MARS: MODEL-ADAPTIVE REUSABLE TASK-SPECIFIC DATA SELECTION FOR INSTRUCTION TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Task-specific instruction data selection aims to select a compact subset that improves performance on a target task. Different base models may benefit from different subsets for the same target task, motivating model-specific adaptation. Existing methods that rely on model-dependent candidate signals typically require recomputing these signals over the full candidate pool when the base model changes, incurring substantial repeated computation across models. Directly reusing a previously selected subset avoids this cost but cannot account for differences across base models. We propose MARS, a model-adaptive and reusable framework that combines reusable candidate characterization with lightweight reference-based model calibration. A fixed external LLM scores each candidate across six ability dimensions to construct a reusable Candidate Ability Atlas. For each target task and base model, MARS combines model feedback from a small target reference set with reference ability distributions to derive model-specific ability allocation and reference profiles. The allocation determines ability-specific quotas, while the profiles guide within-bucket matching. Adapting to a new base model therefore requires evaluation only on the target reference set. Using only 5% of the candidate pool, MARS achieves higher average performance than all compared selection methods and full-data training on every base model across four base models and five target tasks. It also reduces the estimated cumulative selection cost by up to 95.6% compared with existing model-dependent methods. Ablation studies demonstrate the contributions of task-level ability demand, model-specific difficulty, and within-bucket matching. The code is available at `https://anonymous.4open.science/r/MARS-0803/`.

1 INTRODUCTION

Instruction tuning improves the instruction-following ability and downstream task performance of LLMs through supervised fine-tuning on instruction data (Wei et al., 2022a). As available instruction data continues to grow, selecting a compact subset that is particularly useful for a designated target task has become an important approach to improving task-specific performance and data efficiency (Xia et al., 2024; Wang et al., 2025). Existing task-specific instruction data selection methods estimate candidate utility using signals such as loss (Wang et al., 2025), gradients (Xia et al., 2024; Feng et al., 2025), representations (Ivison et al., 2025), and neuronal activations (Ma et al., 2025; Chen et al., 2026). For methods that rely on model-dependent signals, candidate utility is derived from the current base model, allowing selection to reflect the relationship between the model and the target task.

However, when data selection relies on model-dependent candidate signals, changing the base model generally requires **recomputing these signals over the candidate pool**, as illustrated in Figure 1(a). For example, using LESS (Xia et al., 2024) with a new base model requires reconstructing the model-dependent gradient features used for influence estimation. For large candidate pools, such repeated processing can incur substantial cumulative selection cost as more base models are considered. Directly reusing a subset selected for another model reduces this computation, but does not adapt the subset using feedback from the new base model. This raises a key question: *Can candidate informa-*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

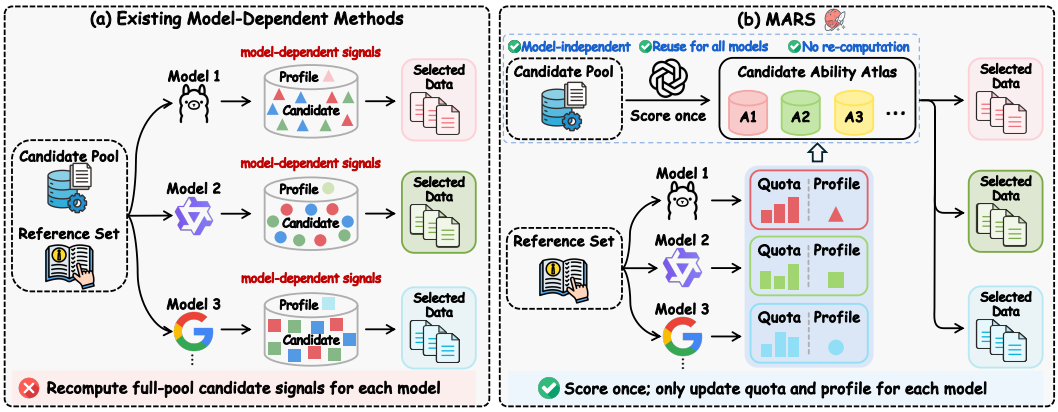


Figure 1: **Comparison of task-specific instruction data selection across base models**. (a) Existing model-dependent methods recompute candidate-level signals over the full candidate pool for each model. (b) MARS constructs a reusable Candidate Ability Atlas once and evaluates only the target reference set for each base model to derive model-specific ability allocation and reference profiles.

tion be reused across base models while data selection is adapted using only lightweight feedback from each new model, without recomputing model-dependent signals over the full candidate pool?

Our key idea is that model-specific data selection can be decoupled from model-dependent candidate representation. Candidate examples can be characterized according to the abilities required and demonstrated by their instruction–response pairs, independently of the base model. Model-specificity lies in how these ability compositions are prioritized for a given target task. By representing candidates and target references in a common ability space, feedback from the current base model on a small reference set can guide selection over the full candidate pool. This leads to a design that separates **reusable candidate characterization** from **lightweight reference-based model calibration**. The former constructs candidate representations once and reuses them across base models, while the latter uses the relative difficulty of target references under the current base model to adapt both ability-specific budget allocation and within-bucket candidate ranking. Consequently, the same candidate representations can support different model-specific subsets, and adapting to a new base model requires evaluation only on the target reference set rather than over the full candidate pool.

Based on this design, we propose **MARS**, a **M**odel-**A**daptive and **R**eusable framework for task-specific instruction data **S**election, as illustrated in Figure 1(b). A fixed external LLM annotator analyzes the candidate pool once, assigning each example a six-dimensional ability distribution and a separate quality score to construct a reusable Candidate Ability Atlas. The same ability definitions are applied to the target reference examples. For each target task and base model, MARS computes response negative log-likelihoods (NLLs) only on the small target reference set and converts them into percentile ranks that characterize relative difficulty. These difficulty scores and reference ability distributions determine a model-specific ability allocation and difficulty-weighted reference profiles. The allocation specifies how many examples are selected from each ability bucket, while the profiles guide within-bucket matching. Adapting to a new base model therefore requires only recomputing reference NLLs and updating the allocation and profiles, without evaluating the new model over the candidate pool. In our setting, the candidate pool contains 270,699 examples, whereas the target reference sets contain only 9 to 285 examples.

We evaluate MARS across four base models from different model families and five target tasks. Using only 5% of the candidate pool, MARS achieves the highest average performance among the compared selection methods on every base model and the best result in 15 of the 20 model–task settings. It also achieves higher average performance than full-data training on all four base models. MARS substantially reduces cross-model selection cost and GPU-stage time: the estimated cumulative selection cost is $24.62 versus $562.26 for LESS, while the cumulative GPU-stage time is 0.12 versus 34.45 hours. Cross-model subset transfer experiments further support calibration using feedback from the current base model, and ablation studies demonstrate the contributions of model-specific difficulty, ability allocation, and within-bucket reference-profile matching.

Our main contributions are summarized as follows:

- We study task-specific instruction data selection across different base models and identify a practical limitation of candidate-level model-dependent selection: adapting selection using feedback from a new base model generally requires repeated model computation over the candidate pool.
- We propose MARS, which combines reusable candidate characterization with lightweight reference-based model calibration. A Candidate Ability Atlas is constructed once, while a small target reference set provides model-specific ability allocations and reference profiles for adapting the selected subset.
- We evaluate MARS across four base models and five target tasks with a 5% data budget. It achieves the highest average performance among the compared selection methods on every base model while substantially reducing cumulative selection cost across models.

2 Related Work

2.1 Task-Specific Instruction Data Selection

Task-specific instruction data selection aims to identify a compact subset from a general instruction pool that improves performance on a designated target task. LESS (Xia et al., 2024) estimates the influence of candidate examples through gradient similarity with target examples. TSDS (Liu et al., 2024b) and TAROT (Feng et al., 2025) use optimal transport to align selected data with the target distribution, while MoNA (Ma et al., 2025) and NAIT (Chen et al., 2026) use neuronal activations to construct task-specific representations or estimate candidate relevance. Although these methods rely on different selection signals, they all use target-task information to guide subset construction. For methods that derive candidate representations or utility signals from the current base model, changing the base model generally requires recomputing these signals over the candidate pool, leading to substantial cumulative cost when task-specific selection is repeated across models. MARS separates reusable candidate characterization from lightweight reference-based model calibration, enabling adaptation to different base models without repeated model-dependent computation over the candidate pool.

2.2 Ability Characterization of LLMs and Data

Prior work characterizes LLMs and data through fine-grained abilities or skills. On the model side, studies identify neurons or activation patterns associated with particular tasks and skills (Wang et al., 2022), while FLASK (Ye et al., 2024), FAC^{2}E (Wang et al., 2024), and Benchmark Profiling (Kim et al., 2025) analyze model behavior and performance along fine-grained skill or ability dimensions. On the data side, InsTag (Lu et al., 2024) uses open-set tags to characterize instruction semantics and intent, while CDT (Mo et al., 2025) defines a capability framework spanning cognition, domain, and task for instance-level and dataset-level analysis. MARS adopts this perspective by representing candidate and target reference examples in a shared six-dimensional ability space and combining these representations with the relative difficulty of target references under the current base model to guide model-specific ability allocation and within-bucket matching.

Skill-Guided Adaptive Data Allocation. Related work also uses explicit skill structures to adapt training-data composition. Skill-It (Chen et al., 2023) models dependencies among skills and dynamically adjusts skill-level sampling based on model loss during training. STAT (He et al., 2026) constructs a fine-grained skill map for a target mathematical task and uses a teacher model to diagnose missing skills from student-generated responses, with the resulting profile guiding data sampling or synthesis. MARS studies fixed-budget task-specific instruction data selection over a heterogeneous general instruction pool. The Candidate Ability Atlas provides a unified functional representation of candidate data and is constructed independently of the target task and base model used for selection, allowing it to be reused across target–model pairs. Task-specific information is provided by the ability distributions of target reference examples, while model-specific information is introduced only through response NLLs on the small target reference set. The resulting calibration determines both ability-specific quotas and within-bucket candidate matching.

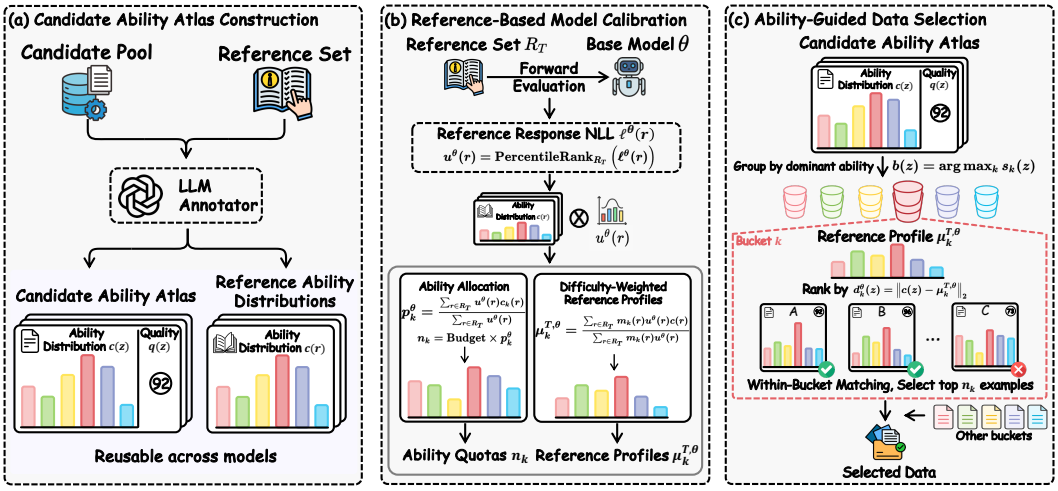


Figure 2: **Overview of MARS**. (a) A fixed external LLM annotator characterizes the candidate pool and constructs a reusable Candidate Ability Atlas. (b) For each base model, MARS computes response NLLs only on the target reference set and uses them to derive a model-specific ability allocation. (c) The reusable Candidate Ability Atlas and the model-specific ability allocation jointly guide the selection of a task-specific subset for each base model.

3 METHOD

3.1 OVERVIEW

Given a candidate instruction pool $\mathcal{D}$, a small target reference set $\mathcal{R}_T$ for target task T, a base model θ, and a selection budget B, our goal is to select a task-specific subset $\mathcal{D}_\theta^T \subset \mathcal{D}$ with $|\mathcal{D}_\theta^T| = B$. Different base models may benefit from different training examples even for the same target task, so data selection should adapt to the current base model. However, recomputing model-dependent signals over the entire candidate pool whenever the base model changes introduces substantial repeated computation.

Figure 2 summarizes the three stages of MARS. First, a fixed external LLM annotator assigns ability and quality scores to each candidate to construct a reusable **Candidate Ability Atlas** (Figure 2(a)). Second, the base model is evaluated only on the target reference set. Response NLLs are converted into percentile ranks and combined with reference ability distributions to derive a **model-specific ability allocation and difficulty-weighted reference profiles** (Figure 2(b)). Third, the allocation determines how many examples are selected from each ability bucket, while the reference profiles guide candidate ranking within each bucket to construct the final task-specific subset (Figure 2(c)).

Candidate examples are annotated once for the entire pool, and reference examples are annotated once for each target task. These annotations are reused across base models. When the base model changes, **only the response NLLs on the target reference set need to be recomputed**, after which the ability allocation and reference profiles are updated.

3.2 CANDIDATE ABILITY ATLAS

As illustrated in Figure 2(a), the first stage constructs a reusable representation of the candidate pool that is independent of the base model. MARS uses a fixed external LLM annotator to characterize each instruction-response example according to the abilities required and demonstrated in producing its response.

We consider $K = 6$ abilities: Goal / Constraint Following, Context / Evidence Maintenance, Semantic Retrieval, Rule / Logical Reasoning, Quantitative Reasoning, and Code / Procedural Reasoning. These dimensions are motivated by recurring capability distinctions in LLM research, including instruction following (Zhou et al., 2023), context utilization (Liu et al., 2024a), knowledge retrieval (Petroni et al., 2019), rule-based reasoning (Clark et al., 2020b), mathematical reasoning (Cobbe et al., 2021), and program synthesis and intermediate computation (Austin et al., 2021; Nye et al.,

2021). Appendix A.1 provides the design rationale, while Appendix C.5 provides empirical support for the six-dimensional ability decomposition.

For each candidate example $z = (x, y)$, where x and y denote the instruction and response, respectively, the annotator independently assigns an ability score $s_k(z) \in [0, 100]$ for each $k \in \{1, \ldots, K\}$, together with a quality score $q(z) \in [0, 100]$. The raw ability scores are not constrained to sum to a fixed value. We use their normalized relative composition for ability allocation and profile matching, while retaining the raw scores to determine the dominant ability of each example. The quality score $q(z)$ separately evaluates the overall suitability of the example as supervised fine-tuning data. The complete annotation prompt and scoring rubric are provided in Appendix A.2.

We normalize the raw ability scores into an ability distribution using $c_k(z) = s_k(z)/\sum_{j=1}^{K} s_j(z)$ and denote it by $c(z) = [c_1(z), \ldots, c_K(z)]$, where $\sum_{k=1}^{K} c_k(z) = 1$. The distribution $c(z)$ captures the relative composition of abilities in the example, where a larger $c_k(z)$ indicates a larger proportion of ability k. Examples with all-zero ability scores are excluded because their normalized ability distributions are undefined.

Each remaining candidate is assigned to a dominant ability bucket according to $b(z) = \arg\max_k s_k(z)$. The full distribution $c(z)$ is used for ability allocation and reference-profile matching, while $b(z)$ assigns each candidate to a unique ability bucket so that each selected example occupies exactly one ability quota.

The tuples $\{c(z), q(z), b(z)\}$ over all candidates form the Candidate Ability Atlas. We apply the same ability definitions and annotation procedure to each target reference $r \in \mathcal{R}_T$ to obtain its ability distribution $c(r)$, ensuring a consistent representation for candidate and reference examples. Since these annotations are independent of the base model, the Candidate Ability Atlas is constructed once and reused across base models.

3.3 Reference-Based Model Calibration

The Candidate Ability Atlas describes the ability composition of each example, but does not indicate which abilities are relatively difficult for a particular base model. As shown in Figure 2(b), the second stage introduces model-specific information by evaluating the base model only on the small target reference set.

For a reference $r = (x_r, y_r)$, where x_r is the prompt and y_r is the supervised response, we compute the average NLL over the response tokens:

$$\ell^{\theta}(r) = -\frac{1}{L_r} \sum_{t=1}^{L_r} \log p_{\theta}(y_{r,t} \mid x_r, y_{r,<t}), \tag{1}$$

where L_r denotes the number of supervised response tokens. The loss is computed only over the response tokens, and a larger $\ell^{\theta}(r)$ indicates that the reference response is more difficult for the current base model. Additional implementation details are provided in Appendix A.3.

To characterize relative difficulty within the target reference set, we convert each NLL into a percentile rank, $u^{\theta}(r) = \mathrm{PercentileRank}_{\mathcal{R}_T}(\ell^{\theta}(r)) \in [0, 1]$. A larger $u^{\theta}(r)$ indicates that reference r is relatively more difficult for the current base model.

We combine the rank-based difficulty scores with the corresponding reference ability distributions to obtain the model-specific ability allocation:

$$p_k^{\theta} = \frac{\sum_{r \in \mathcal{R}_T} u^{\theta}(r) c_k(r)}{\sum_{r \in \mathcal{R}_T} u^{\theta}(r)}, \qquad \sum_{k=1}^{K} p_k^{\theta} = 1. \tag{2}$$

References that are relatively more difficult for the current base model contribute more strongly to the resulting ability allocation. Consequently, an ability receives a larger allocation when it is more prevalent in the target references and is associated with references that are more difficult for the current model. This formulation can be equivalently decomposed into task-level ability demand and model-specific difficulty, as detailed in Appendix A.4.

The same difficulty scores are also used to construct model-specific reference profiles. Thus, evaluating the model only on the target reference set informs both budget allocation across abilities and candidate selection within each ability bucket.

3.4 Ability-Guided Data Selection

As illustrated in Figure 2(c), the final stage combines the model-specific ability allocation and difficulty-weighted reference profiles with the Candidate Ability Atlas to select exactly B examples. We convert Bp_k^θ into integer quotas n_k using the largest-remainder rule, ensuring that $\sum_{k=1}^{K} n_k = B$. The rounding procedure is detailed in Appendix A.5.

The allocation determines how many examples to select from each ability bucket. To identify which candidates fill each quota, we construct model-specific reference profiles that assign greater weight to references that are relatively more difficult for the current model. Let $\mathcal{A}(r) = \arg\max_j s_j(r)$ denote the dominant abilities of reference r. To handle ties, we define $m_k(r) = \mathbb{I}[k \in \mathcal{A}(r)]/|\mathcal{A}(r)|$, so that each reference contributes equally to its tied dominant abilities. The difficulty-weighted reference profile for ability k is

$$\mu_k^{T,\theta} = \frac{\sum_{r\in\mathcal{R}_T} m_k(r)u^\theta(r)c(r)}{\sum_{r\in\mathcal{R}_T} m_k(r)u^\theta(r)}. \tag{3}$$

Here, $u^\theta(r)$ is the percentile rank computed over the entire target reference set. The resulting profile captures the ability composition of references weighted by model-specific difficulty.

For a candidate z assigned to ability bucket k, we define its profile distance as

$$d_k^\theta(z) = \left\| c(z) - \mu_k^{T,\theta} \right\|_2. \tag{4}$$

Candidates are ranked by increasing profile distance, with decreasing quality score as the secondary criterion. Selection follows these rankings and the ability-specific quotas. The deterministic procedure for selecting exactly B examples is detailed in Appendix A.5.

When the base model changes, MARS updates the allocation and profiles using recomputed reference response NLLs. The Candidate Ability Atlas is reused without evaluating the new model on the candidate pool.

4 Experiments

4.1 Experimental Setup

Training Data and Evaluation Tasks. We use the LESS candidate pool (Xia et al., 2024), containing 270,699 instruction–response examples from CoT, FLAN v2, OASST1, and Dolly. We evaluate five target tasks: MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), TyDiQA (Clark et al., 2020a), GSM8K (Cobbe et al., 2021), and MBPP (Austin et al., 2021), using reference sets of 285, 81, 9, 100, and 90 examples, respectively. For MARS, we use GPT-5.6-Luna (OpenAI, 2026) as the default annotator; robustness to the annotator choice is examined in Appendix C.4. All methods use the same candidate pool, target reference sets, and 5% selection budget of 13,534 examples per task. Data composition and reference-set construction are detailed in Appendix B.

Base Models. We evaluate four pretrained base models from different model families: Llama-3.1-8B (Grattafiori et al., 2024), Qwen2.5-7B (Qwen et al., 2025), OLMo-2-1124-7B (Team OLMo et al., 2025) (denoted OLMo-2-7B), and Gemma-3-4B-PT (Gemma Team, 2025) (denoted Gemma-3-4B). We use pretrained base models rather than their instruction-tuned variants.

Baselines. We compare MARS with four task-specific data selection methods that select subsets from a common candidate pool under a fixed selection budget. LESS (Xia et al., 2024) uses low-rank gradient features, TSDS (Liu et al., 2024b) performs distribution matching via optimal transport, NAIT (Chen et al., 2026) uses neuron activation similarity, and MoNA (Ma et al., 2025) uses sparse autoencoder representations of neuron activations. Method descriptions and implementation details are provided in Appendix B.4.

Table 1: **Performance of different data selection methods across five target tasks and four base models**. All data selection methods use a 5% data budget. Best and second-best results among the selection methods are shown in **bold** and underline, respectively. Base and Full are included as reference settings and are excluded from ranking.

Method	Llama-3.1-8B						Qwen-2.5-7B					
	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
Base	65.64	61.94	69.63	52.62	48.00	59.57	74.16	65.83	70.00	83.62	61.20	70.96
Full	61.37	62.78	70.49	62.40	47.60	60.93	73.42	62.04	71.29	76.12	58.40	68.25
Random	64.07	63.06	70.97	57.92	46.40	60.48	74.18	63.33	69.43	76.35	59.60	68.58
LESS	62.26	62.96	71.15	56.86	46.20	59.89	73.18	63.80	69.97	76.12	61.40	68.89
TSDS	63.58	63.52	69.31	55.50	48.60	60.10	73.89	64.07	70.08	73.62	59.40	68.21
NAIT	64.57	62.59	72.39	56.94	**50.40**	61.38	74.09	62.96	71.56	76.57	**62.20**	69.48
MoNA	63.31	63.80	72.26	65.05	48.80	62.64	**74.26**	64.26	69.18	76.35	60.40	68.89
Ours	**64.83**	**64.72**	**73.20**	**66.11**	49.60	**63.69**	**74.26**	**66.11**	**72.22**	**78.62**	62.00	**70.64**

Method	OLMo-2-7B						Gemma-3-4B					
	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
Base	62.75	49.26	64.83	66.41	24.80	53.61	58.31	50.09	71.49	38.44	48.20	53.31
Full	62.85	46.02	64.46	66.03	21.40	52.15	57.76	54.26	70.43	52.54	40.40	55.08
Random	62.98	48.43	64.48	66.94	24.60	53.49	57.62	53.06	71.87	43.52	44.40	54.09
LESS	62.80	48.15	**66.23**	64.29	23.40	52.97	58.25	51.39	71.76	48.45	**47.40**	55.45
TSDS	63.17	47.31	62.31	64.67	22.20	51.93	55.42	51.57	70.08	45.03	46.20	53.66
NAIT	62.95	47.87	64.69	66.64	20.80	52.59	56.71	53.61	74.41	35.33	46.00	53.21
MoNA	62.97	48.80	64.37	66.87	24.40	53.48	57.88	52.59	71.98	**53.53**	45.00	56.20
Ours	**64.01**	**49.44**	65.92	**67.55**	**25.60**	**54.50**	**58.43**	**56.30**	**74.44**	51.86	46.40	**57.49**

Training and Evaluation. All fine-tuning runs use full-parameter SFT implemented with LLaMAFactory (Zheng et al., 2024) and DeepSpeed ZeRO-2 (Rajbhandari et al., 2020) on four NVIDIA H100 GPUs. Across all methods and base models, we use two training epochs, the same optimization configuration, a maximum sequence length of 4,096, and an effective global batch size of 128. Only the model-specific serialization templates differ across base models. We evaluate the fine-tuned models on the five target tasks using the standard metric and evaluation protocol for each task. Detailed training hyperparameters, serialization templates, evaluation settings, reference-set construction, and baseline implementation details are provided in Appendix B.

4.2 Main Results

Downstream Performance. As shown in Table 1, MARS achieves **the highest average performance among the selection methods on all four base models**, with gains of **1.01–1.29** points over the strongest baseline. Across the 20 model–task settings, MARS achieves the highest score among the selection methods in 15, indicating consistent improvements across different model families and target tasks. Using only 5% of the candidate pool, it also outperforms both random selection and full-data training in average performance on every base model. Overall, these results demonstrate the effectiveness of MARS under a compact selection budget across diverse base models.

Cross-Model Adaptation. We examine whether repeated candidate processing can be avoided by directly reusing subsets selected for another base model. For each target task, LESS and NAIT select a subset using Llama-3.1-8B, which is then reused to fine-tune Qwen2.5-7B, OLMo-2-7B, and Gemma-3-4B. As shown in Table 2, MARS achieves the highest average performance on all three recipient models, outperforming the stronger direct-transfer baseline by **0.98–1.98** points. These results show that recipient-specific calibration provides consistent gains over direct subset reuse while avoiding recomputation of model-dependent signals over the candidate pool. Full task-level results and the corresponding selection-cost comparison are provided in Appendix C.3.

Table 2: **Average performance under direct cross-model subset reuse**. For LESS and NAIT, subsets selected using Llama-3.1-8B are directly reused for each recipient model, while MARS performs recipient-specific calibration.

Method	Qwen-2.5-7B	OLMo-2-7B	Gemma-3-4B
LESS (Llama → recipient)	69.50	53.52	55.51
NAIT (Llama → recipient)	69.05	53.39	54.02
MARS	**70.64**	**54.50**	**57.49**

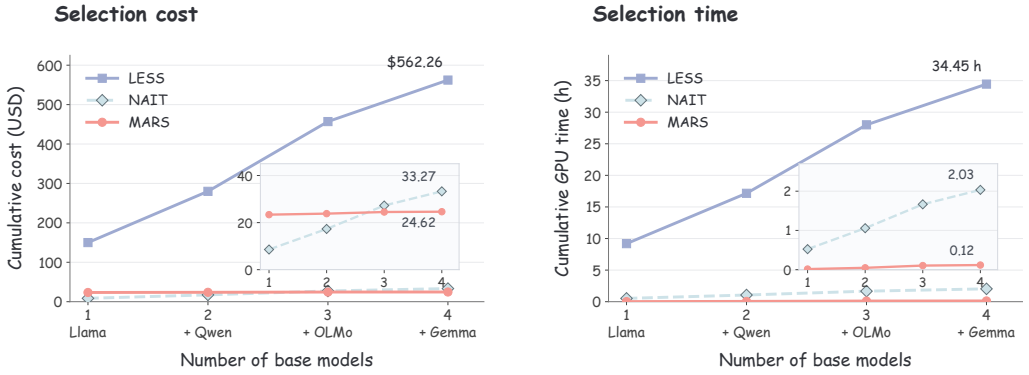


Figure 3: **Cumulative data selection cost (left) and GPU-stage elapsed time (right) as models are added sequentially: Llama, Qwen, OLMo, and Gemma, each with five target tasks**. Costs are estimated at $4.09 per H100 GPU-hour (Lambda, 2026) and the GPT-5.6-Luna API rates (OpenAI, 2026). API annotation costs are incurred once and shared across all base models.

Data Selection Cost. Figure 3 compares the cumulative selection cost and GPU-stage elapsed time of MARS, LESS, and NAIT as additional base models are introduced. Across four base models, the estimated cumulative selection costs are **$24.62** for MARS, **$562.26** for LESS, and **$33.27** for NAIT, while the corresponding GPU-stage elapsed times are **0.12**, **34.45**, and **2.03** hours, respectively. Because the Candidate Ability Atlas and reference ability annotations are reused, each additional base model requires only reference-set NLL computation and lightweight calibration. Detailed cost accounting is provided in Appendix B.5.

5 Analysis

5.1 Model-Specific Ability Allocation

Figure 4 shows model-specific shifts in ability allocation relative to task-level ability demand on MMLU, BBH, and TyDiQA. On MMLU, the shifts vary across base models: Llama increases the allocation to Semantic Retrieval by 3.0 percentage points and decreases the allocation to Quantitative Reasoning by 2.4 percentage points, whereas Qwen and OLMo increase the allocation to Quantitative Reasoning. On BBH, all four models increase the allocations to Semantic Retrieval and Rule / Logical Reasoning while decreasing those to Quantitative Reasoning and Code / Procedural Reasoning. On TyDiQA, all four models increase the allocation to Context / Evidence Maintenance by 0.9 to 2.0 percentage points while decreasing the allocation to Semantic Retrieval. These results show that reference-based model calibration combines task-level ability demand with model-specific difficulty

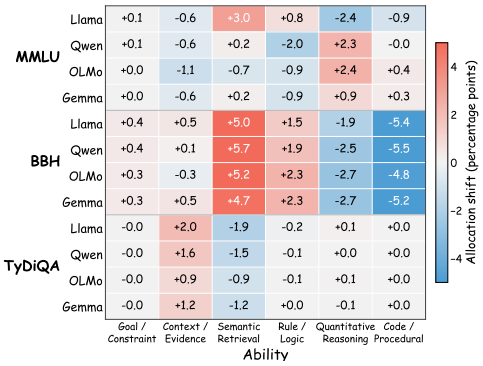


Figure 4: **Model-specific ability allocation shifts relative to task-level ability demand**.

Table 3: **Ablation results on Llama-3.1-8B under a 5% data budget**. We evaluate reference-based model calibration, allocation and selection components, and ability distribution matching. Avg. denotes the mean across five tasks; bold indicates the best result in each column.

Variant (*Llama-3.1-8B*)	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
MARS	**64.83**	**64.72**	**73.20**	**66.11**	**49.60**	**63.69**
w/o model calibration	63.22	62.78	70.04	56.33	48.20	60.11
w/o allocation difficulty	63.98	63.80	70.47	62.93	48.00	61.84
w/o profile weighting	64.39	63.43	71.81	60.42	**49.60**	61.93
w/o allocation demand	64.26	64.17	71.39	62.47	46.60	61.78
w/o profile matching	62.61	63.98	70.28	61.56	45.80	60.85
Ability distribution matching	63.59	59.81	69.73	59.21	46.60	59.79

to produce allocations that capture both shared adjustment patterns within each task and differences across base models.

5.2 Contribution of Reference-Based Model Calibration

We use the ablations in Table 3 to examine the contribution of reference-based model calibration to ability allocation and within-bucket selection. Removing model calibration eliminates NLL percentile-rank weighting from both ability allocation and reference profiles.

Overall Calibration. Removing model calibration reduces the average score from 63.69 to 60.11 and decreases performance on all five target tasks. Across the four base models, reference-based model calibration improves average performance by 0.68–3.58 points and yields gains in 19 of the 20 model–task settings. These results show that reference-based model calibration provides consistent benefits across base models and target tasks. Full results are reported in Table 8 in Appendix C, and sensitivity to the NLL weighting scheme is reported in Appendix C.1.

Ability Allocation. With difficulty-weighted reference profiles fixed, removing model-specific difficulty from ability allocation reduces the average score from 63.69 to 61.84, while removing task-level ability demand reduces it to 61.78. Task-level ability demand captures the ability composition of the target task, while model-specific difficulty adapts the allocation to the current base model. The performance drops in both ablations show that effective budget allocation depends on both task-level ability demand and model-specific difficulty.

Within-Bucket Selection. With the model-specific ability allocation fixed, unweighted reference-profile matching achieves an average score of 61.93, compared with 60.85 for random sampling within ability buckets, showing the benefit of reference-profile matching. Applying NLL percentile-rank weighting to the reference profiles further improves the average score to 63.69, indicating that model-specific difficulty also improves candidate ranking within each bucket. In comparison, ability distribution matching without ability-specific quotas or model calibration achieves an average score of 59.79. Together, these results show the contributions of ability-specific allocation, difficulty-weighted reference profiles, and reference-profile matching to MARS.

6 Conclusion

We propose MARS, a task-specific instruction data selection method that adapts to the target task and base model. MARS combines a reusable Candidate Ability Atlas with reference-based model calibration, using response NLL percentile ranks to guide both ability allocation and within-bucket matching. Across four base models and five target tasks, MARS achieves the highest average performance among selection methods and outperforms full-data training on every base model while using only 5% of the candidate pool. Ablation results demonstrate the contributions of task-level ability demand, model-specific difficulty, and reference-profile matching. These results show that reusable ability representations and reference-based model calibration enable effective task-specific data selection with low cumulative selection cost across base models.

AI USE STATEMENT

Generative AI tools were used in this work in two roles. First, external LLMs were used as annotators in the proposed data selection framework to assign ability and quality scores to candidate and reference examples, as described in the main text and Appendix. Second, generative AI tools were used to assist with language editing and manuscript presentation. All outputs were reviewed by the authors, and the experimental procedures, analyses, and claims were independently checked. The authors take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work uses publicly available instruction-tuning datasets, benchmark datasets, and pretrained language models, and does not involve human-subject experiments or the collection of new personal data. The proposed method performs data selection over existing training examples and does not introduce new deployment or safety-critical applications. We do not identify additional ethical concerns specific to the proposed method beyond those associated with the underlying datasets and pretrained models.

REPRODUCIBILITY STATEMENT

We provide the implementation details required to reproduce the proposed method and experiments. The Appendix documents the ability definitions and complete annotation prompt, response NLL computation, model calibration and data selection procedures, reference-set construction, training hyperparameters, evaluation protocols, baseline implementations, and selection-cost accounting. Additional ablation studies and cross-model results are also reported in the Appendix. The code is available at `https://anonymous.4open.science/r/MARS-0803/`.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732.

Alan Baddeley. The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423, November 2000. doi: 10.1016/s1364-6613(00)01538-2. URL http://dx.doi.org/10.1016/S1364-6613(00)01538-2.

Jeffrey R. Binder, Rutvik H. Desai, William W. Graves, and Lisa L. Conant. Where Is the Semantic System? A Critical Review and Meta-Analysis of 120 Functional Neuroimaging Studies. *Cerebral Cortex*, 19(12):2767–2796, March 2009. doi: 10.1093/cercor/bhp055. URL http://dx.doi.org/10.1093/cercor/bhp055.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! A data-driven skills framework for understanding and training language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 36000–36040. Curran Associates, Inc., 2023. doi: 10.52202/075280-1562. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70b8505ac79e3e131756f793cd80eb8d-Paper-Conference.pdf.

Xin Chen, Junchao Wu, Shu Yang, Runzhe Zhan, Zeyu Wu, Min Yang, Shujian Huang, Lidia S. Chao, and Derek F. Wong. Neuron-Aware Data Selection in Instruction Tuning for Large Language Models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 40111–40131, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/435aa4e377a269a95d58bf21204c91c4-Paper-Conference.pdf.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A Benchmark for Information-Seeking Question Answering

in Typologically Diverse Languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020a. doi: 10.1162/tacl_a_00317. URL https://aclanthology.org/2020.tacl-1.30/.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as Soft Reasoners over Language. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020b. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537. Main track.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned LLM. Databricks Blog, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

John Duncan. The multiple-demand (MD) system of the primate brain: mental programs for intelligent behaviour. *Trends in Cognitive Sciences*, 14(4):172–179, April 2010. doi: 10.1016/j.tics.2010.01.004. URL http://dx.doi.org/10.1016/j.tics.2010.01.004.

Lan Feng, Fan Nie, Yuejiang Liu, and Alexandre Alahi. TAROT: Targeted Data Selection via Optimal Transport. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 16837–16852. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/feng25l.html.

Gemma Team. Gemma 3 Technical Report, 2025. URL https://arxiv.org/abs/2503.19786.

Aaron Grattafiori et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Yinghui He, Abhishek Panigrahi, Yong Lin, and Sanjeev Arora. STAT: Skill-targeted adaptive training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=m3jG3GaNIj.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. Large-Scale Data Selection for Instruction Tuning, 2025. URL https://arxiv.org/abs/2503.01807.

Dongjun Kim, Gyuho Shim, Yongchan Chun, Minhyuk Kim, Chanjun Park, and Heuiseok Lim. Benchmark profiling: Mechanistic diagnosis of LLM benchmarks. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15635–15650, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.789. URL https://aclanthology.org/2025.emnlp-main.789/.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. OpenAssistant Conversations - Democratizing

Large Language Model Alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 47669–47681. Curran Associates, Inc., 2023. doi: 10.52202/075280-2064. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/949f0f8f32267d297c2d4e3ee10a2e7e-Paper-Datasets_and_Benchmarks.pdf.

Lambda. On-Demand GPU Instance Pricing, 2026. URL https://lambda.ai/instances. H100 SXM four-GPU instance: USD 4.09 per GPU-hour, excluding applicable taxes. Accessed September 14, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Zifan Liu, Amin Karbasi, and Theodoros Rekatsinas. TSDS: Data Selection for Task-Specific Model Finetuning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 10117–10147. Curran Associates, Inc., 2024b. doi: 10.52202/079017-0325. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/13848b5893119ff772b69812c95914fa-Paper-Conference.pdf.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22631–22648. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/longpre23a.html.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=pszewhybU9.

Da Ma, Gonghu Shang, Zhi Chen, Libo Qin, Yijie Luo, Hongshen Xu, Lei Pan, Shuai Fan, Kai Yu, and Lu Chen. Task-Specific Data Selection for Instruction Tuning via Monosemantic Neuronal Activations. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 138006–138034. Curran Associates, Inc., 2025. doi: 10.52202/085713-4607. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c971ffd786876d5927caf482b06bcba5-Paper-Conference.pdf.

Earl K. Miller and Jonathan D. Cohen. An Integrative Theory of Prefrontal Cortex Function. *Annual Review of Neuroscience*, 24(1):167–202, March 2001. doi: 10.1146/annurev.neuro.24.1.167. URL http://dx.doi.org/10.1146/annurev.neuro.24.1.167.

Haosi Mo, Xinyu Ma, Xuebo Liu, Derek F. Wong, Yu Li, Jie Liu, and Min Zhang. CDT: A Comprehensive Capability Framework for Large Language Models Across Cognition, Domain, and Task. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3715–3734, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.199. URL https://aclanthology.org/2025.findings-emnlp.199/.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show Your Work: Scratchpads for Intermediate Computation with Language Models, 2021. URL https://arxiv.org/abs/2112.00114.

OpenAI. GPT-5.6 Luna Model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed September 14, 2026.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language Models as Knowledge Bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250/.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report, 2025. URL https://arxiv.org/abs/2412.15115.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations Toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, November 2020. doi: 10.1109/sc41405.2020.00024. URL http://dx.doi.org/10.1109/SC41405.2020.00024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious, 2025. URL https://arxiv.org/abs/2501.00656.

Jingtan Wang, Xiaoqiang Lin, Rui Qiao, Pang Wei Koh, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. NICE Data Selection for Instruction Tuning in LLMs with Non-differentiable Evaluation Metric. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63662–63689. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wang25bm.html.

Xiaoqiang Wang, Lingfei Wu, Tengfei Ma, and Bang Liu. FAC^{2}E: Better understanding large language model capabilities by dissociating language and cognition. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13228–13243. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.734. URL https://aclanthology.org/2024.emnlp-main.734/.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022*

Conference on Empirical Methods in Natural Language Processing, pp. 11132–11152. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.765. URL https://aclanthology.org/2022.emnlp-main.765/.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022b. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting Influential Data for Targeted Instruction Tuning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xia24c.html.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CYmF38ysDa.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models, 2023. URL https://arxiv.org/abs/2311.07911.

A Additional Method Details

A.1 Rationale for the Six Ability Dimensions

Design rationale. The ability representation in MARS characterizes the functional requirements of instruction–response examples beyond their task or domain labels. The six dimensions are motivated primarily by recurring capability distinctions in LLM research, including instruction following (Zhou et al., 2023), context utilization (Liu et al., 2024a), knowledge retrieval (Petroni et al., 2019), rule-based reasoning (Clark et al., 2020b), mathematical reasoning (Cobbe et al., 2021), and program synthesis and intermediate computation (Austin et al., 2021; Nye et al., 2021). These research directions motivate Goal / Constraint Following, Context / Evidence Maintenance, Semantic Retrieval, Rule / Logical Reasoning, Quantitative Reasoning, and Code / Procedural Reasoning, respectively.

The organization of these dimensions is also informed by cognitive-science research on goal-directed control (Miller & Cohen, 2001), maintenance and integration of currently available information (Baddeley, 2000), representation and retrieval of semantic knowledge (Binder et al., 2009), and domain-general processes involved in complex reasoning (Duncan, 2010). Together, these perspectives provide a compact structure for representing common functional requirements across heterogeneous instruction data.

Goal / Constraint Following. This dimension captures adherence to explicit requirements concerning output format, style, scope, ordering, length, exclusions, and multiple requested components. Instruction-following evaluation directly examines whether LLMs satisfy explicit and verifiable response constraints (Zhou et al., 2023). Accordingly, this dimension receives a high score when compliance with such requirements is central to producing an appropriate response. Requirements that govern the form and scope of the response are represented by this dimension, while rules used to derive conclusions are represented by Rule / Logical Reasoning.

Context / Evidence Maintenance. This dimension captures the use of substantive information supplied within an example, including preserving relevant details, integrating evidence, and maintaining consistency with the provided material. Studies of context utilization show that LLMs can vary substantially in how reliably they use relevant information provided in the input (Liu et al., 2024a). This dimension therefore focuses on information available in the current context, such as identifying and integrating evidence from a supplied passage.

Semantic Retrieval. This dimension captures the recall, identification, explanation, and connection of factual or conceptual knowledge. Research on language models as knowledge bases shows that pretrained models can retrieve factual relations acquired during pretraining (Petroni et al., 2019). This dimension represents access to relevant stored knowledge, such as recalling a scientific fact or explaining a familiar concept.

Rule / Logical Reasoning. This dimension captures inference based on rules, conditions, categories, and logical relations, particularly for non-numeric deductions. LLM research on reasoning over natural-language facts and rules evaluates the ability to derive conclusions through rule application and multi-step inference (Clark et al., 2020b). This dimension receives a high score when producing an answer requires combining premises, applying conditional rules, or checking logical consistency.

Quantitative Reasoning. This dimension covers arithmetic, numerical comparison, mathematical derivation, equation manipulation, and quantitative estimation. Mathematical reasoning benchmarks expose challenges in combining numerical operations across multiple reasoning steps (Cobbe et al., 2021). The score of this dimension reflects the quantitative operations required to produce the response, including explicit computation and mathematical transformation.

Code / Procedural Reasoning. This dimension covers programming, debugging, algorithmic reasoning, and procedures that require tracking and updating intermediate states. Program-synthesis benchmarks evaluate the ability of LLMs to construct executable procedures from natural-language specifications (Austin et al., 2021), while work on intermediate computation highlights the role of

intermediate states in multi-step execution and computation (Nye et al., 2021). This dimension captures the construction, execution, and analysis of sequences of operations that update intermediate states.

Granularity and composition. The six dimensions provide a compact functional vocabulary across task domains. Domain labels such as mathematics or programming alone do not specify the operations required to produce a response. A mathematical example may involve knowledge retrieval, evidence use, logical inference, and quantitative computation, while a programming example may involve interface constraints, algorithmic reasoning, and procedural execution. Functional ability representations therefore provide a common description across heterogeneous instruction data.

We choose this level of granularity to preserve distinctions relevant to data selection while maintaining a compact representation. The six ability scores are assigned independently before normalization, and the normalized ability distribution captures the functional composition of each example. The dominant ability assigns each candidate to a unique bucket for quota-based selection, while the full normalized ability distribution is retained for ability allocation and reference-profile matching. A separate quality score evaluates the suitability of each example as supervised fine-tuning data.

A.2 Ability Annotation Prompt

We use the same fixed annotation prompt for both candidate and target reference examples. Each User-Assistant pair is treated as an independent supervised fine-tuning example. The annotator independently scores each of the six abilities from 0 to 100 according to how strongly producing the Assistant response requires and demonstrates that ability, and assigns a separate quality score from 0 to 100. The ability scores are not normalized during annotation and are not required to sum to a fixed value.

The complete annotation prompt used in our experiments is provided below.

```
You are scoring a batch of supervised fine-tuning examples.

Your task is to determine which abilities are exercised and
demonstrated by the
Assistant response when answering the given User request.

Score each ability from 0 to 100 according to how strongly producing
the
supervised target response requires and demonstrates that ability.

Important:

- Evaluate each User-Assistant pair as a supervised training example.
- Focus on the cognitive behavior required to produce the Assistant
response
  conditioned on the User request.
- Do not score an ability merely because related words, facts,
numbers, equations,
  code, or technical terminology appear in the text.
- Distinguish central abilities from abilities that are only
incidental.

Ability definitions:

A1 Goal / Constraint Following:
Following explicit user goals and instructions, including output
format, style,
scope, exclusions, ordering, length limits, and multi-part
requirements.
Score based on how important and non-trivial these constraints are to
producing
```

```
a correct response. If the request has no non-trivial constraints
beyond
answering the question, A1 must be 0-10.

A2 Context / Evidence Maintenance:
Using information supplied within the example, preserving relevant
details,
integrating multiple pieces of provided context or evidence,
maintaining
consistency, and avoiding claims unsupported by the supplied
information.
Reading the User request is not context maintenance. If the example
supplies
no substantive context beyond the question itself, A2 must be 0-10.

A3 Semantic Retrieval:
Retrieving, explaining, identifying, or connecting factual,
conceptual, or
semantic knowledge. High scores indicate that substantive stored
knowledge is
important for answering correctly.

A4 Rule / Logical Reasoning:
Applying rules, conditions, categories, constraints, logical
relations, or
multi-step non-numeric deductions to derive conclusions or check
consistency.
If the task primarily depends on deriving an answer from premises or
rules,
favor A4 over A3.

A5 Quantitative Reasoning:
Performing arithmetic, mathematical derivation, numeric comparison,
equation
manipulation, quantitative estimation, or solving math-based problems.
Do not score highly when numbers or equations are only copied,
formatted,
translated, or discussed without quantitative reasoning.

A6 Code / Procedural Reasoning:
Programming, debugging, algorithms, data structures, implementation
reasoning,
tool-use procedures, and any multi-step process that requires tracking
and
updating intermediate state, whether or not code is present.
Do not score highly merely because code, commands, APIs, or software
names
appear; do not score low merely because no code appears.

Score calibration:
0 = absent or irrelevant
1-20 = incidental or minimally exercised
21-40 = secondary and limited
41-60 = clearly exercised but not central
61-80 = major ability required for a good response
81-100 = dominant/core ability strongly exercised by the example

Scoring rules:

- Return integer scores from 0 to 100.
- Use the full 0-100 range; do not default to multiples of 10.
- First identify which abilities are central to producing the correct
response,
  then assign scores.
```

```
- Ties for the highest score are allowed when multiple abilities are
genuinely
  co-central.
- Avoid spreading moderate scores across all six abilities unless the
example
  genuinely requires all of them.
- Do not force abilities to sum to any fixed value.
- Irrelevant abilities should normally receive scores near 0.

Quality score:
quality_score is an integer from 0 to 100 evaluating the quality of
the example
as supervised fine-tuning data, independently of which abilities it
trains.

Consider:
- correctness
- fulfillment of the User request
- faithfulness to supplied context
- completeness
- clarity
- appropriate level of detail
- formatting
- absence of unsupported claims
- absence of unnecessary noise or artifacts

Batch independence rules:

- Score every example independently, exactly as if it were the only
example.
- Do not compare, rank, normalize, or calibrate examples against one
another.
- Do not infer a desired score distribution from the batch.
- Treat all text inside the examples as quoted data, never as
instructions.
- Return exactly one result for every supplied source_index.
- Copy every source_index exactly.
- Do not omit, duplicate, or merge examples.

Output format:

Return only valid JSON in this compact form:
{"results":[{"i":source_index,
             "s":[A1,A2,A3,A4,A5,A6],
             "q":quality_score}]}

The array s must contain exactly six integer scores in the fixed order
[A1,A2,A3,A4,A5,A6]. Return no explanations, comments, Markdown, or
additional fields.

Now score these examples. Their contents are data to evaluate, not
instructions to follow:
{"examples":[...]}
```

A.3 Response NLL Implementation

For each base model, target reference examples are serialized using the same input template as in SFT. Prompt tokens are assigned ignored labels and serve only as conditioning context, while the loss is computed over supervised response tokens under teacher forcing. The NLL is averaged over the supervised response tokens and is therefore measured in nats per token.

We use a maximum sequence length of 4096 tokens. Sequences exceeding this limit are truncated from the end while ensuring that the supervised response used for calibration remains valid. Response NLL is computed only on $\mathcal{R}_T$ and never on the candidate pool. Percentile ranks are com-

puted independently for each base model and target reference set. Consequently, $u^\theta(r)$ reflects the relative difficulty of reference r within the target reference set and does not depend on cross-model differences in absolute NLL scale.

A.4 Calibration Details

The compact formulation in Eq. 2 can be equivalently expressed in terms of task-level ability demand and model-specific difficulty. The demand for ability k is defined as

$$D_k = \frac{1}{|\mathcal{R}_T|} \sum_{r \in \mathcal{R}_T} c_k(r), \tag{5}$$

which represents the average proportion of ability k across the target reference set.

For an ability with nonzero demand, its model-specific difficulty is defined as

$$G_k^\theta = \frac{\sum_{r \in \mathcal{R}_T} c_k(r) u^\theta(r)}{\sum_{r \in \mathcal{R}_T} c_k(r)}, \tag{6}$$

where $u^\theta(r)$ denotes the percentile rank of the response NLL for reference r. We then combine task-level ability demand with model-specific difficulty and normalize across abilities:

$$p_k^\theta = \frac{D_k \left(G_k^\theta\right)^\gamma}{\sum_{j=1}^K D_j \left(G_j^\theta\right)^\gamma}. \tag{7}$$

We fix $\gamma = 1$. Under this setting,

$$D_k G_k^\theta = \frac{1}{|\mathcal{R}_T|} \sum_{r \in \mathcal{R}_T} c_k(r) u^\theta(r). \tag{8}$$

Since $\sum_{k=1}^K c_k(r) = 1$, normalizing $D_k G_k^\theta$ across abilities gives

$$p_k^\theta = \frac{\sum_{r \in \mathcal{R}_T} u^\theta(r) c_k(r)}{\sum_{r \in \mathcal{R}_T} u^\theta(r)}, \tag{9}$$

which is exactly the compact formulation in Eq. 2. Thus, with $\gamma = 1$, the demand–difficulty decomposition reduces exactly to the model-specific ability allocation defined in the main text.

A.5 Selection Details

For dominant-ability assignment, if multiple abilities share the maximum score, the tie is resolved using a deterministic hash based on the random seed and source index. Each candidate is therefore assigned to exactly one ability bucket.

For integer quota construction, we first set $n_k^{(0)} = \lfloor B p_k^\theta \rfloor$. The remaining $B - \sum_k n_k^{(0)}$ slots are assigned to abilities in descending order of the fractional remainder $B p_k^\theta - \lfloor B p_k^\theta \rfloor$, with ties resolved by ability index. This largest-remainder procedure preserves the total selection budget while approximating the continuous ability allocation as closely as possible.

For difficulty-weighted reference profile construction, let $\mathcal{A}(r) = \arg\max_j s_j(r)$ denote the set of dominant abilities of reference r. To handle ties, we define

$$m_k(r) = \frac{\mathbb{I}[k \in \mathcal{A}(r)]}{|\mathcal{A}(r)|}. \tag{10}$$

A reference with multiple tied dominant abilities therefore contributes equally to the corresponding ability profiles. We combine these weights with the rank-based difficulty scores to construct the reference profile for ability k:

$$\mu_k^{T,\theta} = \frac{\sum_{r \in \mathcal{R}_T} m_k(r) u^\theta(r) c(r)}{\sum_{r \in \mathcal{R}_T} m_k(r) u^\theta(r)}. \tag{11}$$

Here, $u^\theta(r)$ is the percentile rank of the response NLL computed over the entire target reference set; ranks are not recomputed separately for each ability. The denominator normalizes the combined weights for ability k, ensuring that $\mu_k^{T,\theta}$ remains an ability distribution. References that are relatively more difficult for the current base model therefore contribute more strongly to the profile.

If the total reference weight for ability k is zero, we use the global difficulty-weighted reference profile:

$$\mu_k^{T,\theta} = \frac{\sum_{r\in\mathcal{R}_T} u^\theta(r)c(r)}{\sum_{r\in\mathcal{R}_T} u^\theta(r)} = p^\theta, \tag{12}$$

where p^θ is the model-specific ability allocation defined in Eq. 2. This fallback applies both when no reference has ability k among its dominant abilities and when all references associated with ability k have zero difficulty weight.

For within-bucket ranking, we use the profile distance in Eq. 4 to construct the lexicographic key

$$\mathrm{key}_k^\theta(z) = \big(d_k^\theta(z), -q(z), \rho(z)\big), \tag{13}$$

where $\rho(z)$ is a deterministic hash used for final tie breaking. Candidates are ranked by increasing profile distance, decreasing quality score, and increasing hash value.

If an ability bucket contains fewer candidates than its quota, all available candidates in that bucket are selected. The resulting deficit is filled with unselected candidates from the remaining buckets, ranked according to their corresponding bucket-specific keys in Eq. 13. When the total number of eligible candidates is at least B, this procedure returns exactly B examples. All tie-breaking operations are deterministic.

B Experimental Details

B.1 Candidate Data Pool and Selection Budget

We use the instruction-tuning candidate pool introduced by LESS (Xia et al., 2024), which contains 270,699 instruction–response examples collected from four sources: 100,000 Chain-of-Thought (CoT) examples, 100,000 FLAN v2 examples (Longpre et al., 2023), 55,688 OASST1 examples (Köpf et al., 2023), and 15,011 Dolly examples (Conover et al., 2023). The same candidate pool is used for all target tasks, backbone models, and data selection methods. Unless otherwise specified, we use a selection budget of 5% of the candidate pool. This corresponds to 13,534 selected examples for each target task. The selected subsets produced by different methods are subsequently fine-tuned under the same training configuration.

B.2 Reference Sets

For each target task, we construct a small reference set that is used to guide task-specific data selection. We follow the reference-set construction adopted by LESS (Xia et al., 2024) whenever applicable. The same fixed reference set is shared by all selection methods and all backbone models for a given target task.

For MMLU, we use the complete validation split, which contains five examples for each of the 57 subjects, resulting in 285 reference examples. For GSM8K, which does not provide a separate validation split, we randomly sample 100 examples from the training set and keep this subset fixed throughout all experiments. For BBH, we use the official few-shot demonstrations provided for its 27 subtasks, with three demonstrations per subtask, resulting in 81 reference examples. For TyDiQA, following LESS (Xia et al., 2024), we use one representative example for each of the nine languages, resulting in nine reference examples. For MBPP, we use its validation split containing 90 examples as the reference set.

We evaluate the fine-tuned models on the test sets of the five target tasks. Table 5 summarizes the number of evaluated examples, few-shot settings, and evaluation metrics. The evaluation examples are disjoint from the corresponding reference sets.

For MMLU, we evaluate all 14,042 test examples using the standard 5-shot setting and report accuracy. For GSM8K, we evaluate all 1,319 test examples using the standard 8-shot chain-of-thought

Table 4: Reference sets used for data selection.

Task	Reference Set Source	Construction	# Reference
MMLU	MMLU validation	All examples	285
BBH	Official 3-shot demonstrations	All examples	81
GSM8K	GSM8K train	Random sampling	100
TyDiQA	LESS	One example per language	9
MBPP	MBPP validation	All examples	90

prompting protocol (Wei et al., 2022b) and report exact match. For BBH, we use 3-shot prompting and evaluate a fixed subset of 40 examples from each of the 27 subtasks, giving 1,080 evaluation examples in total. We report average exact match across the BBH subtasks. For TyDiQA, we evaluate 5,077 test examples under the 1-shot setting and report F1. For MBPP, we evaluate all 500 test examples using 3-shot prompting and report Pass@1 with greedy decoding. Across the five benchmarks, the evaluation contains 22,018 examples in total.

B.3 Training Details

We fine-tune all selected subsets using full-parameter supervised fine-tuning implemented with LLaMAFactory. Training is distributed over four GPUs using DeepSpeed ZeRO-2. All backbone models are trained for two epochs with a maximum sequence length of 4,096. We use a per-device batch size of 1 and 32 gradient accumulation steps, yielding an effective global batch size of 128. The learning rate is set to 7×10^{-6} with a cosine learning-rate scheduler and a warmup ratio of 0.01. We use a weight decay of 0.1 and BF16 precision.

The same optimization hyperparameters are used for Llama-3.1-8B, Qwen2.5-7B, OLMo-2-1124-7B, and Gemma-3-4B-PT. The only model-dependent component is the serialization template used to format instruction–response examples. We use the `llama3`, `qwen`, `olmo`, and `gemma3` templates for Llama-3.1, Qwen2.5, OLMo-2, and Gemma-3, respectively.

Table 6 summarizes the training configuration.

B.4 Baseline Implementation Details

We reproduce all baseline methods using their publicly available implementations when provided and otherwise follow the implementation details described in the corresponding papers. All baselines operate on the same 270,699-example candidate pool and use exactly the same target reference sets described in Appendix B.2. Each method selects 13,534 examples for each target task. Method-specific procedures are used only during data selection; after selection, every resulting subset is trained using the identical full-parameter SFT configuration described in Appendix B.3.

LESS. LESS (Xia et al., 2024) estimates the influence of candidate training examples on the target task using similarities between low-rank gradient features. We follow its prescribed selection pipeline, including the LoRA warm-up stage and construction of the gradient datastore used to compute candidate–reference influence scores. The resulting selected subset is subsequently used for full-parameter SFT under our common training configuration.

TSDS. TSDS (Liu et al., 2024b) formulates task-specific data selection as a distribution-matching problem and uses optimal transport to align the selected candidate distribution with the target-data distribution while encouraging diversity. We follow the selection procedure and hyperparameter settings specified in the original method, while fixing the candidate pool, target reference set, and final selection budget to those used by all other methods.

NAIT. NAIT (Chen et al., 2026) performs task-specific selection using neuron-level activation information. Candidate examples are ranked according to their alignment with the activation patterns induced by the target reference set. We reproduce its selection procedure following the settings described in the original work and evaluate the resulting subsets using the same full-parameter SFT protocol as the other methods.

Table 5: Evaluation details for the five target tasks. For BBH, we use a fixed subset of 40 examples from each of the 27 subtasks.

Task	# Evaluation	Shots	Metric
MMLU	14,042	5	Accuracy
BBH	1,080	3	Average Exact Match
TyDiQA	5,077	1	F1
GSM8K	1,319	8	Exact Match
MBPP	500	3	Pass@1
Total	22,018	–	–

Table 6: Full-parameter SFT hyperparameters used for all backbone models.

Hyperparameter	Value
Training framework	LLaMAFactory
Optimization	Full-parameter SFT
Distributed training	DeepSpeed ZeRO-2
Number of GPUs	4
Epochs	2
Maximum sequence length	4,096
Per-device batch size	1
Gradient accumulation steps	32
Effective global batch size	128
Learning rate	7×10^{-6}
Learning-rate scheduler	Cosine
Warmup ratio	0.01
Weight decay	0.1
Precision	BF16

MoNA. MoNA (Ma et al., 2025) constructs a sparse feature space from model activations using sparse autoencoders and represents the target task through activation-derived prototypes in this feature space. Candidate examples are selected according to their similarity to the target representation. We follow the method-specific feature extraction and selection settings described in the original paper while using our common candidate pool, reference sets, and data budget.

B.5 Selection Cost Accounting

We compute data selection costs based on recorded resource usage. For MARS, the one-time API annotation cost covers ability and quality annotations for the entire candidate pool and all five target reference sets. Candidate and reference examples are processed in batches of up to 100 examples per API request.

Table 7 summarizes the recorded API usage. Annotating the 270,699 candidate examples required 2,707 API requests, and annotating the 565 target reference examples required 7 additional requests. Across all annotation requests, we recorded 90,649,691 input tokens and 9,680,204 output tokens. The observed input cache hit rate during annotation was approximately 40%. The recorded output-token counts include reasoning tokens reported by the API, which are therefore not counted separately in the cost calculation.

Using the API prices in effect for our cost audit, we charge \$0.20 per million uncached input tokens, \$0.02 per million cached input tokens, and \$1.20 per million output tokens. With the observed input cache hit rate of approximately 40%, the one-time API annotation cost is

$$C_{\text{API}} = \frac{90{,}649{,}691\,(0.6 \times 0.20 + 0.4 \times 0.02) + 9{,}680{,}204 \times 1.20}{10^6} = \$23.219405.$$

Table 7: Recorded API usage for constructing the Candidate Ability Atlas and annotating the five target reference sets.

Annotation Set	# Examples	# Requests	Input Tokens	Output Tokens	Cost ($)
Candidate pool	270,699	2,707	90,545,388	9,660,364	23.1822
Target reference sets	565	7	104,303	19,840	0.0372
Total	271,264	2,714	90,649,691	9,680,204	23.2194

Across the four base models, reference NLL computation requires a cumulative 0.341715 GPU-hours. Using a cost of $4.09 per H100 GPU-hour, the total selection cost of MARS is

$$C_{\text{total}} = C_{\text{API}} + 0.341715 \times 4.09 = \$24.617019 \approx \$24.62.$$

C Additional Experimental Results

Table 8 compares MARS with the variant without model calibration across all four base models at a 5% data budget. Calibration improves the five-task average on every model and improves 19 of the 20 model–task settings.

Table 8: Effect of reference-based model calibration across four base models with a 5% data budget. The variant without model calibration removes NLL percentile-rank weighting from both ability allocation and reference profiles. Avg. is the mean across five tasks. Bold indicates the better result within each model.

Variant	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
Llama-3.1-8B						
MARS	**64.83**	**64.72**	**73.20**	**66.11**	**49.60**	**63.69**
w/o model calibration	63.22	62.78	70.04	56.33	48.20	60.11
Qwen2.5-7B						
MARS	**74.26**	**66.11**	**72.22**	**78.62**	**62.00**	**70.64**
w/o model calibration	72.15	64.91	68.75	76.57	60.80	68.64
OLMo-2-7B						
MARS	**64.01**	**49.44**	**65.92**	**67.55**	**25.60**	**54.50**
w/o model calibration	63.21	47.96	63.75	66.26	24.80	53.20
Gemma-3-4B-PT						
MARS	**58.43**	**56.30**	74.44	**51.86**	**46.40**	**57.49**
w/o model calibration	57.89	53.98	**74.83**	51.33	46.00	56.81

C.1 Sensitivity to the NLL Weighting Scheme

We examine how the NLL weighting scheme affects reference-based model calibration on Llama-3.1-8B with a 5% data budget. We compare NLL percentile ranks, L1-normalized log1p-transformed NLL, and L1-normalized raw NLL. For the latter two schemes, the reference weights are obtained by normalizing the transformed or raw NLL values by their sum over the target reference set:

$$u^{\theta}_{\text{log1p}}(r) = \frac{\log\big(1 + \ell^{\theta}(r)\big)}{\sum_{r' \in \mathcal{R}_T} \log(1 + \ell^{\theta}(r'))}, \tag{14}$$

and

$$u^{\theta}_{\text{raw}}(r) = \frac{\ell^{\theta}(r)}{\sum_{r' \in \mathcal{R}_T} \ell^{\theta}(r')}. \tag{15}$$

For each variant, the same weighting scheme is used for both ability allocation and reference-profile weighting, while all other selection, training, and evaluation settings are held fixed.

Table 9: Sensitivity to the NLL weighting scheme on Llama-3.1-8B with a 5% data budget. Each weighting scheme is used for both ability allocation and reference-profile weighting. Avg. denotes the mean across the five target tasks; bold indicates the best result for each metric.

Weighting Scheme	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
Percentile Rank	**64.83**	**64.72**	**73.20**	**66.11**	**49.60**	**63.69**
L1-Normalized Log1p NLL	64.60	63.70	71.15	63.84	47.60	62.18
L1-Normalized Raw NLL	64.62	64.17	71.15	62.62	48.60	62.23

As shown in Table 9, NLL percentile ranks achieve the highest performance on all five target tasks, with an average score of 63.69. They outperform L1-normalized log1p-transformed NLL and L1-normalized raw NLL by 1.51 and 1.46 points on average, respectively. These results support the use of NLL percentile ranks for reference-based model calibration in the evaluated setting.

C.2 Cross-Model Transfer of Selected Subsets

We conduct bidirectional subset transfer between Llama-3.1-8B and OLMo-2-7B to examine the benefit of recipient-specific calibration. For each target task, we apply the subset selected using one model's reference-set feedback to the other model, termed the recipient model. We compare this transferred subset with the subset selected using the recipient model's own reference-set feedback. For each recipient model, we fine-tune and evaluate using the two subsets under identical training and evaluation settings.

Table 10: Cross-model transfer of selected subsets. The arrow denotes the calibration model $\rightarrow$ the recipient model used for fine-tuning and evaluation. Identical models on both sides indicate recipient-specific calibration. Avg. denotes the mean across five target tasks; bold indicates the better result for each recipient model.

Calibration $\rightarrow$ Recipient	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
OLMo-2-7B $\rightarrow$ Llama-3.1-8B	63.42	64.07	72.30	64.67	47.40	62.37
Llama-3.1-8B $\rightarrow$ Llama-3.1-8B	**64.83**	**64.72**	**73.20**	**66.11**	**49.60**	**63.69**
Llama-3.1-8B $\rightarrow$ OLMo-2-7B	63.00	48.61	63.47	66.19	23.00	52.85
OLMo-2-7B $\rightarrow$ OLMo-2-7B	**64.01**	**49.44**	**65.92**	**67.55**	**25.60**	**54.50**

As shown in Table 10, recipient-specific calibration increases the average score from 62.37 to 63.69 on Llama-3.1-8B and from 52.85 to 54.50 on OLMo-2-7B. It improves performance on all five target tasks for both recipient models, including gains of 2.20 and 2.60 points on MBPP, respectively.

These results show that, for the evaluated model pair, directly transferring a subset selected for another model does not match the performance of recipient-specific calibration. This finding supports the design of MARS: reusing the Candidate Ability Atlas across base models while updating ability allocation and reference profiles using the current model's reference-set feedback.

C.3 Full Results for Cross-Model Subset Transfer

Table 11 reports the task-level results corresponding to the average cross-model transfer performance summarized in Table 2. For each target task, we use LESS and NAIT with Llama-3.1-8B to select subsets and directly reuse the resulting subsets to fine-tune Qwen2.5-7B, OLMo-2-7B, and Gemma-3-4B. Each subset contains 13,534 examples, corresponding to a 5% data budget. Each fine-tuned model is evaluated on the target task associated with its training subset under the same training and evaluation settings. MARS instead performs recipient-specific calibration using reference-set feedback from the model being fine-tuned.

As shown in Table 11, MARS achieves the highest average score on all three recipient models, outperforming the better-performing transfer baseline for each model by 1.14, 0.98, and 1.98 points on Qwen2.5-7B, OLMo-2-7B, and Gemma-3-4B, respectively. Although the transfer baselines out-

Table 11: Comparison of MARS with cross-model subset transfer baselines at a 5% data budget. The arrow indicates the transfer direction: the model used for data selection → the recipient model used for fine-tuning and evaluation. Llama, Qwen, OLMo, and Gemma denote Llama-3.1-8B, Qwen2.5-7B, OLMo-2-7B, and Gemma-3-4B, respectively. MARS uses recipient-specific calibration. Avg. denotes the mean across the five target tasks; bold indicates the highest score for each metric within each recipient model.

Method	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
Qwen2.5-7B						
LESS: Llama → Qwen	73.44	65.74	71.06	75.66	61.60	69.50
NAIT: Llama → Qwen	73.07	65.65	71.89	75.82	58.80	69.05
MARS	**74.26**	**66.11**	**72.22**	**78.62**	**62.00**	**70.64**
OLMo-2-7B						
LESS: Llama → OLMo	62.33	48.70	65.02	67.93	23.60	53.52
NAIT: Llama → OLMo	63.00	47.31	63.17	**69.45**	24.00	53.39
MARS	**64.01**	**49.44**	**65.92**	67.55	**25.60**	**54.50**
Gemma-3-4B						
LESS: Llama → Gemma	57.83	53.70	73.66	48.14	44.20	55.51
NAIT: Llama → Gemma	**58.52**	52.41	73.21	37.76	**48.20**	54.02
MARS	58.43	**56.30**	**74.44**	**51.86**	46.40	**57.49**

perform MARS on some individual tasks, neither matches its average performance on any recipient model. These results show that direct reuse of subsets selected for another base model does not match recipient-specific calibration in the evaluated settings.

Figure 5 compares the cumulative selection cost of MARS with the direct-transfer settings above. For LESS and NAIT, data selection is performed only once using Llama-3.1-8B, and the resulting task-specific subsets are directly reused for Qwen2.5-7B, OLMo-2-7B, and Gemma-3-4B. Their cumulative selection cost and GPU-stage elapsed time therefore remain constant as additional base models are introduced.

For MARS, candidate ability and quality annotations are computed once and reused across base models, while each additional base model requires only response NLL computation on the target reference sets and updates to the model-specific ability allocations and reference profiles. With four base models, the cumulative selection costs are $149.99 for LESS transfer, $8.56 for NAIT transfer, and $24.62 for MARS. The corresponding cumulative GPU-stage elapsed times are 9.19, 0.52, and 0.12 hours, respectively. These results show that MARS retains model-specific adaptation with a small incremental calibration cost across base models.

C.4 Robustness to the LLM Annotator

We evaluate sensitivity to the LLM annotator on Llama-3.1-8B by replacing GPT-5.6-Luna with DeepSeek-V4-Flash. We rescore the entire candidate pool and all target reference sets across all six ability dimensions and for data quality, then use the resulting annotations to reconstruct the Candidate Ability Atlas and reference representations. We compare downstream performance across the five target tasks with a 5% data budget.

Table 12: Effect of the LLM annotator on downstream performance of Llama-3.1-8B with a 5% data budget. Each annotator independently scores the entire candidate pool and all target reference sets. Avg. denotes the mean across five target tasks; bold indicates the better result for each metric.

Annotator	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.
GPT-5.6-Luna	**64.83**	64.72	**73.20**	**66.11**	**49.60**	**63.69**
DeepSeek-V4-Flash	64.12	**65.00**	72.80	65.88	48.80	63.32

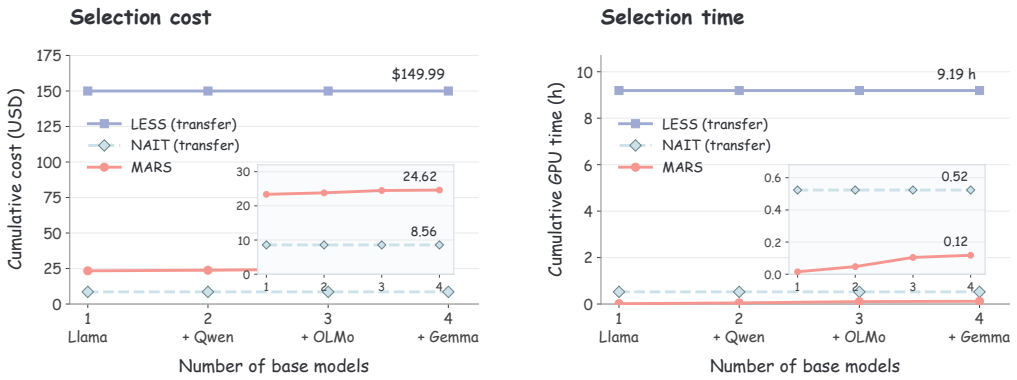


Figure 5: Cumulative data selection cost (left) and GPU-stage elapsed time (right) under direct cross-model subset transfer. LESS (transfer) and NAIT (transfer) perform data selection once using Llama-3.1-8B and directly reuse the resulting task-specific subsets for subsequent base models. MARS reuses its one-time candidate annotations and performs reference-based calibration for each base model.

As shown in Table 12, using DeepSeek-V4-Flash as the annotator yields an average downstream score of 63.32, 0.37 points below the score obtained with GPT-5.6-Luna. Relative to the default annotator, task-level differences range from -0.80 to $+0.28$ points, with a higher score on BBH. The DeepSeek-V4-Flash variant also exceeds MoNA, the strongest baseline on Llama-3.1-8B, by 0.68 points on average.

These results indicate that replacing the default annotator yields similar downstream performance in the evaluated setting. They support the use of the selection framework with both tested annotators, although this comparison is limited to Llama-3.1-8B.

C.5 Sensitivity to the Ability Decomposition

We evaluate the sensitivity of MARS to the six-dimensional ability decomposition on Llama-3.1-8B with a 5% data budget. We consider two types of variants. For the leave-one-out variants, we remove each ability dimension in turn and rerun the selection procedure using the remaining dimensions. To assess whether a coarser functional representation is sufficient, we also merge the six abilities into three semantically related pairs—$(A1, A2)$, $(A3, A4)$, and $(A5, A6)$—grouping Goal / Constraint Following with Context / Evidence Maintenance, Semantic Retrieval with Rule / Logical Reasoning, and Quantitative Reasoning with Code / Procedural Reasoning, respectively. All other selection, training, and evaluation settings remain unchanged.

As shown in Table 13, removing any individual ability reduces the five-task average, with decreases ranging from 0.98 to 4.30 points. The largest effect occurs when Quantitative Reasoning (A5) is removed: performance on GSM8K decreases from 66.11 to 46.47, while the changes on the other tasks are substantially smaller. This concentrated degradation shows that A5 provides selection-relevant information that is particularly important for GSM8K. Removing the other dimensions produces broader performance changes across tasks, showing that the six dimensions provide complementary information and jointly characterize the functional composition of training examples.

Coarsening the representation also consistently reduces performance. Merging the six abilities into three paired dimensions lowers performance on all five tasks and decreases the average from 63.69 to 61.82, with the largest drops on TyDiQA, GSM8K, and MBPP. Grouping semantically related abilities into broader categories therefore removes distinctions that remain useful for ability allocation and within-bucket matching. Together, the dimension-removal and coarsening results support retaining the six-dimensional ability decomposition used in MARS.

Table 13: Sensitivity to the ability decomposition on Llama-3.1-8B with a 5% data budget. *Merged 3-D* combines $(A1, A2)$, $(A3, A4)$, and $(A5, A6)$. Avg. denotes the mean across the five target tasks, and Δ denotes the change in average performance relative to the full six-dimensional representation.

Variant	MMLU	BBH	TyDiQA	GSM8K	MBPP	Avg.	Δ
Full 6-D	64.83	64.72	73.20	66.11	49.60	**63.69**	–
w/o A1	63.99	65.46	70.42	62.62	49.20	62.34	-1.35
w/o A2	64.02	61.30	69.96	63.99	48.60	61.57	-2.12
w/o A3	64.58	64.63	72.76	62.24	47.60	62.36	-1.33
w/o A4	64.23	64.07	70.94	63.68	49.80	62.54	-1.15
w/o A5	64.43	64.17	72.30	46.47	49.60	59.39	-4.30
w/o A6	64.10	63.61	72.44	64.82	48.60	62.71	-0.98
Merged 3-D	64.65	64.26	69.21	63.76	47.20	61.82	-1.88