Fast and Accurate Probing of In-Training LLMs' Downstream Performances

Zhichen Liu[*1], **Tianle Lun**[*1], **Yang Xu**[†1]
Zhibin Wen[1], **Hao An**[1], **Yulin Ou**[1], **Jianhui Xu**[1]
Hao Zhang[2], **Wenyi Fang**[2], **Yang Zheng**[2]
[1] Department of Computer Science and Engineering,
Southern University of Science and Technology
[2] Huawei Technologies Co., Ltd.

Abstract

The paradigm of scaling Large Language Models (LLMs) in both parameter size and test time has pushed the boundaries of AI capabilities, but at the cost of making the traditional generative evaluation paradigm prohibitively expensive, therefore making the latency of LLM's in-training downstream performance evaluation unbearable. However, simple metrics like training loss (perplexity) are not always correlated with downstream performance, as sometimes their trends diverge from the actual task outcomes. This dilemma calls for a method that is computationally efficient and sufficiently accurate in measuring model capabilities. To address this challenge, we introduce a new in-training evaluation paradigm that uses a lightweight probe for monitoring downstream performance. The probes take the internal representations of LLM checkpoints (during training) as input and directly predict the checkpoint's performance on downstream tasks measured by *success probability* (i.e., pass@1). We design several probe architectures, validating their effectiveness using the OLMo3-7B's, OLMo3-32B's, and Apertus-8B's checkpoints across a diverse set of downstream tasks. The probes can accurately predict a checkpoint's performance (with avg. AUROC$>$0.75), have decent generalizability across checkpoints (earlier predicts later), and reduce the computation latency from $\sim$1 hr (using conventional generative evaluation method) to $\sim$3 min. In sum, this work presents a practical and scalable in-training downstream evaluation paradigm, enabling a more agile, informed, and efficient LLM development process. Our code is available at `https://github.com/CLCS-SUSTech/Probe-Prediction`.

1 Introduction

With the continuous scaling of model parameters Kaplan et al. (2020); Hoffmann et al. (2022), the engineering behind training Large Language Models (LLMs) has become increasingly complex, necessitating auxiliary mechanisms for in-training supervision. Unlike traditional supervision that prioritizes training stability Chowdhery et al. (2022), capability-oriented in-training evaluation focuses on verifying whether the capabilities acquired by the model align with expectations for each downstream task Biderman et al. (2023); Groeneveld et al. (2024). This approach involves tracking evaluation metrics across various downstream tasks rather than relying solely on training perplexity (PPL, i.e., loss). When detecting performance deviations in a specific task, researchers can intervene by adjusting data mixtures to rebalance the model's overall competence Xie et al. (2023); Chen et al. (2023). Crucially, perplexity metrics lack this granularity–while perplexity tends to plateau after

[*]Equal contribution.
[†]Corresponding author: `xuyang@sustech.edu.cn`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

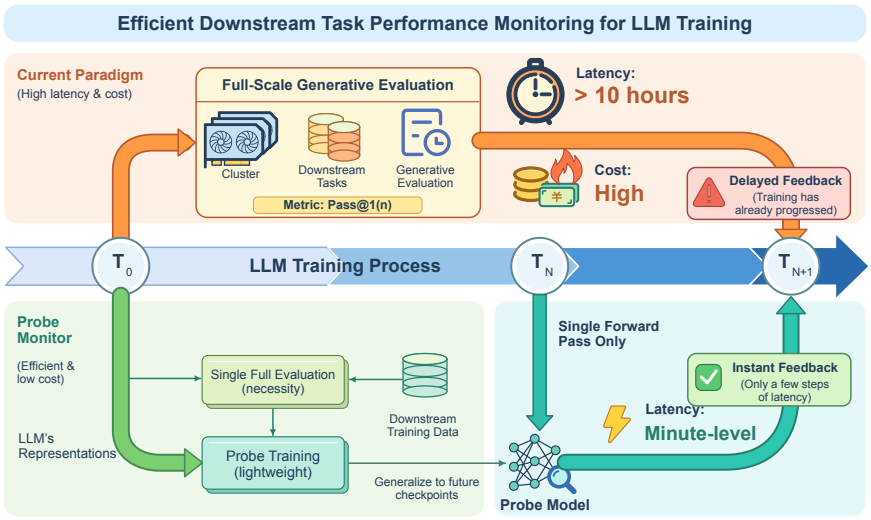


Figure 1: Workflow comparison between the current generative evaluation paradigm and the proposed probing evaluation paradigm for in-training downstream monitoring. Current paradigm suffers from high evaluation cost and excessive latency, resulting in delayed feedback to training progress. In contrast, probe can bypasses the generation process during the evaluation phase, thereby enabling rapid and timely assessment.

convergence, performance on downstream tasks often continues to evolve as training progresses Wei et al. (2022); Suzgun et al. (2022b).

As the capability boundaries of LLMs expand, the paradigm of in-training evaluation is shifting from traditional formats (e.g., multiple-choice) toward generative evaluation Liang et al. (2023); Myrzakhan et al. (2024). This transition is driven by the predominance of generative demands in modern downstream tasks; generative evaluation not only better reflects LLM's generalizability, but also assesses LLM's ability to handle increasingly complex problems Suzgun et al. (2022b); Bubeck et al. (2023). However, this expansion aligns with the evolution of scaling laws: while previous scaling focused on parameter count and training data size, recent trends emphasize test-time scaling Snell et al. (2024). This shift not only escalates the computational cost of a single forward but also significantly increases the token consumption required for generative evaluation. The resulting latency poses a severe bottleneck: for instance, if running a full evaluation on a benchmark takes tens of hours, low-latency in-training monitoring becomes computationally prohibitive and practically infeasible.

For the reasons outlined above, perplexity remains an inadequate surrogate to serve as a reliable proxy for downstream performance. While engineering solutions attempt to reduce evaluation latency by sampling a small-enough subset Polo et al. (2024), this approach inevitably introduces estimation bias Myrzakhan et al. (2024). Consequently, there is an urgent requirement for a new in-training evaluation paradigm that is both accurate and low-latency. To address this challenge, we introduce an efficient probe evaluation framework that utilizes a probe model to monitor downstream performance. The probe model takes the internal states of the model under training and the task input text as inputs, and predicts the corresponding performance metrics (e.g., accuracy and pass@k). Crucially, this allows us to bypass the computationally expensive generation, directly yielding performance predictions.

We investigate several probe architectures, specifically the LoRA-based probe and submodel-based probe, evaluate their performance against some simple baselines. These experiments are conducted under a generative evaluation framework across diverse domains, including general tasks, mathematics, and code generation. The results demonstrate that our method maintains high predictive fidelity (AUROC > 0.75) while drastically reducing evaluation latency, slashing the time required from $\sim$1 hour to $\sim$3 minutes (on OLMo3-7B-base). Consequently, our framework revitalizes the possibility of low-latency, lightweight in-training evaluation once again in the era of modern LLM training.

Our main contributions are summarized as follows:

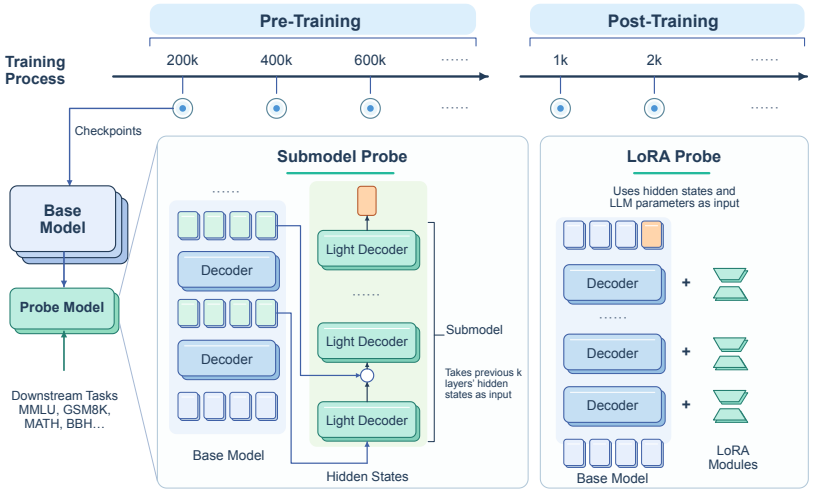


Figure 2: Demonstration of the working process of in-training downstream performance evaluation framework in training stage, and the structure of two lightweight probe models. Training would be applied only in specific checkpoints. Once a probe is trained, it could maintain predictability in the future checkpoints.

❶ We propose a novel framework for efficiently in-training downstream evaluation by leveraging lightweight probe models. The proposed method can give feedback accurately with a low latency.
❷ We design and evaluate three distinct probe architectures (submodel and LoRA) to predict task-specific success probabilities from the model's internal states.
❸ We empirically demonstrate on diverse domains that our trained probes exhibit strong cross-checkpoint generalization, enabling accurate performance prediction for future training steps.
❹ Our method provides a practical, low-latency workflow to the current in-training evaluation paradigm, enabling a more agile and informed LLM training process.

2 Related Works

Predictability AI Zhou et al. (2025) introduces the fundamental idea of predictable AI that an AI system's key validity indicators can be anticipated (e.g., performance). Schellaert et al. (2025) further analyse the predictability of whether the performance of LLM is predictable. Therefore, many works have attempted to better predict the downstream performance of an LLM Chen et al. (2025); Xu et al. (2025). The most popular approach is to train a third-party model for performance prediction Pacchiardi et al. (2024); Koh et al. (2025); Zeng et al. (2025), while another approach predicts the task-level performance directly rather than individual-level performance Ren et al. (2025). These works make efforts on LLMs' performance prediction.

LLM In-training Supervision A vast array of intrinsic metrics inside LLM are proposed to either monitor or stabilize the training process. Dong et al. (2023) explores the rank collapse inside the attention mechanism and uses the relative norm to measure such collapse. Team (2025) discovers the explosion of "MaxLogit" and proposes to stabilize training using QK-Clip. Some works find that the massive values inside attention are crucial for contextual understanding Jin et al. (2025); Gu et al. (2025), while others also find that a stable training process is also related to a stable behavior of attention entropy Zhai et al. (2023); Zhang et al. (2025). Besides in-training monitoring, Du et al. (2025) produces a new perspective for in-training evaluation by building the correlation between the loss value and downstream performance.

Probe Recent work suggests there is a "truthfulness" direction in latent space (Marks & Tegmark, 2024; Azaria & Mitchell, 2023). Some studies employ linear probes to predict whether a model's answers are correct (Li et al., 2025; Han et al., 2024). To reduce reliance on the model's generated responses and speed up prediction, certain approaches directly feed the question's hidden state into the probe to assess the model's ability to provide accurate answers (Snyder et al., 2024; Gottesman &

Geva, 2024). However, correctness and factuality are not entirely linearly encoded (Wehner et al., 2025). Therefore, this work proposes the sub-model probe, which captures the dynamic flow of information within the model to predict its performance.

3 Methods

The primary objective of in-training probe evaluation is to develop a framework for rapidly and efficiently estimating an LLM's performance on downstream tasks without resorting to computationally expensive full-scale evaluations. To achieve this, we first model the generative evaluation process (from input text to generation, then gathering the results to obtain the Pass@1 score) as a regression task, and then introduce a lightweight probe to approximate it.

3.1 Preliminaries: Performance Estimation as Value Approximation

The evaluation process can be modeled as value approximation. Let $\mathcal{M}_\theta$ denote a large language model parameterized by weights θ. Given a prompt x from the dataset $\mathcal{D}$, the model generates a response y following the policy $\pi_\theta(y|x)$. To quantify the performance, we define a ground-truth reward function $R(x, y) \in \{0, 1\}$, which evaluates whether the response y correctly solves the problem in the prompt x. Our goal is to estimate the expected performance of $\mathcal{M}_\theta$ on a downstream task without performing a full-scale evaluation. Adopting a reinforcement learning (RL) perspective, we treat the prompt x as the state $s \in \mathcal{S}$ and the generated response y as the action $a \in \mathcal{A}$. The performance of the model on a specific prompt can be modeled as the state-value function $V^\pi(s)$, which represents the expected reward under the current policy π_θ:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[Q^\pi(s, a)] = \sum_{a\in\mathcal{A}} \pi_\theta(a|s)Q^\pi(s, a) \tag{1}$$

where $Q^\pi(s, a)$ corresponds to the immediate reward (correctness) of taking action a in state s. In our context, since the task is single-turn generation, $Q^\pi(s, a) \equiv R(s, a)$. Thus, predicting the model's accuracy on a prompt is equivalent to approximating the value function $V^\pi(s)$.

3.2 The Monte Carlo Approximation and Pass@1

Because the action space $\mathcal{A}$–encompassing all valid token sequences–is theoretically infinite, calculating the exact expectation $V^\pi(s)$ is intractable. We therefore employ a Monte Carlo estimator. For a given prompt s, we sample n independent responses $\{a_1, a_2, \ldots, a_n\}$ from the policy $\pi_\theta(\cdot|s)$ to get $\hat{v}(s)$:

$$\hat{v}(s) = \frac{1}{n}\sum_{i=1}^{n} R(s, a_i) \tag{2}$$

that $\hat{v}(s)$ acts as an approximation of $V^\pi(s)$. This $\hat{v}(s)$ serves as the ground truth label for our probe. The objective of our probe Φ is to minimize the regression loss over the dataset $\mathcal{D}$:

$$\mathcal{L}(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[||\Phi(s, \theta; \phi) - \hat{v}(s)||^2\right] \tag{3}$$

where ϕ denotes the learnable parameters of the probe.

The widely used Pass@1 metric estimates the probability of correctness when sampling one sample from an n-sample set Chen et al. (2021a). From the aspect of Pass@1, $\hat{v}(s)$ is equivalent to Pass@1(n) when n is held constant. We can analyze this through the definition of Pass@1(n):

$$\text{Pass@1}(n) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i \text{ is correct}) \tag{4}$$

which shows the same formulation as $\hat{v}(s)$ ($\mathbb{I}(\cdot)$ is a decision function). Therefore, modeling Pass@1(n) is equivalent to modeling $V^\pi(s)$. It is worth noting that the commonly used binary metric "accuracy" can be viewed as a special case of Pass@1 when $n = 1$.

3.3 Modeling the Probability of Correctness

3.3.1 Generative Evaluation

The ability of an LLM to answer a given question relies on two fundamental sources of information: its internal **parametric knowledge** (θ), encapsulated within its network weights, and the **contextual knowledge** (S) provided by the input prompt Petroni et al. (2019). The model fuses these two sources to form a high-dimensional latent representation, which we denote as $Rep(S, \theta) \in \mathbb{R}^d$. Subsequently, a generation function, ψ, maps this representation to an answer, and a decision function, $\mathbb{I}(\cdot)$, maps the answer to a binary class that represents correct or incorrect. Therefore, the likelihood p of answering the question correctly can be formulated as:

$$p = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\psi(Rep(S_i, \theta)) \text{ is correct}) \tag{5}$$

The entire process is computationally expensive due to the sequential nature of the generation process ψ. It is infeasible for frequent evaluation during model training or for large-scale model selection.

3.3.2 Probe Evaluation

To overcome this efficiency bottleneck, we propose replacing the whole process of $\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\psi(\cdot))$ with a lightweight and efficient **probe model**, denoted as $f(\cdot; \phi)$, where ϕ represents the probe's parameters. The probe is trained to directly predict the success probability from the LLM's internal representation:

$$p \approx \hat{p} = f(Rep(S, \theta); \phi) \tag{6}$$

While this approximation may introduce some level of fidelity loss, it provides a sufficiently reliable estimate for monitoring training dynamics and identifying potential issues, at a fraction of the computational cost. The design of the probe hinges on the choice of the representation $Rep(S, \theta)$ which extracts from the base model. We explore two distinct probe architectures based on different hypotheses about where the most salient information resides.

3.4 Probe Architectures

In this section, we introduce the architecture of **Submodel Probe**. The architecture about LoRA Probe is included in Appendix B.

Hypothesis 1 We posit that the model's parametric knowledge is encoded in the progressive transformation of internal states across layers. Therefore, the probe should have access to the hidden states from various depths of the LLM.

This architecture captures parametric knowledge via the base model's hidden states. We postulate that the evolution of hidden states across layers forms an "intrinsic dimension" that correlates with the model's confidence and capability. Let $H^{(l)} \in \mathbb{R}^{T \times d}$ be the hidden states of the base model $\mathcal{M}_\theta$ at layer l, where T is the sequence length. We construct a lightweight decoder-only probe model with parameters ϕ, consisting of L_{probe} layers. The probe's input at layer k integrates its previous state and the projected hidden state from the base model. Formally, let $Z^{(k)} \in \mathbb{R}^{T \times d_{probe}}$ denote the hidden state of the probe at layer k. The update rule is:

$$Z^{(k)} = \text{Decoder}^{(k)} \left(Z^{(k-1)} + W_{proj}^{(k)} H^{(k)}; \phi^{(k)} \right) \tag{7}$$

where $W_{proj}^{(k)} \in \mathbb{R}^{d_{model} \times d_{probe}}$ is a learnable projection matrix, and $k \in K$ matches the probe layer index to the corresponding base model layer index (e.g., $K = N$ for a full-layer mapping, or $K < N$ for an early-layer mapping, K is the number of layers in probe, and N is the number of layers in base model). $H^{(k)} = \text{Decoder}^{(k)}(H^{(k-1)}; \theta^{(k)})$ represents the hidden states from base model that contains information from both S and θ. The final prediction is obtained from the last token representation of the final probe layer $\hat{p} = \sigma(\text{Linear}(Z_{[-1]}^{(K)}))$, where $Z_{[-1]}$ means taking the representation at the last position, and σ is the sigmoid function to bound the output within $[0, 1]$.

The probe model can be formulated as:

$$\begin{aligned} f_{sub}(Rep(S,\theta);\phi) =& \text{Decoder}^{(k)} \circ \text{Decoder}^{(k-1)} \circ \dots \\ & \circ \text{Decoder}^{(1)}(W_{proj}^{(1)} H^{(1)};\phi) \end{aligned} \tag{8}$$

where $H^{(1)} = \text{Emb}(X)$ is the output of the embedding layer. Therefore,

$$\hat{p} = \sigma(\text{Linear}(f_{sub}(Rep(S,\theta);\phi)_{[-1]})) \tag{9}$$

3.5 In-Training Probe Evaluation Pipeline

The in-training probe evaluation pipeline requires an initial probe training phase, after which the trained probe can be utilized to evaluate future checkpoints. Consider a set of downstream tasks, each containing a training set and a test set (or manually split it if necessary). The pipeline proceeds as follows:

1. Data Collection: Perform a generative evaluation on the training set to compute the probability of correctness (i.e., Pass@1), which is assigned as the label for each prompt.
2. Probe Training: Formulate the problem as a regression task, using the prompt as input and the calculated probability as the training target.
3. Probe Evaluation: Feed prompts from the test set into the probe to predict the probability of correctness.

In terms of computational cost, phases 1 and 2 are the most intensive. However, once a probe is trained, it can predict the Pass@1 value using a single forward pass, significantly accelerating the evaluation process at phase 3.

4 Experiments

In this section, we empirically validate the effectiveness of the proposed In-Training Probe Evaluation framework. We aim to scrutinize the framework's capability to monitor downstream performance efficiently and accurately throughout the LLM training lifecycle. Specifically, our experiments are designed to address the following research questions:

- **RQ1 (Fidelity):** Can lightweight probes accurately approximate the downstream performance of LLMs solely based on internal representations?
- **RQ2 (Generalizability):** Do probes trained on early checkpoints exhibit transferability to future checkpoints, despite the significant distribution shifts incurred during training?
- **RQ3 (Efficiency):** What is the magnitude of computational acceleration achieved by our framework compared to traditional generative evaluation paradigms?
- **RQ4 (Robustness under Controlled Degradation)**: Do probes still maintain effectiveness on unstable checkpoints?

4.1 Experimental Setup

Datasets Preparation. To ensure a comprehensive assessment of model capabilities, we curate a diverse benchmark suite spanning three primary domains: (1) **Reasoning**: We employ GSM8K Cobbe et al. (2021), MATH Hendrycks et al. (2021b), AIME Veeraboina (2023); Zhang & Math-AI (2025) to evaluate mathematical problem-solving, GPQA Rein et al. (2024) for graduate-level scientific reasoning, and BBH Suzgun et al. (2022a) for symbolic reasoning tasks. (2) **Knowledge**: The MMLU Hendrycks et al. (2021a) benchmark is utilized to assess massive multitask language understanding. (3) **Coding**: We adopt HumanEval Chen et al. (2021b) and MBPP Austin et al. (2021) to evaluate code generation proficiency.

For each dataset, we construct the training targets by performing a standard generative evaluation. Specifically, we define the ground truth label as the empirical success probability, estimated by sampling n responses per prompt (denoted as Pass@1(n)). Detailed statistics and configuration of the datasets are summarized in Table 1. Further implementation details are provided in Appendix A.

Table 1: Summary of datasets used for probe training and evaluation. **N-shot** denotes the number of in-context exemplars. **Stage** indicates the applicable training phase: *Pre* (Pre-training) and *Post* (Post-training/Instruction Tuning). **Sampling Budget (n)** refers to the number of samples generated to estimate the ground-truth Pass@1 score in pre/post stage.

Dataset	Domain	N-shot	Stage	Sampling Budget	Train Size	Test Size
MMLU	Knowledge	5	Pre/Post	8/4	10k	14k
GSM8K	Math	8	Pre/Post	8/4	7.5k	1.3k
MATH	Math	4	Pre/Post	8/4	6k	500
BBH	Reasoning	3	Pre/Post	8/4	5.2k	1.4k
HumanEval	Code	0	Pre/Post	8/4	100	64
MBPP	Code	3	Pre/Post	8/4	710	250
AIME	Math	0	Post	-/16	933	60
GPQA	Science	0	Post	-/8	350	98

Table 2: **Estimation fidelity on Pre-training Checkpoints.** The models are trained and evaluated on the OLMo-3-Base-800k checkpoint. (↑: higher is better; ↓: lower is better).

	Metric	MMLU	GSM8k	MATH	BBH	HumanEval	MBPP	Avg.
Loss fit	AUROC (↑)	0.4588	0.6189	0.6177	0.5219	0.7048	0.5549	0.5795
	MSE (↓)	0.2322	0.1239	0.0399	0.1880	0.1407	0.1755	0.1500
Linear	AUROC (↑)	0.6019	0.5036	0.4714	0.7309	0.5966	0.5818	0.5810
	MSE (↓)	0.1780	0.1310	0.0406	0.1534	0.1773	0.1686	0.1415
Submodel	AUROC (↑)	**0.6737**	**0.7827**	**0.7776**	**0.8014**	**0.9163**	**0.7822**	**0.7890**
	MSE (↓)	**0.1492**	**0.0962**	**0.0333**	**0.1302**	**0.0810**	**0.1395**	**0.1049**
LoRA	AUROC (↑)	0.5519	0.6737	0.6983	0.7731	0.6565	0.7031	0.6761
	MSE (↓)	0.2049	0.1201	0.0436	0.1399	0.1463	0.1501	0.1342

Base Models and Checkpoints. We conduct our experiments mainly using the OLMo3-7B's and OLMo3-32B's checkpoints Olmo et al. (2025), because of its open accessibility to granular checkpoints across the training trajectory (the results of 32B model are demonstrated in Appendix C.4). In addition, we also conduct experiments on Apertus-8BHernández-Cano et al. (2025) to test the transferability of our method. To rigorously evaluate the robustness of our probes against distribution shifts, we utilize a sequence of checkpoints spanning two distinct training phases:

1. **Pre-training Phase:** We sample intermediate checkpoints at varying step intervals during the fundamental language modeling stage (i.e., OLMo-3-1025-7B[1] and Apertus-8B-2509[2]).
2. **Post-training Phase:** We include checkpoints from instruction tuning and alignment stages (i.e., OLMo-3-7B-Instruct[3] and OLMo-3-7B-Think[4]).

Baselines To benchmark the effectiveness of our proposed architectures, we compare them against two established baselines: 1. **Loss fit**, a conventional metric that uses the negative log-likelihood to predict the value. This baseline assumes that lower perplexity correlates with higher downstream accuracy. 2. **Linear probe**, a simplified version of our framework, where each layer trained a linear regressor to predict the final value. This serves to isolate the gains attributed to our proposed non-linear probe architectures (Submodel and LoRA). See Appendix B for more implementation details. We also provide a comparison with subset sampling in Appendix D.

Evaluation Metrics. The probes are trained to regress the success probability $\hat{p}$. To quantify the estimation quality, we applied a binary classification process to the Pass@1 value in metrics computation: We binarized the ground truth Pass@1 values using a threshold of 0.5, i.e., $y = \mathbb{I}(\text{Pass@1} \geq 0.5)$ and report the **Area Under the Receiver Operating Characteristic Curve (AUROC)**. AUROC is threshold-independent and robust to class imbalance, making it an ideal metric to measure how well the probe's predicted scores rank successful instances higher than failed ones. A larger AUROC value means better probe performance: AUROC $= 1.0$ indicates perfect prediction;

[1] `https://huggingface.co/allenai/Olmo-3-1025-7B`
[2] https://huggingface.co/swiss-ai/Apertus-8B-2509
[3] `https://huggingface.co/allenai/Olmo-3-7B-Instruct`
[4] `https://huggingface.co/allenai/Olmo-3-7B-Think`

Table 3: **Forward Transferability Analysis.** We report AUROC scores where probes are trained on earlier checkpoints (rows) and tested on future checkpoints (columns).

	Train Ckpt (k)	Test Ckpt (k)							
		200	**400**	**600**	**800**	**1000**	**1200**	**1400**	**last**
Submodel	200	**0.7639**	0.7346	0.7523	0.7633	0.7504	0.7379	0.7414	0.7513
	400	–	0.7491	0.7654	**0.7764**	0.7647	0.7561	0.7554	0.7657
	600	–	–	0.7773	**0.7850**	0.7697	0.7617	0.7628	0.7685
	800	–	–	–	**0.7890**	0.7719	0.7695	0.7680	0.7731
	1000	–	–	–	–	**0.7728**	0.7608	0.7630	0.7687
	1200	–	–	–	–	–	0.7667	0.7655	**0.7741**
	1400	–	–	–	–	–	–	0.7709	**0.7780**
LoRA	200	**0.6741**	0.6318	0.6457	0.6132	0.5958	0.5728	0.5831	0.5683
	400	–	**0.6729**	0.6733	0.6477	0.5949	0.5741	0.6028	0.5981
	600	–	–	**0.7097**	0.6574	0.6179	0.5964	0.6284	0.6115
	800	–	–	–	**0.6761**	0.6245	0.5891	0.6434	0.6269
	1000	–	–	–	–	**0.6432**	0.5993	0.6313	0.6126
	1200	–	–	–	–	–	**0.6744**	0.6607	0.6433
	1400	–	–	–	–	–	–	**0.6744**	0.6318

AUROC $= 0.5$ indicates random guessing. Additionally, we also reported Mean Squared Error (MSE) $\|\hat{p} - p\|_2$ as an evidence for goodness of fit.

4.2 RQ1: High-Fidelity Performance Estimation

We first assess the fundamental capability of the proposed probes to approximate ground-truth performance metrics (Pass@1) using only internal representations. Table 2 presents the comparative results on the pre-training checkpoints. The Loss Fit baseline yields an average AUROC of only 0.5795, barely outperforming random guessing (0.5). This corroborates the "emergence" phenomenon in LLMs, where training loss cannot correctly correlate with the ability to solve complex reasoning tasks. Similarly, the Linear probe fails to extract sufficient signals (Avg. AUROC of 0.5810), suggesting that the mapping from static representations to task success is highly non-linear. The proposed Submodel architecture significantly outperforms all baselines, achieving an average AUROC of 0.7890 and reducing MSE to 0.1049. From these results, we can conclude that Submodel is sufficient to approximate the correctness with high relevance and low error.

Notably, probes on tasks like HumanEval and BBH can achieve higher AUROCs compared to other tasks, while MMLU is less predictable. The discussion and more related information are in Appendix C, where we demonstrate all results in the pre-training and post-training stages.

4.3 RQ2: Cross-Checkpoint Generalizability

A critical requirement for in-training monitoring is **Forward Transferability**: can a probe trained on an earlier checkpoint (t) accurately forecast the performance of a future checkpoint ($t + \Delta t$)? We investigate this by training probes on specific checkpoints (rows in Table 3) and testing them on subsequent checkpoints (columns).

As shown in Table 3, the Submodel probe demonstrates exceptional temporal robustness. For instance, a probe trained at step 200k maintains an AUROC of ~ 0.75 even when evaluated on the final checkpoint (Last), despite massive updates to the base model's weights. This implies that the mapping between hidden state dynamics and task correctness is an *invariant property* of the model architecture, independent of specific weight values.

Conversely, the LoRA probe suffers from significant degradation when transferred to future checkpoints (e.g., dropping from 0.67 to 0.56). This confirms our hypothesis: LoRA learns to modulate specific weight matrices (W_θ), and as W_θ drifts during training, the learned modulation ΔW_ϕ becomes obsolete. In contrast, the Submodel operates on the *semantic representation space*, which evolves more smoothly.

Transferability on Apertus-8B. To test transferability beyond the OLMo3 model family, we perform the same experiment on Apertus-8B-2509 checkpoints. We train Submodel probes on seven pre-training checkpoints from 400k to 2627k steps and test on the same six benchmarks as Olmo3-7B. As shown in Table 3, probes trained on early checkpoints retain AUROCs of 0.716–0.768 on later checkpoints. In particular, the 400k probe obtains 0.7425 AUROC on the final 2627k checkpoint, and

Table 4: **Forward Transferability on Apertus-8B-2509.** AUROC scores for Submodel probes trained on earlier checkpoints (rows) and tested on later checkpoints (columns).

	Train Ckpt (k)	Test Ckpt (k)						
		400	**800**	**1194**	**1670**	**2000**	**2400**	**2627**
Submodel	400	**0.7812**	0.7163	0.7384	0.7381	0.7482	0.7471	0.7425
	800	–	**0.7265**	0.7413	0.7331	0.7549	0.7417	0.7602
	1194	–	–	**0.7442**	0.7410	0.7541	0.7464	0.7500
	1670	–	–	–	**0.7511**	0.7613	0.7542	0.7614
	2000	–	–	–	–	**0.7707**	0.7557	0.7656
	2400	–	–	–	–	–	**0.7706**	0.7677
	2627	–	–	–	–	–	–	**0.7859**

Table 5: **Controlled degradation test.** Scores are pooled averages over all evaluation samples from the six benchmarks.

Checkpoint	Actual average	Predicted average
OLMo3-800k (original)	0.4977	0.5159
OLMo3-800k + noise 0.1	0.4585	0.5036
OLMo3-800k + noise 0.2	0.2923	0.4556

the mean over all results is 0.7485. These results extend the evidence beyond the OLMo3 trajectory. Complete MSE results are provided in Appendix C.5.

4.4 RQ3: Computational Efficiency and Amortized Cost

While the accuracy and generalizability have been demonstrated, another core focus of the proposed framework is efficiency, that is, whether it can replace traditional generative evaluation to achieve low-latency assessment. We analyze the computational cost of the probe evaluation pipeline compared to standard generative evaluation. The total time cost T can be modeled as $T = T_{init} + N \times t_{eval}$, where N is the number of evaluations. Generative evaluation does not contain a T_{init}, however, it usually contains a high t_{eval}. While probe evaluation requires a preparation phase, its t_{eval} is significantly smaller than generative evaluation. As illustrated in Figure 3, although the probe method requires an initialization phase (data collection + training), it achieves a **15.6**$\times$ **speedup** in inference per checkpoint on the base model–while generative evaluation requires 0.78h to finish an evaluation, probe evaluation only takes 0.05h. The advantage becomes even more pronounced in post-training stages where generation length increases. As detailed in Appendix E, the speedup factors reach **61.4**$\times$ for instruct models and **231.8**$\times$ for think models. This makes our framework particularly suitable for high-frequency monitoring of modern reasoning models.

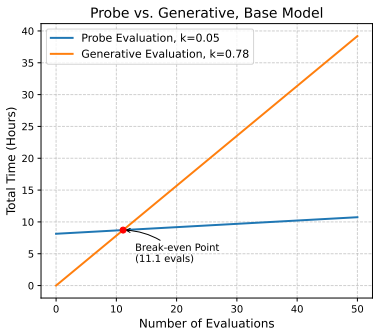


Figure 3: Cumulative time consumption for Probe Evaluation vs. Generative Evaluation on pre-training checkpoints (OLMo-3-Base).

4.5 RQ4: Robustness under Controlled Degradation

The released checkpoints used above are mainly on stable training trajectories. To test the robustness of our probes on unstable checkpoints, we construct two degraded variants of the OLMo3-7B 800k checkpoint by adding zero-mean Gaussian noise to each parameter with relative scales of 0.1 and 0.2, where the standard deviation of the noise for each parameter is proportional to its RMS.

As shown in Table 5, the actual pooled score decreases from 0.4977 to 0.4585 and 0.2923 as the noise level increases. The probe also predicts a decrease, from 0.5159 to 0.5036 and 0.4556, and correctly identifies the stronger degradation under noise level 0.2, although it underestimates the magnitude of

Table 6: **AUROC under controlled degradation.** AUROC is computed over all six benchmarks.

Probe	200k	400k	600k	800k	800k + noise 0.1	800k + noise 0.2
200k	0.7639	0.7346	0.7523	0.7633	0.7566	0.7511
400k	–	0.7491	0.7654	0.7764	0.7660	0.7625
600k	–	–	0.7773	0.7850	0.7754	0.7526
800k	–	–	–	0.7890	0.7758	0.7603

Table 7: **Impact of Input Layer Depth.** Comparison of Submodel probes trained on the first k layers versus the full model (32 layers).

Num of Layers	MMLU	GSM8k	MATH	BBH	HumanEval	MBPP	Avg.
8-layer	0.6079	0.6663	0.6921	0.7894	0.8612	0.5911	0.7013
16-layer	0.6937	0.7463	0.7636	0.7963	0.9184	0.7166	0.7725
Full (32-layer)	0.6737	0.7827	0.7776	0.8014	0.9136	0.7822	0.7890

drop. As shown in Table 6, across probes trained on 200k, 400k, 600k, and 800k, AUROC remains in the ranges 0.757–0.776 and 0.751–0.762 under noise levels 0.1 and 0.2, respectively. We view these results as evidence that the probe can serve as a low-cost, high-frequency warning signal for substantial model degradation.

4.6 Ablation: Layer-wise Information Sufficiency

Finally, we investigate the minimal information required for accurate prediction. The Submodel probe normally aggregates hidden states from all layers. Here, we analyze the performance impact of restricting the input to the first k layers ("Early Exit" strategy). Table 7 reveals that the first 16 layers contain the majority of the signal required for performance estimation (Avg. AUROC 0.7725 vs. 0.7890). However, reducing the scope to only the first 8 layers results in a notable performance drop, indicating that critical reasoning cues are distributed across the mid-to-upper layers of the network. This finding supports the feasibility of further accelerating the pipeline by caching only the upper-layer representations.

5 Conclusion

In this work, we address the critical bottleneck of efficiently monitoring the downstream performance of LLMs during training, where the imperative for frequent monitoring conflicts with the computationally prohibitive cost of standard generative benchmarks. We introduce the In-Training **Probe Evaluation** framework, a novel paradigm that substitutes the high latency generative evaluation paradigm with agile, efficient diagnostic probes. By formulating performance estimation as a value approximation problem, we demonstrate that lightweight probes can accurately regress the probabilities of correctness on downstream tasks from the model's internal representations, bypassing the need for full-scale generation.

Our extensive experiments yield two pivotal insights. First, we establish that the correlation between internal hidden states and task correctness is remarkably invariant to the distribution shifts inherent in model training. Specifically, our Submodel probe architecture exhibits strong *forward transferability*: a probe trained on previous checkpoints can reliably forecast the performance of future checkpoints. Second, we quantify the efficiency gains, demonstrating that our framework reduces evaluation latency by orders of magnitude ($15.6\times$ speed up compared with generative evaluation), rendering high-frequency, granular monitoring feasible once again.

Looking ahead, this work lays the foundation for a more controllable and scientifically grounded LLM development process. The implications extend beyond passive monitoring; our framework offers a viable path toward active training intervention. The high-fidelity signals provided by these probes could serve as real-time feedback mechanisms for dynamic data curriculum learning—allowing developers to adjust data mixtures or hyperparameters on-the-fly to optimize specific downstream capabilities. By decoupling evaluation costs from model scale, we provide a scalable methodology to accelerate the iteration and understanding of next-generation AI systems.

Acknowledge

We sincerely thank all reviewers for giving constructive feedbacks that make this paper much improved. This study is funded by Shenzhen Science and Technology Program (No. JCYJ20240813094612017), Guangdong Province ZJRC Program (No. 2024QN11X145), and Huawei Technologies Co., Ltd. (No. TC20211206598-2024-06).

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Azaria, A. and Mitchell, T. The internal state of an LLM knows when it's lying. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=y2V6YgLaW7.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. URL https://arxiv.org/abs/2303.12712.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021a. URL https://arxiv.org/abs/2107.03374.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021b.

Chen, M., Roberts, N., Bhatia, K., WANG, J., Zhang, C., Sala, F., and Ré, C. Skill-it! a data-driven skills framework for understanding and training language models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 36000–36040. Curran Associates, Inc., 2023.

Chen, Y., Huang, B., Gao, Y., Wang, Z., Yang, J., and Ji, H. Scaling laws for predicting downstream performance in llms, 2025. URL https://arxiv.org/abs/2410.08527.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K.,

Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dong, Y., Cordonnier, J.-B., and Loukas, A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth, 2023. URL https://arxiv.org/abs/2103.03404.

Du, Z., Zeng, A., Dong, Y., and Tang, J. Understanding emergent abilities of language models from the loss perspective, 2025. URL https://arxiv.org/abs/2403.15796.

Gottesman, D. and Geva, M. Estimating knowledge in large language models without generating a single token, 2024. URL https://arxiv.org/abs/2406.12673.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. Olmo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Gu, X., Pang, T., Du, C., Liu, Q., Zhang, F., Du, C., Wang, Y., and Lin, M. When attention sink emerges in language models: An empirical view, 2025. URL https://arxiv.org/abs/2410.10781.

Han, J., Kossen, J., Razzak, M., Schut, L., Malik, S. A., and Gal, Y. Semantic entropy probes: Robust and cheap hallucination detection in LLMs. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024. URL https://openreview.net/forum?id=Zd0XLr6JKn.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

Hernández-Cano, A., Hägele, A., Huang, A. H., Romanou, A., Solergibert, A.-J., Pasztor, B., Messmer, B., Garbaya, D., Ďurech, E. F., Hakimi, I., Giraldo, J. G., Ismayilzada, M., Foroutan, N., Moalla, S., Chen, T., Sabolčec, V., Xu, Y., Aerni, M., AlKhamissi, B., Marinas, I. A., Amani, M. H., Ansaripour, M., Badanin, I., Benoit, H., Boros, E., Browning, N., Bösch, F., Böther, M., Canova, N., Challier, C., Charmillot, C., Coles, J., Deriu, J., Devos, A., Drescher, L., Dzenhaliou, D., Ehrmann, M., Fan, D., Fan, S., Gao, S., Gila, M., Grandury, M., Hashemi, D., Hoyle, A., Jiang, J., Klein, M., Kucharavy, A., Kucherenko, A., Lübeck, F., Machacek, R., Manitaras, T., Marfurt, A., Matoba, K., Matrenok, S., Mendoncça, H., Mohamed, F. R., Montariol, S., Mouchel, L., Najem-Meyer, S., Ni, J., Oliva, G., Pagliardini, M., Palme, E., Panferov, A., Paoletti, L., Passerini, M., Pavlov, I., Poiroux, A., Ponkshe, K., Ranchin, N., Rando, J., Sauser, M., Saydaliev, J., Sayfiddinov, M. A., Schneider, M., Schuppli, S., Scialanga, M., Semenov, A., Shridhar, K., Singhal, R., Sotnikova, A., Sternfeld, A., Tarun, A. K., Teiletche, P., Vamvas, J., Yao, X., Ilic, H. Z. A., Klimovic, A., Krause, A., Gulcehre, C., Rosenthal, D., Ash, E., Tramèr, F., VandeVondele, J., Veraldi, L., Rajman, M., Schulthess, T., Hoefler, T., Bosselut, A., Jaggi, M., and Schlag, I. Apertus: Democratizing Open and Compliant LLMs for Global Language Environments. https://arxiv.org/abs/2509.14233, 2025.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., et al. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jin, M., Mei, K., Xu, W., Sun, M., Tang, R., Du, M., Liu, Z., and Zhang, Y. Massive values in self-attention modules are the key to contextual knowledge understanding, 2025. URL https://arxiv.org/abs/2502.01563.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Koh, W., Suk, J., Han, S., Yun, S.-Y., and Shin, J. Predicting llm reasoning performance with small proxy model, 2025. URL https://arxiv.org/abs/2509.21013.

Li, Q., Geng, J., Chen, Z., Zhu, D., Wang, Y., Ma, C., Lyu, C., and Karray, F. Hd-ndes: Neural differential equations for hallucination detection in llms, 2025. URL https://arxiv.org/abs/2506.00088.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. Holistic evaluation of language models, 2023. URL https://arxiv.org/abs/2211.09110.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Marks, S. and Tegmark, M. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aajyHYjjsk.

Myrzakhan, A., Bsharat, S. M., and Shen, Z. Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena, 2024. URL https://arxiv.org/abs/2406.07545.

Olmo, T., :, Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., Heineman, D., Groeneveld, D., Brahman, F., Timbers, F., Ivison, H., Morrison, J., Poznanski, J., Lo, K., Soldaini, L., Jordan, M., Chen, M., Noukhovitch, M., Lambert, N., Walsh, P., Dasigi, P., Berry, R., Malik, S., Shah, S., Geng, S., Arora, S., Gupta, S., Anderson, T., Xiao, T., Murray, T., Romero, T., Graf, V., Asai, A., Bhagia, A., Wettig, A., Liu, A., Rangapur, A., Anastasiades, C., Huang, C., Schwenk, D., Trivedi, H., Magnusson, I., Lochner, J., Liu, J., Miranda, L. J. V., Sap, M., Morgan, M., Schmitz, M., Guerquin, M., Wilson, M., Huff, R., Bras, R. L., Xin, R., Shao, R., Skjonsberg, S., Shen, S. Z., Li, S. S., Wilde, T., Pyatkin, V., Merrill, W., Chang, Y., Gu, Y., Zeng, Z., Sabharwal, A., Zettlemoyer, L., Koh, P. W., Farhadi, A., Smith, N. A., and Hajishirzi, H. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Pacchiardi, L., Cheke, L. G., and Hernández-Orallo, J. 100 instances is all you need: predicting the success of a new llm on unseen data by testing on a few instances, 2024. URL https://arxiv.org/abs/2409.03563.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases?, 2019. URL https://arxiv.org/abs/1909.01066.

Polo, F. M., Weber, L., Choshen, L., Sun, Y., Xu, G., and Yurochkin, M. tinybenchmarks: evaluating llms with fewer examples, 2024. URL https://arxiv.org/abs/2402.14992.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Ren, K., Cortes-Gomez, S., Patiño, C. M., Joshi, A., Lyu, R., Tang, J., Turcan, A., Yamin, K., Wu, S., and Wilder, B. Predicting language models' success at zero-shot probabilistic prediction, 2025. URL https://arxiv.org/abs/2509.15356.

Schellaert, W., Martínez-Plumed, F., and Hernández-Orallo, J. Analysing the predictability of language model performance. *ACM Trans. Intell. Syst. Technol.*, 16(2), February 2025. ISSN 2157-6904. doi: 10.1145/3706118. URL https://doi.org/10.1145/3706118.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Snyder, B., Moisescu, M., and Zafar, M. B. On early detection of hallucinations in factual question answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 2721–2732, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671796. URL https://doi.org/10.1145/3637528.3671796.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., , and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022a.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022b. URL https://arxiv.org/abs/2210.09261.

Team, K. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Veeraboina, H. Aime problem set 1983-2024, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Wehner, J., Abdelnabi, S., Tan, D., Krueger, D., and Fritz, M. Taxonomy, opportunities, and challenges of representation engineering for large language models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=2U1KIfmaU9. Survey Certification.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P., Le, Q. V., Ma, T., and Yu, A. W. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023. URL https://arxiv.org/abs/2305.10429.

Xu, C., Chen, K., Li, X., Shen, K., and Li, C. Unveiling downstream performance scaling of llms: A clustering-based perspective, 2025. URL https://arxiv.org/abs/2502.17262.

Zeng, H., Hui, K., Zhuang, H., Qin, Z., Yue, Z., Zamani, H., and Alon, D. Can pre-training indicators reliably predict fine-tuning outcomes of llms?, 2025. URL https://arxiv.org/abs/2504.12491.

Zhai, S., Likhomanenko, T., Littwin, E., Busbridge, D., Ramapuram, J., Zhang, Y., Gu, J., and Susskind, J. Stabilizing transformer training by preventing attention entropy collapse, 2023. URL https://arxiv.org/abs/2303.06296.

Zhang, Y. and Math-AI, T. American invitational mathematics examination (aime) 2025, 2025.

Zhang, Z., Wang, Y., Huang, X., Fang, T., Zhang, H., Deng, C., Li, S., and Yu, D. Attention entropy is a key factor: An analysis of parallel context encoding with full-attention-based pre-trained language models, 2025. URL https://arxiv.org/abs/2412.16545.

Zhou, L., Moreno-Casares, P. A., Martínez-Plumed, F., Burden, J., Burnell, R., Cheke, L., Ferri, C., Marcoci, A., Mehrbakhsh, B., Moros-Daval, Y., hÉigeartaigh, S. Ó., Rutar, D., Schellaert, W., Voudouris, K., and Hernández-Orallo, J. Predictable artificial intelligence, 2025. URL https://arxiv.org/abs/2310.06167.

A Data Preparation

A.1 Preparation Process

Data preparation requires splitting datasets into training and test sets. For MMLU and GSM8k, which already contain standard splits, we use them directly. However, other datasets do not provide a training split; therefore, we manually perform a train-test split for BBH, HumanEval, MBPP, and GPQA. Specifically, for MATH, we use MATH-500 Lightman et al. (2023) as the test set and the original MATH dataset (excluding MATH-500) as the training set. For BBH, which consists of multiple subsets, we select 50 samples from each subset for the test set and use the remaining data for training. Finally, we randomly downsample the training sets of MMLU and MATH, as their large size would make the evaluation time cost prohibitive.

A.2 MMLU Prompt Format

```
Think step by step and answer the following questions.

Question: {question}
(A) {option 1} (B) {option 2} (C) {option 3} (D) {option 4}

Answer: {answer}

{the rest 4 shots}

Question: {question}

Answer:  Let's think step by step.
```

A.3 GSM8K Prompt Format

```
Question: {question}
Answer: {chain-of-thought}
### {answer}

{the rest 7 shots}

Question: {question}
Answer:
```

A.4 MATH Prompt Format

```
Please reason step by step, and put your final answer within \boxed{}.

Question: {question}

Answer: {answer}

{the rest 3 shots}

Question: {question}

Answer:
```

A.5 BBH Prompt Format

```
{task description}

QUESTION: {question}
OPTIONS:
```

```
{options}
Let's think step by step.
{chain-of-thought}
ANSWER: {answer}

{the rest 2 shots}

QUESTION: {question}
OPTIONS: {options}
Let's think step by step.
```

A.6 MBPP Prompt Format

```
You are an expert Python programmer, and here is your task: {task}.Your
code should pass these tests:

{tests}
[BEGIN]
{code}
[DONE]
{the rest 2 shots}
You are an expert Python programmer, and here is your task: {task}.Your
code should pass these tests:

{tests}
[BEGIN]
```

A.7 AIME Prompt Format

```
Please reason step by step, and put your final answer within \boxed{}.

{question}
```

A.8 GPQA Prompt Format

```
Please reason step by step, and put your final answer within \boxed{}.

Question:
{question}

Answer:
```

B Baselines

B.1 LoRA Probe

Hypothesis 2 We hypothesize that the most direct access to task-specific parametric knowledge is achieved through lightweight perturbations of the frozen model weights.

This architecture captures parametric knowledge by directly interacting with the weights via Low-Rank Adaptation (LoRA) Hu et al. (2021). We interpret the LoRA mechanism not just as fine-tuning, but as a probe that modulates the frozen parametric knowledge θ based on contextual knowledge S. If we treat the LoRA module AB as the probe model's weight W_ϕ, the standard LoRA formulation can be rewritten as:

$$Y = (W_\theta + AB)X = W_\phi X + W_\theta X \tag{10}$$

therefore the probe model takes both $X \in S$ and $W_\theta \in \theta$ as inputs through the following formulation:

$$\begin{aligned} f_{lora}(Rep(S,\theta);\phi) =& \tilde{D}^{(n)} \circ \tilde{D}^{(n-1)} \circ \dots \\ & \circ \tilde{D}^{(1)}(H^{(1)};\theta,\phi) \end{aligned} \tag{11}$$

where $\tilde{D}(H;\theta,\phi)$ is a standard decoder layer operated by LoRA through $\tilde{W} = W_\theta + AB$. We attach LoRA adapters $\phi = \{A_l, B_l\}_{l=1}^{L}$ to the base model. Unlike standard LoRA for text generation, we optimize these adapters solely for the regression task:

$$\hat{p} = \sigma(\text{Linear}(f_{lora}(Rep(S,\theta);\phi)_{[-1]})) \tag{12}$$

This architecture implicitly learns to measure the "competence" of the base model weights W_θ for the given input X by learning the necessary correction W_ϕ required to map the state to its value estimate.

B.2 Linear Probe

The Linear Probe shares the same paradigm as the proposed probing methods. It collects hidden states from the base model and applies linear regression to each layer to obtain a predicted value. The average of these predicted values is then used to estimate the probability of correctness.

B.3 Loss Curve Fit

Inspired by Du et al. (2025), who utilized regression to link loss values to downstream performance, we model the relationship between the model's loss on a given input and its final probability of correctness. Specifically, we collect loss values and the corresponding correctness probabilities for all prompts and perform linear regression using ordinary least squares. To predict downstream performance, we input the loss of a given prompt into the trained regression model to estimate its probability of correctness.

C Full Results

In this section, we present the full results on each dataset for checkpoints from the pre-training stage (base model) and the post-training stage (instruct and think).

C.1 Base Model

Table 8: Comprehensive AUROC Results for OLMo-3-base across Six Datasets (Train ckpt × Test Ckpt)

(a) MMLU Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.6613	0.6739	0.6748	0.6804	0.7062	0.6875	0.6877	0.6984
400	–	0.6685	0.6734	0.6838	0.7119	0.6884	0.6909	0.6997
600	–	–	0.6750	0.6821	0.7087	0.6849	0.6878	0.6945
800	–	–	–	0.6737	0.7141	0.6969	0.7004	0.7072
1000	–	–	–	–	0.7230	0.7014	0.7062	0.7049
1200	–	–	–	–	–	0.6939	0.6967	0.6950
1400	–	–	–	–	–	–	0.6987	0.7018

(b) GSM8k Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.8082	0.7463	0.7279	0.7376	0.7549	0.7112	0.7253	0.7439
400	–	0.7742	0.7592	0.7594	0.7786	0.7336	0.7374	0.7648
600	–	–	0.7771	0.7654	0.7825	0.7453	0.7489	0.7669
800	–	–	–	0.7827	0.7781	0.7548	0.7522	0.7748
1000	–	–	–	–	0.7732	0.7399	0.7478	0.7568
1200	–	–	–	–	–	0.7564	0.7490	0.7671
1400	–	–	–	–	–	–	0.7554	0.7670

(c) MATH Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.7096	0.6997	0.7525	0.7662	0.7944	0.7490	0.7158	0.7224
400	–	0.6863	0.7576	0.7768	0.8130	0.7544	0.7326	0.7361
600	–	–	0.7520	0.7695	0.8072	0.7484	0.7294	0.7283
800	–	–	–	0.7776	0.7966	0.7562	0.7289	0.7328
1000	–	–	–	–	0.8236	0.7628	0.7322	0.7308
1200	–	–	–	–	–	0.7476	0.7263	0.7303
1400	–	–	–	–	–	–	0.7347	0.7538

(d) BBH Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.7749	0.7583	0.7782	0.7773	0.7546	0.7633	0.7885	0.7927
400	–	0.7709	0.7778	0.7649	0.7565	0.7691	0.7721	0.7804
600	–	–	0.8046	0.7853	0.7625	0.7839	0.7827	0.7844
800	–	–	–	0.8014	0.7765	0.7961	0.8020	0.8024
1000	–	–	–	–	0.7719	0.7732	0.7929	0.7933
1200	–	–	–	–	–	0.7982	0.7965	0.7993
1400	–	–	–	–	–	–	0.8046	0.7972

(e) HumanEval Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.9252	0.8483	0.8667	0.9197	0.7836	0.8298	0.8496	0.8748
400	–	0.8594	0.8913	0.9218	0.8012	0.8655	0.8871	0.9093
600	–	–	0.8952	0.9340	0.8064	0.8626	0.8955	0.9174
800	–	–	–	0.9163	0.8211	0.8538	0.8768	0.8905
1000	–	–	–	–	0.7965	0.8485	0.8732	0.8967
1200	–	–	–	–	–	0.8456	0.8720	0.8967
1400	–	–	–	–	–	–	0.8841	0.9105

(f) MBPP Dataset

Train Ckpt (k)	Test Ckpt (k)							
	200	400	600	800	1000	1200	1400	last
200	0.7043	0.6809	0.6938	0.6986	0.7086	0.6862	0.6812	0.6758
400	–	0.7353	0.7331	0.7514	0.7270	0.7257	0.7122	0.7040
600	–	–	0.7598	0.7738	0.7510	0.7449	0.7323	0.7197
800	–	–	–	0.7822	0.7450	0.7590	0.7478	0.7310
1000	–	–	–	–	0.7485	0.7390	0.7256	0.7294
1200	–	–	–	–	–	0.7583	0.7524	0.7563
1400	–	–	–	–	–	–	0.7477	0.7377

C.2 Instruct Model

Table 9: OLMo-3-Instruct Results

(a) Main Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.7402	0.7261	0.7253	0.7421
200	–	0.7367	0.7379	0.7508
300	–	–	0.7285	0.7452
400	–	–	–	0.7293

(b) MMLU Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.6706	0.6723	0.6670	0.6667
200	–	0.6674	0.6630	0.6631
300	–	–	0.6651	0.6655
400	–	–	–	0.6684

(c) GSM8k Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.7215	0.7267	0.7220	0.7264
200	–	0.7310	0.7238	0.7294
300	–	–	0.7200	0.7264
400	–	–	–	0.7164

(d) MATH Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.8365	0.8416	0.8172	0.8551
200	–	0.8527	0.8354	0.8656
300	–	–	0.8288	0.8665
400	–	–	–	0.8606

(e) BBH Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.8856	0.8749	0.8721	0.8566
200	–	0.8825	0.8852	0.8702
300	–	–	0.8878	0.8777
400	–	–	–	0.8816

(f) HumanEval Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.6215	0.6512	0.6861	0.6922
200	–	0.6887	0.7126	0.7172
300	–	–	0.7381	0.7188
400	–	–	–	0.6266

(g) MBPP Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.7999	0.7718	0.7652	0.7919
200	–	0.7694	0.7755	0.7961
300	–	–	0.7208	0.7553
400	–	–	–	0.7545

(h) AIME Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.7267	0.7348	0.6244	0.7094
200	–	0.7593	0.6591	0.7302
300	–	–	0.6196	0.7179
400	–	–	–	0.7063

(i) GPQA Results

Train Ckpt	Test Ckpt			
	100	200	300	400
100	0.6594	0.5353	0.6484	0.6385
200	–	0.5428	0.6486	0.6342
300	–	–	0.6480	0.6336
400	–	–	–	0.6199

C.3 Think Model

Table 10: OLMo-3-Think AUROC Results (Total 9 Datasets)

(a) Main Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.7946	0.7922	0.7849	0.7969	0.8001	0.7872
400	–	0.7925	0.7934	0.7977	0.8030	0.7859
600	–	–	0.7917	0.8055	0.8108	0.7941
800	–	–	–	0.7967	0.7992	0.7862
1000	–	–	–	–	0.8126	0.7941
1200	–	–	–	–	–	0.7780

(b) MMLU Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.6573	0.6558	0.6556	0.6493	0.6538	0.6541
400	–	0.6626	0.6621	0.6556	0.6600	0.6602
600	–	–	0.6683	0.6638	0.6686	0.6679
800	–	–	–	0.6578	0.6630	0.6630
1000	–	–	–	–	0.6669	0.6664
1200	–	–	–	–	–	0.6649

(c) GSM8k Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.7578	0.7632	0.7408	0.7424	0.7635	0.7540
400	–	0.7581	0.7417	0.7416	0.7642	0.7520
600	–	–	0.7594	0.7588	0.7801	0.7591
800	–	–	–	0.7481	0.7703	0.7564
1000	–	–	–	–	0.7681	0.7465
1200	–	–	–	–	–	0.7430

(d) MATH Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.9299	0.9129	0.8870	0.8938	0.8884	0.8667
400	–	0.9061	0.8816	0.8897	0.8806	0.8623
600	–	–	0.8812	0.8903	0.8866	0.8733
800	–	–	–	0.8853	0.8850	0.8728
1000	–	–	–	–	0.8917	0.8825
1200	–	–	–	–	–	0.8657

(e) BBH Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.9557	0.9542	0.9290	0.9514	0.9465	0.9303
400	–	0.9505	0.9258	0.9460	0.9475	0.9331
600	–	–	0.9188	0.9475	0.9476	0.9298
800	–	–	–	0.9573	0.9482	0.9317
1000	–	–	–	–	0.9533	0.9364
1200	–	–	–	–	–	0.9298

(f) HumanEval Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.7480	0.7434	0.7492	0.7990	0.8090	0.7902
400	–	0.7606	0.8041	0.8000	0.8343	0.7897
600	–	–	0.7646	0.8049	0.8194	0.7946
800	–	–	–	0.7926	0.8021	0.7788
1000	–	–	–	–	0.8462	0.8006
1200	–	–	–	–	–	0.7639

(g) MBPP Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.7191	0.7238	0.7476	0.7454	0.7394	0.7269
400	–	0.7171	0.7452	0.7535	0.7312	0.7180
600	–	–	0.7576	0.7679	0.7626	0.7297
800	–	–	–	0.7387	0.7270	0.7143
1000	–	–	–	–	0.7493	0.7325
1200	–	–	–	–	–	0.7005

(h) AIME Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.6575	0.6903	0.7069	0.7008	0.6855	0.6722
400	–	0.6726	0.7028	0.6834	0.6584	0.6580
600	–	–	0.6992	0.6863	0.6697	0.6610
800	–	–	–	0.6887	0.6765	0.6657
1000	–	–	–	–	0.6663	0.6492
1200	–	–	–	–	–	0.6675

(i) GPQA Results

Train Ckpt	Test Ckpt 200	400	600	800	1000	1200
200	0.6808	0.6294	0.6412	0.6408	0.6509	0.6112
400	–	0.6417	0.6747	0.6551	0.6822	0.6287
600	–	–	0.6522	0.6614	0.6551	0.6120
800	–	–	–	0.6672	0.6709	0.6210
1000	–	–	–	–	0.6500	0.6058
1200	–	–	–	–	–	0.6222

C.4 Results on OLMo3-32B Base Model

Table 11 shows the results on OLMo3-32B based model[5] using submodel probe for prediction.

Table 11: OLMo-3-32B AUROC Results (all tested on checkpoint 656k).

Train Ckpt (k)	Overall	BBH	GSM8k	HumanEval	MBPP	MMLU	MATH
100	0.7473	0.6928	0.7580	0.7885	0.7256	0.7062	0.8125
200	0.7556	0.7365	0.7608	0.7652	0.7512	0.7033	0.8169
300	0.7582	0.7474	0.7636	0.7637	0.7600	0.6982	0.8160
400	0.7605	0.7758	0.7641	0.7373	0.7562	0.7220	0.8078
500	0.7694	0.7832	0.7809	0.7803	0.7500	0.7071	0.8150
600	0.7573	0.7772	0.7770	0.7687	0.7162	0.7021	0.8027

C.5 MSE Results on Apertus-8B

Table 12: Apertus-8B-2509 MSE Results

	Train Ckpt (k)	**Test Ckpt (k)**						
		400	**800**	**1194**	**1670**	**2000**	**2400**	**2627**
Submodel	400	**0.0798**	0.0819	0.0821	0.0922	0.1079	0.1232	0.1381
	800	–	**0.0798**	0.0833	0.0943	0.1057	0.1213	0.1452
	1194	–	–	**0.0820**	0.0950	0.1096	0.1295	0.1603
	1670	–	–	–	**0.0864**	0.1004	0.1169	0.1361
	2000	–	–	–	–	**0.0953**	0.1145	0.1349
	2400	–	–	–	–	–	**0.1029**	0.1186
	2627	–	–	–	–	–	–	**0.1093**

[5] https://huggingface.co/allenai/Olmo-3-1125-32B

C.6 Discussion on Performance

Comparing across datasets, HumanEval, BBH, MATH and MBPP are relatively easy to predict, while MMLU, AIME and GPQA are hard. AIME and GPQA are extremely hard benchmarks that require ultra long reasoning steps to provide the correct answer. Contrary to our initial expectations of poor performance, we were surprised to find that the probe exhibited meaningful predictive capability on these two datasets. Specifically, it generally achieved scores above 0.6 on both instruct model and think model, and even surpassed 0.75 on AIME in instruct model. This suggests a possibility that, with further research, datasets like AIME, which typically require tens of thousands of tokens for evaluation, could be assessed without generation, relying solely on the internal states elicited by the input prompt to predict correctness. For MMLU, we attribute the subpar performance to insufficient training data. Although the full MMLU training set contains nearly 100k samples, we utilized only 10k (concerning the generation time). This sample size is relatively small compared to the $\sim$14k test samples, preventing the probe from fully capturing the dataset's features and ultimately resulting in mediocre predictive capability.

D Comparison with Subset Sampling

Subset sampling is an efficient evaluation method commonly used in industry. It approximates an LLM's performance on a full dataset by evaluating it on a smaller subset. The comparison between probes and subset sampling primarily focuses on two aspects: 1. If both methods are constrained to the same level of efficiency, which one provides a more accurate assessment of the model's performance on the full dataset? 2. Without excessive efficiency constraints, can the probe method maintain its performance on the sampled subset, thereby consistently achieving higher efficiency than the subset approach?

For aspect 1, we constrain the number of evaluation steps to 50 iters. Therefore, to match the probe's efficiency, the time cost of subset sampling is $50 \times T_{subset} = T_{init} + 50 \times t_eval \geq T_{init}$, as t_{eval} is negligible. Then, we can get that $T_{subset} = \frac{T_{init}}{50} = \frac{T_{fullset}+T_{train}}{50} \geq \frac{T_{fullset}}{50}$. Therefore, we set 2% as the proportion of subset size. The evaluation results are shown in Table 13. It demonstrates that compared with subset sampling, probe evaluation obtains an overall smaller absolute error in assessing the performance on the full dataset.

Table 13: Comparison between probe evaluation and subset sampling on absolute error (smaller is better).

Absolute Error	HumanEval	MATH	BBH	MMLU	GSM8k	MBPP
Probe	0.0263	0.0009	0.0447	0.0159	0.0081	0.0952
Subset	0.2305	0.0012	0.1221	0.0320	0.009	0.0761

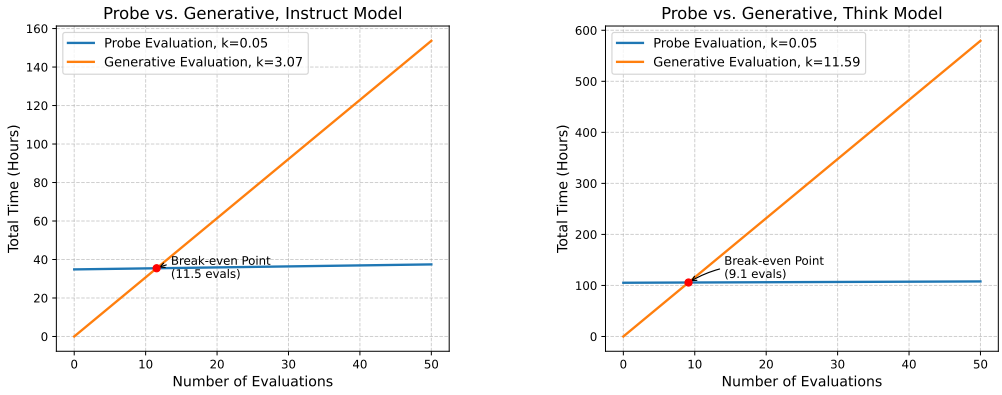
For aspect 2, we set the subset size to 20%, and conduct a probe evaluation on the subset. Table 14 shows the comparison between probe trained on subset and trained on fullset. It demonstrates that even when the subset size is reduced to 20%, the probe's performance on the subset remains close to its performance on the fullset, with only a slight and minor decline.

Table 14: Comparison between probe trained on subset and fullset.

Metrics	Datasets	HumanEval	Math	BBH	MMLU	GSM8K	MBPP
AUROC	Subset	0.8946	0.7766	0.7373	0.6824	0.7232	0.6251
	Fullset	0.9163	0.7776	0.8014	0.6737	0.7827	0.7822
MSE	Subset	0.1035	0.0354	0.1520	0.1566	0.1167	0.1692
	Fullset	0.0810	0.0333	0.1302	0.1492	0.0962	0.1395

E Time Comparison

In this section, we present the time costs for the two models in the post-training stage (Instruct and Think). For the Instruct model, probe evaluation achieves a 61.4x speedup compared to generative evaluation, with its amortized time cost falling below that of generative evaluation after 11 evaluation steps. For the Think model, the efficiency gain reaches 231.8x, becoming more cost-effective than generative evaluation after just 9 evaluation steps.



(a) OLMo-3-7B-Instruct (b) OLMo-3-7B-Think

Figure 4: Cumulative time consumption for Probe Evaluation vs. Generative Evaluation on post-training checkpoints for Instruct and Think models.

The probe preparation time includes the data collection time (i.e., conduct a generative evaluation on trainset) and the probe training time, as indicated by the y-axis intercept of the "probe evaluation" line in Figure 3 and Figure 4. The specific time values are presented in Table 15.

	data collection	probe training
base	5.9h	2.5h
instruct	35.5h	2.6h
think	112h	2.6h

Table 15: Detailed time cost of probe preparation.

F Discussion

F.1 Data Selection in Real-World LLM Training Monitoring

Another discussion concerns the selection of training data. To ensure experimental rigor in this study, we maintained a strict separation between training and test sets, ensuring the test data remained unseen during the probe's training. This setup implicitly tests the probe's ability to generalize from training samples to test samples. However, in real-world industrial scenarios, this type of generalization is not strictly necessary. In practice, the primary objective is to accurately model the mapping from input data to downstream metrics; overfitting to the specific dataset is acceptable, provided that cross-checkpoint generalization is preserved. Consequently, practical applications can forgo the train/test split and train the probe directly on the target evaluation set. This approach would yield higher monitoring accuracy and further reduce the time overhead of the probe preparation phase.

F.2 Limitation

We have discussed the principles of using probes for LLM downstream performance monitoring and verify the feasibility of this approach. However, due to space constraints, we were unable to

investigate how to further improve the predictive performance of the probes. This could be achieved by modifying the probe architecture or altering the input features, and future work will focus on further research in this direction.

F.3 Potential Impact

This work provides an efficient downstream performance monitoring tool for LLM training. However, we have not yet identified any potential negative impacts it might introduce.

G Hardware Specifications

Key specifications relevant to the GPU used in experiments are detailed below:

- Memory Capacity: 48 GB GDDR6
- Memory Bandwidth: 960 GB/s
- CUDA Cores: 14,080
- Tensor Cores: 440
- Single-Precision Performance: 69.3 TFLOPS
- Tensor Performance: 1,108.4 TFLOPS
- Interface: PCIe 4.0 $\times$16

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims about proposing an in-training probe evaluation framework and validate its accuracy and efficiency against standard generative methods.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a section discussing the limitation in further improving the predictive performance of the probes.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper focuses on empirical evaluation, probe architectures, and modeling LLM performance, and does not include formal theoretical proofs or theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed experimental setups, including dataset configurations, model choices, evaluation metrics, and specific prompts used in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper utilizes open-source models and public datasets, and provides url source of these models. Code is available in supplement materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings, such as data preparation, subsets used, and specific prompt formats for each task, are detailed in Section 4.1 and extensively in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports aggregated evaluation metrics, which does not require error bar or statistical analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper explicitly details the hardware specifications used for the experiments in Appendix, including GPU memory, bandwidth, CUDA cores, and TFLOPS.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research involves standard LLM benchmarking and modeling on publicly available data and models, posing no apparent conflict with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper includes a discussion about potential impact in appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new models or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly cites all the existing datasets (e.g., MMLU, GSM8K, MATH) and the base models (OLMo-3) utilized in the research.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper utilizes existing public datasets and models and does not introduce or release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research relies solely on automated generative evaluation and probe evaluation of language models and does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The core methodology involves using LLMs (specifically OLMo3-7B, OLMo3-32B, and Apertus-8B checkpoints) and analyzing their internal representations.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.