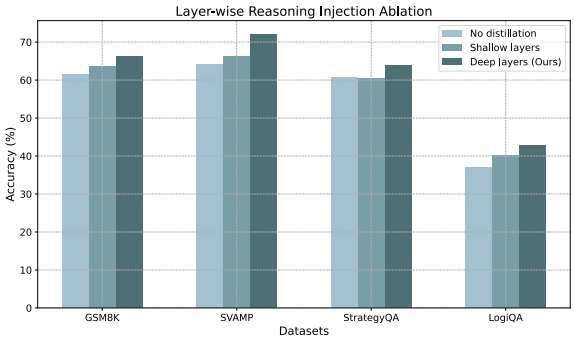

Figure 4: Layer-wise ablation: deep-layer distillation improves performance, while shallow-layer injection yields limited effects.

4.2 Main Results

Table 1 summarizes the performance of our method compared with direct LLM evaluation and a range of distilled student models across mathematical, commonsense, and logical reasoning benchmarks.

Overall, MARD consistently achieves strong performance while maintaining substantially smaller student models. For example, the **1.5B**

student trained with MARD reaches **68.34** on GSM8K and **74.12** on SVAMP, outperforming several previously reported distilled models with comparable or larger parameter counts, such as ORCA2-7B, Llama-7B+NCE, and Llama2-7B+ReversalMath. These results indicate that the proposed reasoning-aware distillation mechanism can effectively transfer multi-step reasoning capability into compact student models.

Importantly, performance improvements become more evident when comparing models of similar scale. For instance, the **Qwen-2.5-Math-7B** student trained with MARD achieves **73.88** on GSM8K and **78.55** on SVAMP, which is competitive with or exceeds other distilled models in the 7B8B range. At the same time, MARD maintains strong performance on non-mathematical reasoning tasks, reaching **68.91** on StrategyQA and **46.22** on LogiQA, indicating that the transferred reasoning capability generalizes beyond purely mathematical domains.

It is worth noting that some baseline methods achieve higher scores in individual columns (e.g., Llama3.1-8B+ReDistill on GSM8K). However, these models typically rely on larger student architectures or different distillation settings, making direct parameter-matched comparison difficult. In contrast, MARD focuses on improving reasoning transfer efficiency under constrained student capacity, achieving competitive results while using smaller or comparable models.

The observed gains are consistent with the design principles of our framework. First, reasoning supervision is injected into carefully selected layers, enabling the student to capture multi-step reasoning patterns more efficiently. Second, supervision strength is allocated adaptively based on problem difficulty, allowing different network modules to specialize according to task complexity. Together, these mechanisms promote better internalization of teacher reasoning signals and improved generalization.

Finally, we further evaluate the robustness of the framework under heterogeneous teacherstudent architectures (e.g., Llama teacher to Qwen student). The detailed results, reported in Appendix F, show that MARD maintains competitive performance with only minor degradation, suggesting stable cross-architecture transfer.

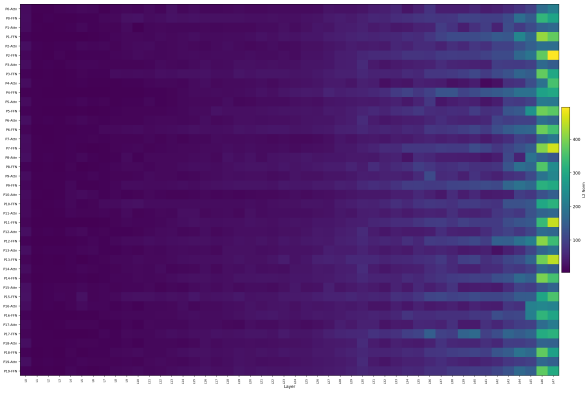

Figure 5: Instance-level module-wise activation heatmaps for 20 individual GSM8K problems.

4.3 Ablation: Layer-wise Reasoning Injection

To validate our layer selection strategy for reasoning injection, we conduct a targeted layer-wise ablation study that compares different adapter placement schemes in the student model. Specifically, we evaluate three representative settings: 1) inserting adapters into the first three Transformer layers; 2) inserting adapters into the last three Transformer layers; and 3) inserting adapters into layers selected by our saliency-guided method. Results in Figure 4 show that injecting supervision into shallow layers yields only marginal or inconsistent improvements, confirming that these layers, primarily responsible for lexical and syntactic processing, are not effective loci for multistep reasoning. Deeplayer injection consistently outperforms shallowlayer placement, underscoring the greater relevance of deeper layers for reasoningrelated computation.

Nevertheless, our saliencyguided selection achieves the best overall performance across all tasks, surpassing the lastlayer baseline while using the same number of adapters. This indicates that reasoningcritical computation is not strictly confined to the final blocks. As illustrated in Figure 5 (activation heatmaps from 20 random GSM8K problems), reasoningrelated activity typically spans a contiguous band in the middletoupper layers, distributed across both intermediate and deep layers.

Taken together, these findings demonstrate that while deeper layers are generally more relevant for reasoning than shallow ones, restricting supervision solely to the final layers is suboptimal. By aligning adapter placement with teacherderived reasoning saliency, our method more precisely tar-

gets the layers actively involved in multistep reasoning, leading to more effective and parameterefficient knowledge transfer.

4.4 Ablation: Module-aware Distillation within Attention

We further investigate the effect of inserting distillation adapters into different attention submodulesnamely the query (Q), key (K), value (V), and output (O) projectionsusing a 1.5B-parameter student model evaluated on the GSM8K. For each position, we compare two variants: applying the adapter only to the attention submodule, and applying the adapter to both the attention submodule and the subsequent FFN layer. Figure 6 shows that FFN supervision is essential, with O+FFN yielding the largest gains, highlighting the importance of combining attention outputs with FFN transformations for effective reasoning distillation.

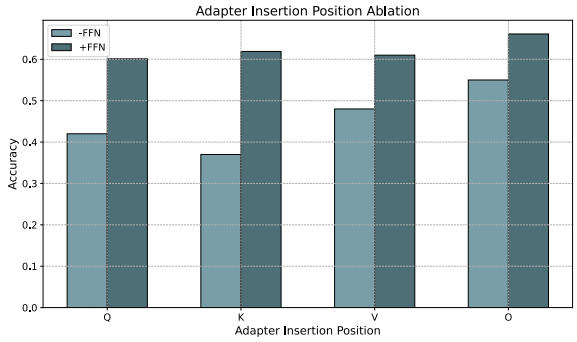

Figure 6: Ablation study of adapter insertion within different attention submodules with and without FFN.

5 Conclusion

In this work, we propose MARD, a framework that transfers structured, multi-step reasoning from large teachers to compact students. MARD combines module-aware localization with difficulty-adaptive supervision: it identifies reasoning-critical layers via teacher saliency and injects lightweight adapters, while a meta-learned controller dynamically reweights distillation signals based on input complexity. Experiments on mathematical and logical reasoning benchmarks show that MARD outperforms strong fine-tuning and reasoning-aware distillation baselines with minimal overhead. Beyond performance gains, the framework offers interpretable insights into *where* reasoning is structured and *how* supervision should be allocated, providing a general pathway toward capable yet compact reasoning models.

Limitations

Despite the promising results, our approach has several limitations. We only evaluate student and teacher models with architectures similar to Qwen and DeepSeek, and the generalization to substantially different model families or non-Transformer architectures remains unexplored. Our experiments are also primarily focused on mathematical reasoning tasks, leaving open questions about the effectiveness of module-aware and difficulty-adaptive distillation in other reasoning domains or natural language understanding benchmarks. Additionally, the framework relies on access to high-capacity teachers to extract layer-wise activation signals, which may not be practical in scenarios with resource constraints. The inclusion of a meta-learned controller introduces further training complexity through bi-level optimization, increasing computational and memory requirements compared to standard distillation. Finally, although the activation-based analysis and controller weights provide interpretable insights into reasoning computation, they are still proxy measures and may not fully capture the internal mechanisms underlying multi-step reasoning. Addressing these limitations in future work would involve exploring more diverse student and teacher architectures, extending evaluation to broader reasoning tasks, and developing more efficient or robust controller designs.

References

Shivam Adarsh, Kumar Shridhar, Nicholas Monath, and Mrinmaya Sachan. 2025. Siked: Self-guided iterative knowledge distillation for mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9868–9880.

Jing Bi, Yuting Wu, Weiwei Xing, and Zhenjie Wei. 2025. Enhancing the reasoning capabilities of small language models via solution guidance fine-tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9074–9084.

Giannis Chatziveroglou, Richard Yun, and Maura Kelleher. 2025. Exploring llm reasoning through controlled prompt variations. *arXiv preprint arXiv:2504.02111*.

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, and 1 others. 2025. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15094–15119.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652*.

Yihong Dong, Yuchen Liu, Xue Jiang, Bin Gu, Zhi Jin, and Ge Li. 2025. Rethinking repetition problems of llms in code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 965–985.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 8823–8838.

Tao Feng, Yicheng Li, Li Chenglin, Hao Chen, Fei Yu, and Yin Zhang. 2024. Teaching small language models reasoning through counterfactual distillation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5831–5842.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Pei Guo, Wangjie You, Juntao Li, Yan Bowen, and Min Zhang. 2024. Exploring reversal mathematical reasoning ability for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13671–13685.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, and 1 others. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

Xingwei He, Yeyun Gong, A-Long Jin, Weizhen Qi, Hang Zhang, Jian Jiao, Bartuer Zhou, Biao Cheng, Sm Yiu, and Nan Duan. 2022. Metric-guided distillation: Distilling knowledge from the metric to ranker and retriever for generative commonsense reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 839–852.

Lanxiang Hu, Tajana Rosing, and Hao Zhang. 2025. Trimllm: Progressive layer dropping for domain-specific llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 667–681.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pages 1049–1065.

Jinghan Jia, Hadi Reisizadeh, Chongyu Fan, Nathalie Baracaldo, Mingyi Hong, and Sijia Liu. 2025. Epic: Towards lossless speedup for reasoning training through edge-preserving cot condensation. *arXiv preprint arXiv:2506.04205*.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. *Advances in Neural Information Processing Systems*, 36:48573–48602.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025a. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Bin Sun, Xinglin Wang, Heda Wang, and Kan Li. 2024a. Turning dust into gold: Distilling complex reasoning capabilities from llms by leveraging negative data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18591–18599.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025b. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*.

Zhiming Li, Yushi Cao, Xiufeng Xu, Junzhe Jiang, Xu Liu, Yon Shin Teo, Shang-Wei Lin, and Yang Liu. 2024b. Llms for relational reasoning: How far are we? In *Proceedings of the 1st International Workshop on Large Language Models for Code*, pages 119–126.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2025a. Are your llms capable of stable reasoning? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17594–17632.

Zitao Liu, Ying Zheng, Zhibo Yin, Jiahao Chen, Tianqiao Liu, Mi Tian, and Weiqi Luo. 2025b. Arithmeticgpt: empowering small-size large language models with advanced arithmetic skills. *Machine Learning*, 114(1):24.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 2: short papers)*, pages 1773–1781.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, and 1 others. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Xuefei Ning, Zifu Wang, Shiyao Li, Zinan Lin, Peiran Yao, Tianyu Fu, Matthew Blaschko, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Can llms learn by teaching for better reasoning? a preliminary study. *Advances in Neural Information Processing Systems*, 37:71188–71239.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *arXiv preprint arXiv:2506.02867*.

Leonardo Ranaldi, Marco Valentino, and Andre Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17222–17240.

Jie Shao and Jianxin Wu. 2025. Who reasons in the large language models? *arXiv preprint arXiv:2505.20993*.

Aman Sharma and Paras Chopra. 2025. Think just enough: Sequence-level entropy as a confidence signal for llm reasoning. *arXiv preprint arXiv:2510.08146*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Yijun Tian, Yikun Han, Xiusi Chen, Wei Wang, and Nitesh V Chawla. 2025. Beyond answers: Transferring reasoning capabilities to smaller llms using multi-teacher knowledge distillation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 251–260.

Chengyu Wang, Taolin Zhang, Richang Hong, and Jun Huang. 2025. A short survey on small reasoning models: Training, inference, applications and research directions. *arXiv preprint arXiv:2504.09100*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025a. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

Shuyao Xu, Cheng Peng, Jiangxuan Long, Weidi Xu, Wei Chu, and Yuan Qi. 2025b. Harnessing negative signals: Reinforcement distillation from teacher data for llm reasoning. *arXiv preprint arXiv:2505.24850*.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective distillation of table-based reasoning ability from llms. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5538–5550.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. 2025. Rulearena: A benchmark for rule-guided reasoning with llms in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 550–572.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. 2024. Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2571–2597.

Appendix

A The pseudocode of the proposed method

The pseudocode of the proposed method is provided in Algorithm 1.

B Dataset Details

We provide detailed descriptions of the datasets used in our experiments, focusing on their task characteristics, reasoning requirements, and evaluation protocols.

GSM8K. GSM8K is a benchmark for grade-school-level mathematical reasoning, consisting of short word problems that require multi-step arithmetic and logical reasoning. Each problem is accompanied by a single ground-truth numerical

Algorithm 1 Module-Aware Reasoning Distillation with Difficulty-Adaptive Supervision

Require: Frozen teacher T, student S, controller $\mathcal{M}_\psi$, training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$
Ensure: Student parameters θ_S, controller parameters ψ

Stage I: Teacher Analysis and Layer Selection
Compute layer-wise reasoning saliency from teacher activations
Select top-k salient layers $\mathcal{L}_S$ for supervision
Insert lightweight adapters into selected student modules
Stage II: Difficulty-Adaptive Distillation
while not converged **do**
  for each training sample $x \in \mathcal{D}_{\text{train}}$ **do**
    Run teacher and collect hidden representations
    Predict module-wise supervision strengths via controller $\mathcal{M}_\psi$
    Compute student loss with token-level and selected module-level distillation
    Update student parameters θ_S
  end for
  Update controller parameters ψ using validation performance
end while
  return θ_S, ψ

answer. We report exact-match accuracy on the test set, following standard practice.

SVAMP. SVAMP is designed to evaluate robustness in mathematical problem solving by introducing semantic variations to existing math word problems. Compared to GSM8K, SVAMP places greater emphasis on understanding problem structure and avoiding spurious correlations. Performance is measured using exact-match accuracy.

LogiQA. LogiQA is a multiple-choice benchmark for logical reasoning, covering tasks such as deductive reasoning, causal inference, and logical consistency checking. Each instance consists of a context paragraph, a question, and multiple candidate answers. We evaluate models using classification accuracy.

StrategyQA. StrategyQA focuses on multi-hop commonsense reasoning, where answering a question requires combining several implicit facts rather than retrieving a single explicit statement. The task is formulated as binary classification, and accuracy is used as the evaluation metric.

All datasets follow their official train, validation, and test splits. No additional filtering or data augmentation is applied unless explicitly stated.

C Baseline Methods

This section provides additional details on the baseline methods used for comparison, highlighting their core modeling choices and reasoning supervision strategies.

T5-XXL+CoT. This baseline augments the T5-XXL model with chain-of-thought prompting, encouraging the model to generate intermediate reasoning steps before producing a final answer. The approach relies on explicit reasoning traces provided during training or prompting.

GPT-J+Self-Reflection. GPT-J+Self-Reflection improves reasoning by iteratively refining generated solutions. The model evaluates its own intermediate outputs and performs corrective reasoning steps, aiming to reduce logical and arithmetic errors.

ORCA2-7B. ORCA2-7B is trained using explanation-rich supervision, where teacher-generated rationales are used to guide the student toward better reasoning generalization across tasks.

CodeT5-Large+PaD. This method introduces program-aided decomposition, decomposing complex mathematical problems into executable intermediate programs that facilitate structured reasoning and solution verification.

Llama-7B+NCE and Llama2-7B+ReversalMath. These approaches focus on contrastive and curriculum-based objectives tailored for mathematical reasoning. They emphasize learning from hard negative examples or progressively increasing problem difficulty.

Qwen2.1.5B+SIKeD and Llama3.1-8B+ReDistill. These recent methods propose reasoning-aware distillation strategies that selectively transfer teacher knowledge to student models. Supervision is typically applied at the instance or sequence level, without explicitly distinguishing reasoning-critical layers or modules.

Model	Method	Placement	Rank	Params (M)	GSM8K	SVAMP
1.5B	Uniform LoRA	All 28 layers	2	0.34	55.84	63.27
	Uniform Adapter	All 28 layers	2	0.34	57.63	65.18
	MARD	Top-3 layers	16	0.30	**66.12**	**72.05**
7B	Uniform LoRA	All 28 layers	4	1.61	66.94	71.42
	Uniform Adapter	All 28 layers	4	1.61	68.57	73.16
	MARD	Top-3 layers	32	1.38	**71.50**	**76.33**

Table 2: Parameter-matched comparison across model scales. All methods use frozen backbones and identical training settings, differing only in adapter placement strategy.

All baseline results are obtained using publicly available implementations or reimplemented under identical training and evaluation settings to ensure fair comparison.

D Parameter-Matched Uniform Adapter Baselines

To further isolate the effect of adapter placement from parameter scale, we conduct additional comparisons with parameter-matched uniform adapter baselines. In particular, we compare MARD against uniform LoRA and uniform bottleneck adapter configurations where the backbone model remains fully frozen and adapters are inserted uniformly across all layers.

D.1 Parameter Matching Protocol

To ensure a fair comparison, all methods follow the same experimental setup:

- **Frozen backbone:** All backbone Transformer parameters remain fixed.
- **Insertion modules:** Adapters are inserted after the attention output projection (`o_proj`) and the FFN down projection (`down_proj`).
- **Adapters per layer:** Two adapters per layer.
- **Training objective:** Reasoning distillation with supervised cross-entropy on teacher-generated reasoning trajectories.
- **Optimization settings:** Identical across all methods.

The only varying factor is the **adapter placement strategy**. Uniform baselines distribute adapters across all Transformer layers, while MARD concentrates adapters in a small subset of reasoning-critical layers selected by teacher-derived saliency.

D.2 Qwen2.5-Math-1.5B

The 1.5B student model contains 28 Transformer layers with hidden size $d = 1536$. For a bottleneck adapter with rank r, the parameter count is $2dr$. Since MARD inserts two adapters per selected layer, the total parameter count for k layers is $4kdr$.

In our configuration, MARD selects the top-3 reasoning-salient layers ($k = 3$) with rank $r = 16$, resulting in:

$$4 \times 3 \times 1536 \times 16 = 294{,}912 \approx 0.30M$$

For the uniform baselines, adapters are inserted into all 28 layers, resulting in 56 adapters in total. The parameter count becomes:

$$112dr$$

Matching the MARD budget gives:

$$112 \times 1536 \times r_u \approx 294{,}912$$

which yields $r_u \approx 1.7$. Since the rank must be an integer, we use $r_u = 2$ for both LoRA and Adapter baselines.

D.3 Qwen2.5-Math-7B

The 7B student model has hidden size $d = 3584$ and 28 layers. MARD again selects three reasoning-critical layers with rank $r = 32$:

$$4 \times 3 \times 3584 \times 32 = 1{,}376{,}256 \approx 1.38M$$

For uniform insertion across all layers:

$$112 \times 3584 \times r_u \approx 1{,}376{,}256$$

which yields $r_u \approx 3.4$. We therefore use $r_u = 4$ for the uniform baselines.

As shown in Table 2, under matched parameter budgets, MARD consistently outperforms uniform adapter baselines across both model scales.

D.4 Discussion

Under strictly controlled parameter budgets and identical frozen-backbone training settings, uniform adapter placement forces the per-layer rank to be extremely small. This spreads limited adaptation capacity across early and non-critical layers, effectively diluting representational capacity for reasoning-related computation.

In contrast, MARD concentrates the same parameter budget on a small set of saliency-identified reasoning bottlenecks. Since backbone parameters, training objectives, optimization settings, and total trainable parameter counts are all controlled (Table 2), the observed improvements can be directly attributed to **module-aware selective placement**, rather than increased parameter scale.

E Training and Evaluation Details

This appendix provides a detailed description of the training and evaluation protocols used for our *Interpretable Module-Aware Reasoning Distillation with Difficulty-Adaptive Supervision* framework. The description complements the method section by clarifying hyperparameters, optimization procedures, and evaluation setups.

E.1 Student Training

Training Data and Split. All student models are trained exclusively on the **GSM8K training set**, which consists of grade-school–level mathematical word problems requiring multi-step reasoning. To support meta-controller optimization and mitigate overfitting, we randomly reserve **10% of the GSM8K training samples as a validation set**, while the remaining 90% are used for student parameter updates. This split is performed once prior to training and kept fixed across all runs to ensure reproducibility and fair comparison. No samples from the GSM8K test set or other evaluation benchmarks are used during training or validation.

Backbone and Adapter Initialization. The student backbone architectures (Qwen-2.5-Math-1.5B and Qwen-2.5-Math-7B) remain frozen, except for the injected adapters. Adapters are initialized with small random weights following a zero-centered normal distribution with a standard deviation of 0.02. Only layers selected based on teacher reasoning saliency ($\mathcal{L}_S$) are augmented with adapters.

Loss Functions. We jointly optimize token-level and module-level distillation losses, as described in Section 3. The overall training objective is defined as:

$$\mathcal{L}_{\text{student}} = \lambda_{\text{token}}\mathcal{L}_{\text{token}} + \sum_{s\in\mathcal{L}_S}\left(\lambda_{\text{attn}}^{(s)}\mathcal{L}_{\text{attn}}^{(s)} + \lambda_{\text{ffn}}^{(s)}\mathcal{L}_{\text{ffn}}^{(s)}\right), \tag{17}$$

where the weighting coefficients λ are dynamically reweighted by the meta-controller $\mathcal{M}_\psi$ according to the estimated input complexity.

Meta-Controller Optimization. The meta-controller $\mathcal{M}_\psi$ is trained jointly with the student using a bi-level optimization procedure:

- **Inner loop:** Student parameters θ_S are updated via gradient descent on training batches using controller-supplied supervision weights.
- **Outer loop:** Controller parameters ψ are updated by minimizing the validation loss $\mathcal{L}_{\text{val}}$ evaluated on the held-out 10% GSM8K validation set, enabling adaptive reweighting of module-level supervision.

We employ the AdamW optimizer for both the student adapters and the controller, with a learning rate of 1×10^{-4} for adapters and 5×10^{-5} for the controller, and a weight decay of 0.01.

Batching and Sequence Handling. Training is conducted with a batch size of 64 sequences per GPU. Input sequences are truncated or padded to a maximum length of 512 tokens. Gradients are accumulated over two steps, resulting in an effective batch size of 128.

Regularization and Stabilization. To prevent catastrophic forgetting and stabilize training for small student models, we apply the following techniques:

- Layer-wise gradient clipping with a maximum norm of 1.0.
- Dropout with a rate of 0.1 applied to adapter outputs.
- Temperature smoothing of the teacher distribution with $T = 2.0$ for KL-based token-level distillation.

Hyperparameter	Value (1.5B)	Value (7B)
Batch size	64	64
Effective batch size	128	128
Max sequence length	512	512
Adapter LR	$1e{-4}$	$1e{-4}$
Controller LR	$5e{-5}$	$5e{-5}$
Weight decay	0.01	0.01
Dropout	0.1	0.1
Gradient clipping	1.0	1.0
Temperature (KL)	2.0	2.0

Table 3: Key training hyperparameters for student and meta-controller.

E.2 Evaluation Protocols

Decoding. During evaluation, we use greedy decoding without self-consistency or ensemble-based inference. Chain-of-thought reasoning traces are available only to teacher models during training and are not provided to student models at test time.

Metrics. We report task-specific accuracy using standard evaluation protocols:

- **Arithmetic and Logical Reasoning Tasks (GSM8K, SVAMP):** Exact-match accuracy on final numeric answers.
- **Multiple-Choice Reasoning Tasks (LogiQA, StrategyQA):** Classification accuracy over the provided answer options.

Validation and Early Stopping. Models are evaluated on the held-out GSM8K validation set after each training epoch. Early stopping is triggered if validation accuracy does not improve for five consecutive epochs.

Hardware. All experiments are conducted on a single machine equipped with four NVIDIA RTX 4090 GPUs and one NVIDIA A800 80GB GPU. Mixed-precision (FP16) training and gradient accumulation are employed to accommodate larger effective batch sizes under GPU memory constraints.

Reproducibility. Random seeds are fixed across all experiments, and we report mean performance over three independent runs. Complete hyperparameter specifications, including optimizer settings, learning rate schedules, and controller architecture details, are provided in Table 3.

E.3 Efficiency Analysis

We analyze the computational efficiency of MARD in terms of parameter cost, FLOPs, and runtime overhead. Since the backbone remains frozen, the additional cost comes only from lightweight adapters inserted into a small subset of layers.

Model Setup. We consider two Qwen backbones: **Qwen2.5-Math-1.5B** (L=28, d=1536) and **Qwen2.5-Math-7B** (L=28, d=3584). Following MARD, adapters are inserted into the **top-3 saliency-selected layers**, with two adapters per layer (after `o_proj` and `down_proj`), resulting in 6 adapters in total.

Each adapter adopts a bottleneck design with rank r (16 for 1.5B, 32 for 7B). The parameter count of a single adapter is $2dr$, leading to a total of $12dr$ trainable parameters.

Parameter Cost. Substituting model configurations, we obtain:

- **1.5B:** $12 \times 1536 \times 16 = 294{,}912 \approx 0.30$M
- **7B:** $12 \times 3584 \times 32 = 1{,}376{,}256 \approx 1.38$M

This corresponds to only **0.02% of backbone parameters** at both scales, demonstrating that MARD introduces minimal overhead while enabling targeted reasoning-aware adaptation.

Incremental FLOPs. We next estimate the additional computation introduced by the adapters. For a sequence length of $T = 512$, the extra forward FLOPs are approximately:

- **1.5B model**: ~ 0.15B FLOPs
- **7B model**: ~ 0.70B FLOPs

For comparison, the forward computation of a 28-layer Transformer scales as $O(LTd^2)$ and typically lies in the tera-FLOP range per forward pass for models of this size. Therefore, the additional computation introduced by MARD accounts for well below 1% of the total FLOPs.

Moreover, because the backbone parameters are frozen, the backward pass only involves the small adapter modules, further reducing the effective training cost.

Measured Training and Inference Overhead. To complement the theoretical estimates above, we also measure the practical runtime overhead introduced by MARD. Experiments are conducted on **A100 80GB GPUs** using **bf16 precision**, with the same batch sizes and optimizer settings as in the main experiments.

Training overhead is reported as the relative throughput change compared to frozen-backbone baseline training. Inference latency is measured as the average per-token decoding time with KV cache enabled.

Notably, the meta-controller operates only on low-dimensional saliency signals during training and is removed at inference time, introducing no additional runtime cost.

Overall, these results demonstrate that MARD achieves reasoning-aware distillation with negligible computational overhead while remaining highly parameter-efficient.

F Heterogeneous Teacher–Student Architectures

The main experiments in this work adopt architecturally aligned teacher–student pairs (e.g., DeepSeek-Distill-Qwen → Qwen) to facilitate faithful layer-wise and module-wise alignment when analyzing where reasoning capabilities are localized. To further evaluate the robustness of MARD under architectural mismatch, we additionally study a heterogeneous teacher–student setting.

Experimental Setup. We construct a cross-architecture distillation scenario where the student backbone remains fixed while the teacher architecture changes.

- **Heterogeneous Teacher:** DeepSeek-R1-Distill-Llama-8B
- **Student:** Qwen2.5-Math-1.5B
- **Homogeneous Reference Teacher:** DeepSeek-R1-Distill-Qwen-14B
- **Datasets:** GSM8K and SVAMP
- **Metric:** Exact Match (EM)

In this setting, only the teacher architecture is replaced, while the student model, training procedure, and evaluation protocol remain identical.

Teacher → Student	GSM8K	SVAMP
Llama-8B → Qwen-1.5B	64.87	70.94
Qwen-14B → Qwen-1.5B	**66.12**	**72.05**

Table 4: Performance of MARD under heterogeneous and homogeneous teacher–student architectures.

Results. As shown in Table 4, the heterogeneous teacher yields slightly lower performance than the architecturally aligned teacher, with decreases of 1.25 EM on GSM8K and 1.11 EM on SVAMP. Nevertheless, the performance gap remains modest, indicating that MARD maintains stable cross-architecture transfer.

Discussion. Several factors may contribute to the small degradation observed in the heterogeneous setting. First, the saliency-mass preserving alignment used by MARD implicitly assumes a certain degree of structural similarity between teacher and student representations. When transferring from Llama to Qwen architectures, intermediate feature geometries may differ, which can slightly reduce alignment precision.

Second, subtle architectural differences, such as normalization placement or attention parameterization, may influence how module-level supervision is transferred across models.

Finally, the heterogeneous teacher used in this experiment (8B) is smaller than the homogeneous reference teacher (14B), which may provide slightly weaker reasoning supervision.

Despite these factors, the overall performance remains competitive, indicating that MARD is robust to moderate architectural mismatch. These results suggest that while architecturally aligned teacher–student pairs provide the strongest supervision, MARD can generalize effectively to heterogeneous distillation scenarios with only minor performance degradation.