TUNEAHEAD: PREDICTING FINE-TUNING PERFORMANCE BEFORE TRAINING BEGINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning large language models (LLMs) is compute-intensive and error-prone: model performance depends sensitively on data quality and hyperparameter choices, and naïve runs can even degrade model performance. This raises a fundamental question: *Can we predict fine-tuning performance before training begins*? We present **TUNEAHEAD**, a lightweight framework for *pre-hoc prediction of fine-tuning performance*. **TUNEAHEAD** encodes each fine-tuning run as a meta-feature vector that combines static dataset descriptors with dynamic probe features from a short simulated run. A predictor maps these features to performance predictions, while SHAP-based attributions provide interpretable diagnostics that reveal which specific features are driving performance. Across 1,300+ fine-tuning runs on Qwen2.5-7B-Instruct, **TUNEAHEAD** consistently outperforms strong baselines such as Early-Stop Extrapolation and ProxyLM. On a held-out test set of 370 runs, by defining 'success' as exceeding a performance threshold, it accurately predicted **89.4% of successful runs (110/123) and 91.0% of failure runs (225/247)**, enabling practitioners to proactively avoid costly unsuccessful runs before training begins. This leads to **computational savings of 58.4%** in total.

1 INTRODUCTION

Fine-tuning large language models (LLMs) has become the standard path to domain adaptation, but it remains costly and unpredictable: performance depends sensitively on data quality and hyperparameter choices, and naïve runs can even *degrade* downstream performance in real-world pipelines (Barnett et al., 2024). For practitioners, the key question is not only ***how*** to fine-tune, but increasingly ***whether*** a run is worth doing at all.

Predicting fine-tuning success. Consider a healthcare provider deciding whether to fine-tune a general LLM on a clinical dataset: a failed run may consume hundreds of GPU hours yet underperform the base model. Without prediction, practitioners often discover *only after training* that performance falls short, wasting both time and budget (Figure 1(A)). This raises a crucial question: *can we predict fine-tuning performance before training begins?*

Prior art and their limitations. Scaling-law analyses (Kaplan et al., 2020) capture general trends across models and datasets but offer limited insight for a *specific* dataset. Proxy models such as COSMOS (Wang et al., 2025) and short-horizon extrapolations (Kuramoto & Suzuki, 2025) demonstrate that low-cost prediction is feasible. However, they aggregate all features into a single score, conflating the base model's own characteristics and limitations with dataset properties, leaving practitioners unable to answer the crucial question of 'why' a run might fail and thus unable to make targeted improvements to avoid costly failures.

Predicting with TUNEAHEAD. We introduce **TUNEAHEAD**, a diagnostic prediction framework that predicts fine-tuning performance *before* training begins. The core idea is to capture two complementary categories of low-cost features. The first one is *static dataset descriptors*, which are computed from the dataset itself to provide a foundational, model-agnostic assessment of its intrinsic quality (*e.g.,* lexical diversity, data size). The second is *dynamic probe features,* which are extracted from a short probe run (*e.g.,* early loss decay, gradient stability); their unique advantage is capturing the model-specific learnability of the data, revealing early signs of optimization instability or data-model mismatch that are invisible to static analysis alone. A lightweight gradient-boosting predictor maps these features to expected performance, while SHAP-based attribution (Lundberg & Lee, 2017)

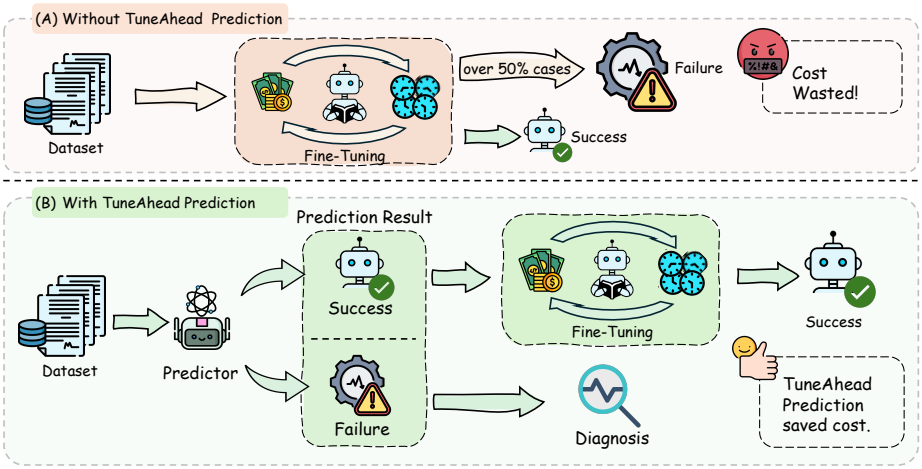


Figure 1: Predicting fine-tuning performance: **(A)** Without **TuneAhead**: failed runs are only identified after training, wasting computational resources and time. **(B)** With **TuneAhead**: low-cost features predict performance in advance, enabling go/no-go decisions and diagnosis for the failure cases.

converts predictions into explanations that reveal which dataset properties matter most. We define a run as 'success' if its predicted score exceeds a predefined practical threshold, and 'failure' otherwise.

With **TuneAhead** (see Figure 1(B)), practitioners can detect unpromising runs *before* training, cutting substantial compute costs while gaining actionable guidance for data refinement. By preventing wasted resources, **TuneAhead** overcomes key limitations of prior work, offering a training-free, interpretable, and dataset-aware approach to reliable fine-tuning.

Figure 2(A) shows that without prediction, all 10 runs (including failures) require ∼30 GPU-hours. With **TuneAhead** (Figure 2(B)), failures are flagged in advance (hatched/blue), so only promising runs are trained, reducing compute to ∼18.7 GPU-hours—a 37.4% saving for 4/6 success/failure cases.

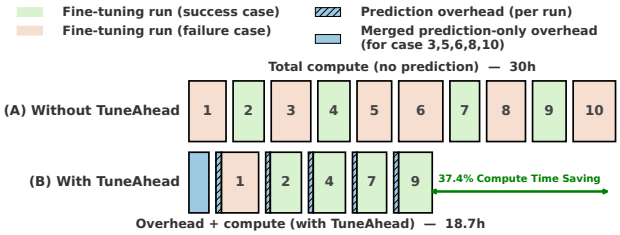


Figure 2: Compute time for 10 runs without (A) vs. with prediction (B).

Contributions. We make the following notable contributions.

(1) *The problem of predicting fine-tuning performance.* We cast fine-tuning outcome prediction as a *pre-hoc, diagnostic meta-learning task*. This formulation supports early go/no-go decisions and principled dataset ranking before expensive training is attempted. (Sec. 3)

(2) *The* **TuneAhead** *framework.* We design a hybrid feature space that combines static dataset descriptors with dynamic probe features, and pairs it with a lightweight predictor and SHAP-based attributions, yielding both accurate predictions and interpretable diagnostics. (Sec. 4)

(3) *Extensive experiments.* We conduct over 1,300 fine-tuning runs on Qwen2.5-7B-Instruct (Qwen Team, 2025; Yang et al., 2025), evaluated on Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2021). **TuneAhead** consistently outperforms SOTA solutions such as Early-Stop Extrapolation, ProxyLM, and Early-Dynamics baseline. (Sec. 5)

2 Related Work

Our work sits at the junction of two threads: *LLM performance prediction* and *dataset-level quality assessment*, with connections to fine-tuning dynamics and interpretability. We position **TuneAhead** as a pre-hoc, dataset-level approach that is both *predictive* and *diagnostic*.

LLM performance prediction. Early efforts predict performances by extrapolating short training curves (Domhan et al., 2015), an approach that struggles with modern LLM fine-tuning where

dynamics can be non-monotonic and late-emerging. Recent works (*e.g.,* LENSLLM (Zeng et al., 2025), Lin et al. (2024)) analyze during-training behavior (such as the pre-power phase) using NTK (Jacot et al., 2020) and scaling laws (Kaplan et al., 2020) on small subsets,sharing that the philosophy of 'fitting small to predict large' involves extrapolating the effect of data quantity. However, they do not diagnose the specific root causes of performance degradation (*e.g.,* data repetition or gradient instability) and profile data quality. A parallel line uses inexpensive surrogates: proxy heads or smaller models can predict final accuracy at a low cost (COSMOS (Wang et al., 2025), PROXYLM (Anugraha et al., 2024), probing-based predictors (Zhu et al., 2022)). However, these methods typically return a single, entangled score that mixes model bias with data properties, offering limited visibility into *why* a run succeeds or fails. Our formulation departs from prior work by combining pre-hoc prediction with an explicit, dataset-level diagnosis.

Data quality assessment. A complementary literature scores data quality at the *instance* level—*e.g.,* Data Cartography (Swayamdipta et al., 2020) and Data Shapley (Ghorbani & Zou, 2019)—and more recently curates instruction-tuning data via refinement pipelines (Refine-n-Judge (Cayir et al., 2025)). Others move toward holistic descriptors: Dataset Nutrition Labels (Holland et al., 2018), distributional measures like MAUVE (Pillutla et al., 2021), and generative teaching evaluations (GENTLE (Aoyama et al., 2023)). They improve transparency but generally stop short of *predicting* a dataset's fine-tuning payoff. **TUNEAHEAD** closes this gap by treating each dataset as a meta-instance and tying aggregated *static* descriptors and *early* interaction features to downstream performance.

Fine-tuning dynamics and interpretability. Work on early training signals (*e.g.,* gradient/loss dynamics) informs our choice of low-cost probes (Jastrzebski et al., 2020; Hao et al., 2019). For interpretability, we adopt model-agnostic SHAP attributions (Lundberg & Lee, 2017) to move beyond predicting toward actionable diagnosis at the dataset level.

3 PROBLEM OF PREDICTING FINE-TUNING PERFORMANCE

This section formalizes the task of predicting fine-tuning performance before actual training. Given a dataset–hyperparameter pair, we seek a **low-cost** predictor that approximates the **expensive ground-truth** score that would be obtained after completing full fine-tuning and evaluation. Specifically, let M be a base LLM (*e.g.,* Qwen2.5-7B-Instruct), and A denote a general fine-tuning algorithm. Fine-tuning model M on the dataset–hyperparameter pair (D_i, H_j) produces an adapted model:

$$M'_{i,j} = A(M, D_i, H_j).$$

We then evaluate $M'_{i,j}$ on a downstream benchmark T (*e.g.,* MMLU) to obtain the *ground-truth performance score* $R_{i,j}$. However, acquiring this score is expensive as it requires a full fine-tuning and evaluation cycle, which motivates the need for prediction.

We thus seek a *low-cost prediction function* F that consumes a *meta-feature vector* $V_{i,j}$ describing the dataset and hyperparameter configuration, producing a predicted performance score $P_{i,j}$ that well approximates the ground-truth score:

$$P_{i,j} = F(V_{i,j}) \quad \text{with} \quad P_{i,j} \approx R_{i,j}.$$

The prediction function F is trained on a meta-dataset of past fine-tuning experiments drawn from an empirical distribution Dist over pairs (D_i, H_j). Minimizing an appropriate loss Δ (such as mean squared error), we solve

$$\min_F \; \mathbb{E}_{(D_i,H_j)\sim\text{Dist}} \Big[\Delta\big(F(V_{i,j}),\, R_{i,j}\big) \Big].$$

This formulation supports several key practical goals: (i) *making go/no-go decisions* before expensive fine-tuning; (ii) *ranking dataset and hyperparameter settings* for resource allocation; and (iii) *diagnosing fine-tuning performance* by linking predictions to dataset and hyperparameter characteristics.

Why pre-hoc failure prediction is feasible. Failed fine-tuning runs often leave clear, low-cost features. Examples include dataset-model mismatch (*e.g.,* high reference perplexity), redundancy or limited diversity (flat or noisy short-horizon progress), and unstable optimization dynamics (volatile gradients and irregular loss decay). Notably, failures are often easier to detect than successes: even a single **strong deficiency can reliably indicate likely failure**, enabling early *rule-out* and data-centric remedies at minimal cost.

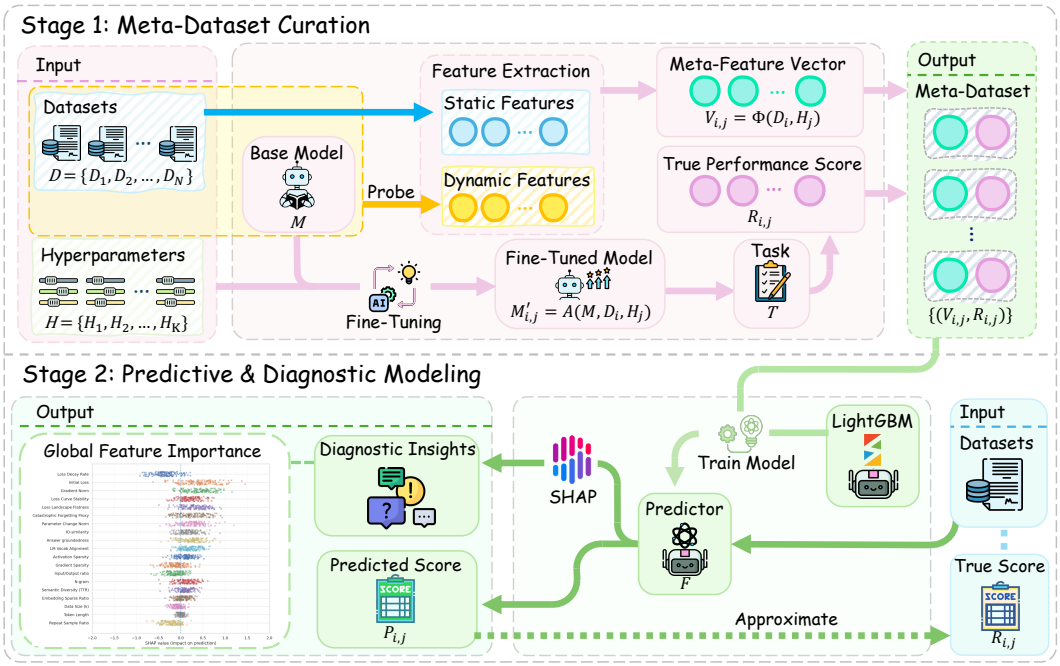


Figure 3: **TuneAhead** Overview. Stage 1 (Meta-dataset curation) builds meta-feature vectors $V_{i,j}$ by combining static features with dynamic features. Stage 2 (Predictive & Diagnostic Modeling) maps $V_{i,j}$ to performance predictions and uses SHAP for interpretable diagnostics.

4 The TuneAhead Framework

4.1 Design Goals and Framework Overview

Design Goals. We identify three key goals that guide our framework design for predicting fine-tuning performance: ***(G1) Low-cost yet informative features:*** The meta-features must respect a computational cost budget while still effectively making predictions. ***(G2) Reliable and generalizable prediction:*** Predictions must be accurate, well-calibrated, and generalizable across diverse datasets. ***(G3) Diagnostic interpretability:*** Predictions must come with human-interpretable attributions that highlight actionable guidance for targeted improvement.

Framework Overview. To satisfy these three goals, **TuneAhead** is structured into two complementary stages: meta-dataset curation (stage 1) and predictive & diagnostic modeling (stage 2). Figure 3 illustrates the framework overview. Stage 1 constructs a compact meta-feature vector $V_{i,j}$ for each dataset-hyperparameter pair by combining static features that summarize dataset-intrinsic properties and dynamic features (G1) that capture early training behavior via a short, fixed-budget probe run (100 steps). These features are fed into stage 2 for predicting model performance $P_{i,j}$ (G2) along with detailed diagnostic attributions (G3). These explanations identify specific failure modes such as data mismatch, redundancy, or instability, helping enable early rejection of failure runs and guiding focused data-centric improvements.

4.2 Stage 1: Meta-Dataset Curation

The predictive capability of **TuneAhead** heavily depends on how well the meta-feature vector $V_{i,j}$ characterizes each (D_i, H_j). To balance informativeness with efficiency, $V_{i,j}$ integrates two complementary categories of features: *static features* derived from the dataset itself, providing a model-agnostic prior on dataset quality, and *dynamic features* probed from the base model M via a short, fixed-budget run, exposing early signs of instability or mismatch. To ensure our meta-feature vector $V_{i,j}$ is both predictive (effective) and computationally lightweight (efficient), we construct it through a rigorous pipeline.

Feature Selection Process. Instead of using an arbitrary feature set, we initiated our framework with a broad candidate pool of over 50 features. We implemented a SHAP-guided selection pipeline (The feature selection procedure is detailed in Appendix A.3) to distill this pool. Specifically, we

trained a preliminary LightGBM regressor on the candidate set and filtered features based on three criteria:

Global Importance (s_f). For each feature f, we computed the mean absolute SHAP value $s_f = \frac{1}{N}\sum_{i=1}^{N}|\phi_{i,f}|$, where $\phi_{i,f}$ is the SHAP value of feature f for run i, and N is the number of runs. Features with s_f below the 15th percentile were pruned as non-informative.

Directional Consistency (c_f). We defined consistency as $c_f = \text{sign}(\rho_f) \cdot \rho_f$, where ρ_f is the Spearman correlation between feature f and the target. We removed features with unstable directions ($c_f < 0.2$) to ensure interpretability (*e.g.,* "lower loss should always be better").

Redundancy Pruning. We iteratively removed highly correlated features unless their exclusion significantly degraded the cross-validated RMSE (i.e., the increase in RMSE $\Delta > 0.01$).

This data-driven process reduced the pool from over 50 features to the 24 most discriminative features (14 static, 10 dynamic).

The final set of *24* meta-features (14 static, 10 dynamic) is chosen to be both *predictive* and *lightweight*, so that the vector $\mathbf{V}_{i,j}$ jointly encodes (i) dataset quality *before* training and (ii) model learnability *during* the first few steps. Below we summarize each group with brief computation notes; full definitions remain in Appendix B.

Static Features. These are dataset-intrinsic descriptors computed prior to training. We retained 14 features across four categories:

- **A. Global statistics.** Basic aggregations from a single tokenization pass provide powerful signals. *Dataset Size* bounds "how much there is to learn"; *Token Lengths* (mean μ_{len} and std σ_{len}) warn of memory pressure; the *Input–Output Length Ratio* separates compressive tasks from elaborative ones, directly influencing adaptation speed; and the *Special-Character Ratio* flags insufficiently cleaned text.
- **B. Lexical diversity.** To distinguish simple lexical repetition from noise caused by excessively rare or idiosyncratic tokens, we compute the Type–Token Ratio and *N-gram Repetition*. We also estimate *Instruction Complexity* via lightweight parse-tree depth to detect prompts that under-challenge the model.
- **C. Semantic diversity.** Surface statistics miss semantic redundancy, so we add *Approximate Duplicates* (using MinHash to avoid quadratic checks) and *Embedding Outlier Ratio* (applying a robust 3-sigma rule: $\frac{\left|\{E(s):\|E(s)-\mu\|>3\sigma\}\right|}{N}$ where $E(s)$ is the embedding of sample s, μ and σ are the mean and standard deviation of embeddings over the dataset, and N is the number of samples). We also measure *IO-Semantic Similarity* and *Output Semantic Diversity* to identify uninformative paraphrasing or incoherent scatter.
- **D. Model-based complexity.** We include difficulty measured by computing *Reference Perplexity*. The mean tracks domain mismatch, while the standard deviation reveals within-dataset heterogeneity. We also compute *KL-divergence* between dataset and pretraining-corpus token distributions to quantify vocabulary shift, and *Answer Groundedness* to detect over-copying vs. hallucination.

Dynamic Features. While static features describe the dataset in isolation, they cannot capture how the specific base model interacts with it. We therefore run a standardized 100-step probe fine-tuning. This budget was selected as the optimal cost–accuracy trade-off based on our ablation study (see Fig. 6). From this probe, we extract 10 features focusing on:

- **E. Loss dynamics.** We capture initial difficulty via *Initial Loss*. Crucially, we compute *Loss Decay* by fitting a robust linear slope to the log-loss curve; more negative slopes indicate faster adaptation. *Loss Stability* is quantified by the variance of the probe losses: $\sigma_L^2 = \frac{1}{T}\sum_{t=1}^{T}(L_t - \bar{L})^2$, where L_t is the training loss at probe step t and $\bar{L}$ is the mean loss. Larger values of σ_L^2 indicate higher loss volatility, which in turn signals noisy optimization.
- **F. Gradient signals.** The mean and variance of the *Gradient Norm* indicate explosion or vanishing risks. *Gradient Consistency* reflects update coherence, while *Gradient Sparsity* flags under-utilized capacity.

G. Generalization cues. We monitor the *Parameter Change Norm* – too little movement fails to learn, while too much often correlates with overfitting. Finally, we compute a *Landscape Flatness* proxy by perturbing parameters and observing loss change; flatter regions in the probe tend to generalize better.

Efficiency Analysis. A critical requirement for pre-hoc prediction is low overhead. We conducted a detailed cost analysis (see Appendix C). The static feature extraction constitutes a one-time offline cost (approximately 0.2% of full training time), and the dynamic probe requires only about 4.3% of the compute of a full fine-tuning run. Thus, the total cost of **TUNEAHEAD** is $< 5\%$ of the standard training budget, yielding a high return on investment given the 58.4% total compute savings.

4.3 STAGE 2: PREDICTION AND DIAGNOSTIC MODEL

Based on the meta-feature vector extracted in stage 1, we aim to train a lightweight predictor that both well approximates the fine-tuning performance $R_{i,j}$ (G2) while also explaining why a run is likely to succeed or fail (G3). We adopt LightGBM, a gradient-boosted tree model particularly well-suited for heterogeneous, tabular meta-features. As demonstrated in Appendix D, LightGBM achieves accuracy comparable to state-of-the-art alternatives (*e.g.,* SVR) while providing significantly better interpretability and scalability. These properties make it a principled design choice rather than a simple off-the-shelf baseline.

In addition to its powerful prediction capability, LightGBM seamlessly integrates TreeSHAP, a theoretically rigorous method for Shapley value attribution. The SHAP framework decomposes each prediction $P_{i,j}$ into additive contributions from individual meta-features. Unlike black-box or proxy baselines that provide only an opaque overall score, our model delivers transparent attributions. For example, a predicted failure can now be traced back to low lexical diversity (static feature) or unstable gradient norms (dynamic probe feature), which guide targeted improvement.

5 EXPERIMENTS

Setup. We validate **TUNEAHEAD** across diverse data and hyperparameter settings on the MMLU task, using Qwen2.5-7B-Instruct, Llama-3-8B-Instruct, and Qwen2-0.5B as base models. All implementation details, data curation protocols, model training procedures, calibration experiments, and baseline implementations are moved to Appendix A. Ground-truth labels are seed-averaged MMLU test accuracy from full LoRA fine-tuning; unless stated, we average over three seeds and evaluate on a held-out test split. To test cross-benchmark/task generalization, we conduct experiments on **TruthfulQA** (Lin et al., 2022)(see Table 3) and on a **open ended generation task(summarization)**.

Baselines. We compare **TUNEAHEAD** against (1) literature/practice-inspired baselines and (2) ablation variants: (i) *Early-Stop Extrapolation* — linear extrapolation from the 100-step probe validation loss (Domhan et al., 2015; Adriaensen et al., 2023); (ii) *Early Training Dynamics* — a method hypothesizes that the speed of loss decay in the initial steps is the strongest predictor of final convergence. We implemented this established heuristic using the slope of the probe loss curve . (Paul et al., 2023; Zhou et al., 2021); (iii) *Domain-Proxy Baseline* — method using the pretrained model's perplexity on the target data as a standard proxy for "difficulty" or "distribution shift" to do data selection or prediction without needing full fine-tuning (Gururangan et al., 2020; Harada et al., 2025; Huang et al., 2022); (iv) *ProxyLM* (Anugraha et al., 2024) — regress on proxy models' accuracy (*e.g.,* small LMs) optionally combined with dataset features (*e.g.,* TTR, vocabulary size and average token length); (v) **TUNEAHEAD**-*Static-Only* — LightGBM trained on static dataset-intrinsic features only; (vi) **TUNEAHEAD**-*Dynamic-Only* — LightGBM trained on dynamic 100-step probe features only. All baselines share the same train/val/test splits and are tuned on the validation set;

Metrics. We report complementary metrics on the test set, all in percentage-point (pp) units for accuracy predictions: *RMSE* measures absolute error in the model performance prediction; R^2quantifies explained variance; *Pearson* r measures linear correlation between $P_{i,j}$ and $R_{i,j}$; *Acc@kpp* is the fraction of predictions with $|P_{i,j} - R_{i,j}| \leq k$ percent. We additionally provide per-domain breakdowns in Appendix A.7, calibration curves in Appendix A.8, and 95% CIs (bootstrap) with paired permutation tests for significance in Appendix A.9.

Exp-1: Predicting Fine-Tuning Performance and Generalization. In the first set of experiments, we establish the end-to-end predictive strength of **TUNEAHEAD** when both dataset properties and

Table 1: Main results on predicting fine-tuning performances (MMLU). Arrows indicate direction: RMSE↓, R^2 ↑, r ↑, Acc@$k$pp↑. **TUNEAHEAD** Pareto-dominates all baselines.

Method	RMSE ↓	R^2 ↑	r ↑	Accuracy (±$k$pp) ↑		
				k=1	k=2	k=3
Early-Stop Extrapolation	7.43	0.81	0.90	11.2	23.9	32.8
Domain-Proxy Baseline	6.58	0.85	0.92	8.6	22.0	32.8
TUNEAHEAD-Static-Only	3.50	0.96	0.99	14.9	32.8	49.3
TUNEAHEAD-Dynamic-Only	3.38	0.96	0.99	19.8	35.8	55.6
Early-Dynamics Baseline	3.33	0.96	0.98	29.9	50.0	67.5
ProxyLM	2.11	0.98	0.99	40.7	67.9	85.8
TUNEAHEAD (Full)	**1.47**	**0.99**	**0.99**	**50.0**	**82.5**	**95.1**

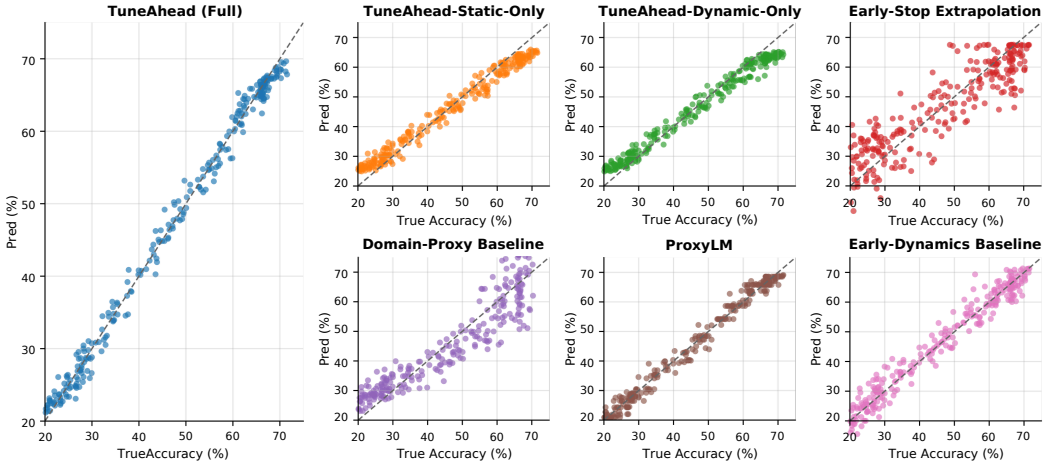


Figure 4: Predicted vs True accuracy across methods. The diagonal line (y=x) indicates a perfect prediction.

Table 2: Cross-model generalization with baselines. Arrows indicate direction: RMSE↓, R^2 ↑, r ↑, Acc@$k$pp↑.

Base Model	**Method**	**RMSE↓**	$\boldsymbol{R^2}$ ↑	$\boldsymbol{r}$ ↑	Accuracy (±$k$pp) ↑		
					k=1	k=2	k=3
Llama-3-8B-Instruct	Domain-Proxy Baseline	10.01	0.66	0.81	2.1	7.4	16.1
	Early-Stop Extrapolation	10.46	0.66	0.81	3.7	8.8	17.8
	Early-Dynamics Baseline	7.33	0.78	0.88	21.5	37.7	48.6
	ProxyLM	6.76	0.86	0.93	30.9	45.8	62.5
	TUNEAHEAD (Full)	**5.02**	**0.86**	**0.93**	**36.0**	**55.8**	**73.3**
Qwen2-0.5B	Early-Stop Extrapolation	10.41	0.69	0.83	6.3	11.7	18.7
	Domain-Proxy Baseline	9.37	0.73	0.85	10.0	16.4	25.3
	Early-Dynamics Baseline	7.47	0.81	0.90	19.3	28.8	39.7
	ProxyLM	4.00	0.90	0.95	35.6	53.7	72.1
	TUNEAHEAD (Full)	**3.75**	**0.91**	**0.95**	**39.2**	**58.4**	**74.6**

LoRA hyperparameters vary. This set of experiments is designed to validate the accuracy and generalizability of our predictor (G2).

We construct a meta-dataset of over 1,300 complete fine-tuning runs spanning heterogeneous instruction-tuning sources and typical LoRA settings. Unless otherwise specified, all experiments reported in the main text use Qwen2.5-7B-Instruct as the base LLM, with ground truth defined as the *seed-averaged* MMLU test accuracy from full fine-tunes. Curation and training details are provided in Appendix A. To further test generalizability across architectures and scales, we also construct additional meta-datasets on Llama-3-8B and Qwen2-0.5B, experiment results reported in Table 2.

TUNEAHEAD *sets a new accuracy bar for this prediction task.* As Table 1 shows, on the held-out test set, it **cuts RMSE by 30%** relative to the strongest non-**TUNEAHEAD** baseline (2.11→1.47 , ProxyLM) and by **80%** relative to Early-Stop Extrapolation (7.43→1.47). Tight-tolerance accuracy improves markedly: **Acc@1pp** +9.3 points (**+22.9%** rel.), **Acc@2pp** +14.6 points (**+21.5%**), and **Acc@3pp** +9.3 points (**+10.8%**) over the best baseline, while maintaining near-perfect ranking

Method	RMSE↓	R^2↑	r↑	MC2 ACC@ $k$pp↑	
				k=3	k=5
Domain-Proxy Baseline	6.33	0.85	0.92	37.0	53.8
Early-Stop Extrapolation	7.10	0.83	0.91	38.4	61.1
TuneAhead-Static-Only	3.90	0.95	0.98	48.6	64.6
TuneAhead-Dynamic-Only	3.60	0.95	0.98	52.3	71.7
Early-Dynamics Baseline	3.45	0.96	0.98	62.5	73.7
ProxyLM	2.54	0.97	0.99	70.6	84.4
TuneAhead (Full)	**2.17**	**0.98**	**0.99**	**74.8**	**88.2**

Table 3: TruthfulQA (MC2) prediction quality. Errors are in percentage points (pp). MC2 ACC@ $k$pp is the fraction of test cases with absolute prediction error $\leq k$ pp.

correlation (Pearson $= 0.99$). These gains reflect the *complementarity* of static data descriptors and dynamic 100-step probe signals: either source alone yields $\sim$ 3.4–3.5 RMSE, whereas their combination reaches **1.47** (**-56%** vs. dynamic-only; **-58%** vs. static-only). Figure 4 visually corroborates these numerical results. The plot for **TUNEAHEAD** (Full) exhibits the tightest clustering of points along the diagonal, confirming its superior accuracy. In contrast, even the strongest baseline, ProxyLM, shows larger deviations, particularly at the tails of the distribution.

Generalization Across Architectures and Scales. A critical question is whether **TUNEAHEAD**'s predictive power is confined to the Qwen2.5-7B-Instruct model. To proactively assess this, we constructed two additional, smaller-scale meta-datasets using Llama-3-8B (different architecture, 400 runs) and Qwen2-0.5B (smaller scale, 450 runs). As shown in Table 2, **TUNEAHEAD** continues to capture predictive signal in both cases, achieving $R^2 = 0.86$ on Llama-3-8B and $R^2 = 0.91$ on Qwen2-0.5B, with reasonably high Acc@kpp. Although RMSE is higher, accuracy is lower than in our primary experiments (due to the significantly smaller meta-datasets), the results provide compelling preliminary evidence that the **TUNEAHEAD** framework is not a model-specific solution.

Generalization Across Benchmarks and Tasks. To mitigate concerns that our features might overfit MMLU, we re-evaluate all 1,300+ fine-tuned checkpoints on **TruthfulQA**. Using the *exact same* meta-features and training protocol, **TUNEAHEAD** achieves consistently low prediction error and high R^2 on TruthfulQA as well (Table 3), outperforming zero/low-cost baselines (ProxyLM, Early-Stop Extrapolation, Early-Dynamics Baseline). Beyond benchmarks, we also build a **Summarization** mini meta-dataset (on Samsum (Gliwa et al., 2019), with ROUGE-L Score (Lin, 2004)) and observe similarly strong predictability; full results and implementation details are in **Appendix Table E.1**.

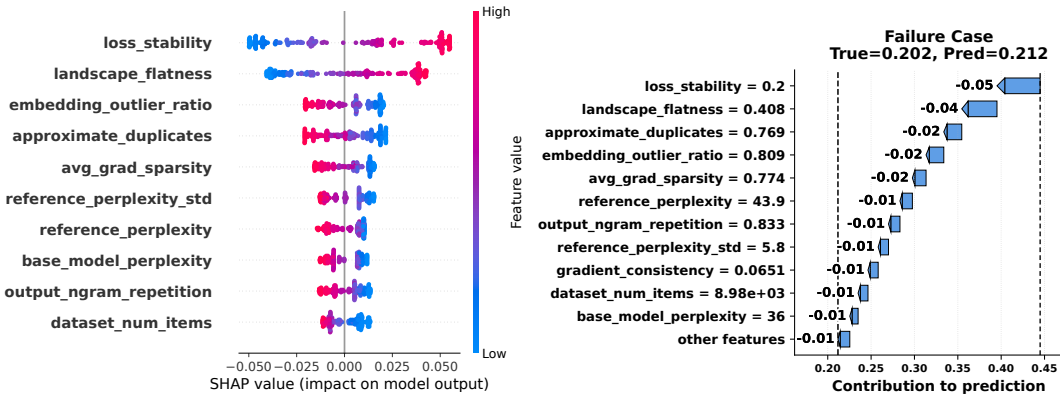
Summary. (i) Heuristics based solely on early learning curves are *not* sufficient for high-precision prediction; (ii) dataset-intrinsic signals provide complementary, non-redundant information; (iii) their integration in **TUNEAHEAD** yields **low-error, tightly calibrated** predictions (95.1% within $\pm$3pp), which is the regime practitioners care about for reliable pre-screening without full fine-tuning; (iv) preliminary cross-model experiments suggest that **TUNEAHEAD** generalizes beyond Qwen2.5-7B-Instruct, maintaining predictive accuracy across different model scales and architectures.

Exp-2: Diagnosis. Beyond predictive accuracy, a core contribution of **TUNEAHEAD** is its ability to provide diagnostic insights (G3). In this section, we use TreeSHAP to analyze the trained model and understand the key drivers of fine-tuning success or failure.

Global Feature Importance. The SHAP analysis in Figure 5(a) identifies the most influential features of fine-tuning performance. *loss stability* and *landscape flatness* emerge as the most critical factors, confirming that a stable initial learning phase is paramount. Following these, data quality metrics like *embedding outlier ratio* and *approximate duplicates* are the strongest negative predictors, quantifying the high cost of noisy data. We trained predictors under multiple random seeds and computed TreeSHAP attributions for each. While the exact ranking of features varied slightly across seeds, the same core set of top features consistently emerged, with only minor shifts in their weights.

Case Study: Diagnosing a Predicted Failure Run. While the summary plot reveals global trends, **TUNEAHEAD**'s utility shines in diagnosing individual runs. To demonstrate its practical value, we conduct an in-depth analysis of a representative failure case, which our model correctly predicted.

The SHAP plot Figure 5(b) explains how meta-features contribute to pushing the performance prediction from the average prediction ($E[f(X)] = 0.45$) down to its final low score ($f(x) = 0.212$).



(a) Global feature importance (SHAP summary). **(b)** Failure case ($E[f(X)] = 0.45$, $f(x) = 0.212$).

Figure 5: (a) SHAP summary plot ranking the global importance of meta-features for predicting fine-tuning success; (b) SHAP waterfall plot for a representative failure case (the model correctly predicted low performance).

The diagnosis reveals a multi-faceted failure, jointly driven by poor learning dynamics and low data quality. On the dynamics side, *loss stability* is extremely low (0.20), whereas successful runs typically range from 0.5 to 0.8; values below 0.3 almost always fail, indicating an unstable trajectory. *Landscape flatness* is also poor (0.408), compared to successful runs clustering around 0.6, with values below 0.5 linked to sharp minima and poor generalization. On the data side, the duplicate rate is 0.769, far above the successful-run median of 0.387. The *embedding outlier ratio* is similarly extreme at 0.809, while successful runs typically stay below 0.5 (median 0.413).

This diagnosis provides a set of targeted, evidence-based prescriptions for the practitioner: **(1) Optimization instability**: Lower the learning rate or adjust optimizer hyperparameters to encourage stable convergence, and **(2) Data quality defects**: Apply semantic de-duplication to reduce redundancy and use embedding-based outlier removal to clean the dataset before fine-tuning.

As a proof of actionability, we applied simple adjustments guided by the diagnosis. Concretely, on the model side, we reduced the learning rate from 3×10^{-5} to 1×10^{-5} and adjusted the optimizer hyperparameters by lowering the AdamW momentum parameters (β_1 from 0.9 to 0.85 and β_2 from 0.999 to 0.98), which encourages more stable convergence. On the data side, we applied SemDeDup (Abbas et al., 2023) for semantic de-duplication, which removes near-duplicates within a cosine threshold of 0.95, and performed embedding-based outlier removal by filtering out samples whose sentence embeddings lie beyond 3 standard deviations from the mean in the representation space. These adjustments improved the run's final MMLU score significantly, from 20.2% to 48.7%.

Exp-3: Ablation Study. To validate our core design choices, we conduct two key ablation studies.

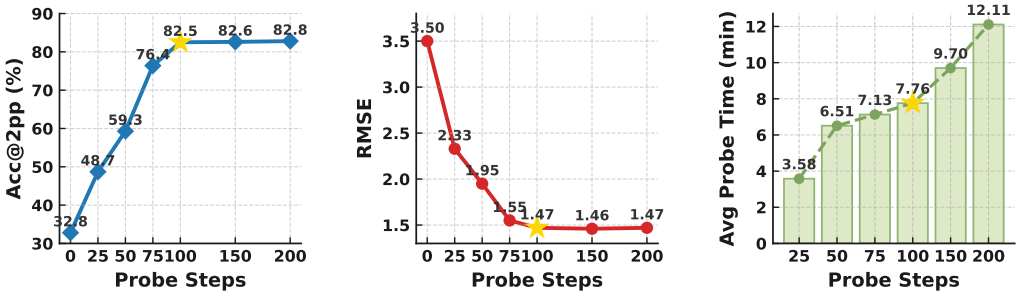
(i) Static vs. Dynamic Feature Ablation. This ablation validates our central hypothesis on the synergy between static and dynamic features. While **Exp-1** established the superior performance of the full model, a deeper look at the ablation results in Table 1 and Figure 4 reveals the nature of this synergy. This result demonstrates that a hybrid view can better achieve reliability.

To evaluate this synergy, we partitioned runs into buckets (Table 4) based on whether the subset predictors succeeded (with Acc@2pp metric). The full model demonstrated robust improvements across all scenarios: (*i*) When only the dynamic predictor succeeded, the full model boosted Acc@2pp from 38.7% to 75.0%; (*ii*) conversely, when only the static predictor succeeded, it raised Acc@2pp from 37.3% to a perfect 100.0%. (*iii*) Even when both subset predictors succeeded, the full model still improved Acc@2pp from 40% to 82.1%, (*iv*) and when both failed, it dramatically mitigated errors, improving Acc@2pp from 12.5% to 70.2%. These rescue effects confirm that static and dynamic features capture different failure modes (data-level flaws vs. model-level instabilities), and their combination is the key to achieving robust prediction.

(ii) Impact of Probe Budget. A key hyperparameter in our framework is the length of the dynamic probe run. To justify our choice of 100 steps, we examine the trade-off between prediction accuracy and the computational cost of the probe. As illustrated in Figure 6, a clear "elbow point" emerges around the 100-step mark, indicating a point of diminishing returns. Specifically, increasing the probe budget from 0 to 100 steps yields a significant improvement in accuracy, with RMSE dropping from

Table 4: Complementarity analysis on the test set. We report RMSE↓ and Acc@2pp↑; Buckets partition runs by whether static-only and dynamic-only predictions are correct.

(i) Rescued-by-Dynamic			(ii) Rescued-by-Static		
Model	RMSE	Acc@2pp	Model	RMSE	Acc@2pp
TUNEAHEAD (Full)	**1.43**	**75.0**	**TUNEAHEAD** (Full)	**0.69**	**100.0**
Static-only	4.90	0.0	Static-only	2.92	37.3
Dynamic-only	3.22	38.7	Dynamic-only	4.60	0.0
(iii) Both-subsets-correct			**(iv) Both-subsets-wrong**		
Model	RMSE	Acc@2pp	Model	RMSE	Acc@2pp
TUNEAHEAD (Full)	**1.49**	**82.1**	**TUNEAHEAD** (Full)	**1.61**	**70.2**
Static-only	3.34	36.2	Static-only	4.63	12.5
Dynamic-only	3.16	40.6	Dynamic-only	5.09	12.5



(a) Acc@2pp vs Probe Steps (b) RMSE vs Probe Steps (c) Time Cost vs Probe Steps

Figure 6: Effect of probe length on prediction accuracy, stability, and time cost. (a) Accuracy at 2pp steadily improves with longer probe runs but exhibits diminishing returns beyond 100 steps. (b) RMSE decreases sharply in the early stage and stabilizes after 100 steps. (c) Average probe time cost grows near-linearly with probe length, with 200 steps requiring about 1.5 times the cost of 100 steps.

3.50 to 1.47 and Acc@2pp rising from 32.8% to 82.5%. However, extending the budget further to 200 steps provides only a negligible gain, with Acc@2pp improving by just 0.3 percentage points to 82.8%. This minimal accuracy improvement comes at a substantial cost, as the average probe time increases linearly, nearly rising from 7.76 minutes to 12.11 minutes. Therefore, we select a 100-step probe as our default configuration, as it offers a robust balance between high predictive power and the low computational overhead mandated by our design goals (G1).

6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

Conclusion. We introduced **TUNEAHEAD**, a framework that predicts fine-tuning performance by combining static and dynamic meta-features. Our experiments show this approach is highly effective, enabling practitioners to avoid costly failed runs and make principled, data-driven decisions.

Limitation. Our primary limitation is generalizability. While **TUNEAHEAD** showed strong performance on several models (see Table 2), these experiments do not yet constitute a comprehensive validation across model families, scales, and task distributions. Specifically, the predictive power of our dynamic features, such as loss stability, is intrinsically tied to the fine-tuning setup. Their importance could shift significantly with different architectures (*e.g.,* dense vs. MoE) or training patterns (*e.g.,* PEFT vs. full-parameter tuning). This implies that while the **TUNEAHEAD** framework is general, a trained predictor instance is likely specific to a model family and requires re-training on a new meta-dataset. Additionally, our compute-saving estimates may vary in other deployments.

Future Work. Our future priorities are twofold. First, we will enhance generalizability not only by expanding our meta-dataset, but more profoundly, by quantifying our feature space's stability across training regimes to develop a meta-learning-based predictor that **adapts to new model families with minimal re-training**. We are actively exploring few-shot calibration methods. Preliminary results suggest that a pre-trained predictor can be adapted to a new model family using only 20-50 anchor runs, enabling near-zero-shot transferability. The second is to leverage **TUNEAHEAD**'s diagnostics to develop systems that use its feature attributions not just as a passive report, but as an active tool for **automated data quality assessment** and **dataset quality improvements**.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to ensure reproducibility. All dataset curation steps, feature definitions, model training, and evaluation protocols are detailed in Appendix A & B. An anonymized code zip file is provided as supplementary material, containing the full implementation of our framework, including modularized training and evaluation code, SHAP-based diagnostic analysis, and calibration routines. To facilitate reproduction, the meta-dataset we provided illustrates the exact data format and enables full, end-to-end execution of our pipeline. All hyperparameters, experimental settings, and statistical testing procedures are explicitly specified in the main text and the Appendix, ensuring that others can reproduce our reported results.

ETHICS STATEMENT

The **TUNEAHEAD** framework presented in this work is intended to improve the computational efficiency and resource management of machine learning research and development. We do not foresee any direct negative societal impacts from its application. However, we acknowledge that, like any tool that accelerates model development, its application to sensitive domains or potentially harmful models should be undertaken with caution and ethical oversight. Furthermore, while **TUNEAHEAD** aims to predict fine-tuning success, the meta-features it relies on (*e.g.,* perplexity, gradient norms) could inadvertently reflect biases present in the base models (*e.g.,* Qwen) or the datasets used for the meta-dataset. Future work could investigate the fairness implications of these predictive features across different demographic and linguistic groups. All research was conducted in compliance with the ICLR Code of Ethics.

LLM USAGE

Our use of Large Language Models (LLMs) in this research was strictly limited to assistance roles, with all core algorithmic ideas, experimental design, and analysis conducted by the human authors.

We used Gemini 2.5 Pro and ChatGPT-5 for English grammar polishing, and asked ChatGPT-5 for suggestions on figure layout and color palettes.

For coding, we consulted ChatGPT-5 and Claude 4.0 for targeted support, such as debugging specific issues and drafting boilerplate scripts (*e.g.,* fine-tuning scaffolds, classic ML baselines). All final code was authored, audited, and verified by the authors.

A portion of the instruction-tuning data in our meta-dataset was synthetically generated using GPT-4o mini and Doubao to increase data diversity. The generation process involved providing seed examples from public datasets and prompting the models to create new, topically related instruction-following pairs. All synthetically generated data underwent a human-led quality control and filtering process before inclusion.

No non-public or sensitive data was used in prompts. LLMs did not contribute any novel algorithmic ideas, hypotheses, or scientific claims presented in this paper. The authors take full responsibility for all content. LLMs do not meet the criteria for authorship.

REFERENCES

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023. URL https://arxiv.org/abs/2303.09540.

Steven Adriaensen, Herilalaina Rakotoarison, Samuel Müller, and Frank Hutter. Efficient bayesian learning curve extrapolation using prior-data fitted networks, 2023. URL https://arxiv.org/abs/2310.20447.

Roee Aharoni and Yoav Goldberg. Unsupervised domain clusters in pretrained language models, 2020. URL https://arxiv.org/abs/2004.02105.

Miltiadis Allamanis, Earl T. Barr, Premkumar Devanbu, and Charles Sutton. A survey of machine learning for big code and naturalness, 2018. URL https://arxiv.org/abs/1709.06182.

David Anugraha, Genta Indra Winata, Chenyue Li, Patrick Amadeus Irawan, and En-Shiun Annie Lee. Proxylm: Predicting language model performance on multilingual tasks via proxy models, 2024. URL https://arxiv.org/abs/2406.09334.

Tatsuya Aoyama, Shabnam Behzad, Luke Gessler, Lauren Levine, Jessica Lin, Yang Janet Liu, Siyao Peng, Yilun Zhu, and Amir Zeldes. Gentle: A genre-diverse multilayer challenge set for english nlp and linguistic evaluation, 2023. URL https://arxiv.org/abs/2306.01966.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks, 2017. URL https://arxiv.org/abs/1706.05394.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In Regina Barzilay and Mark Johnson (eds.), *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 355–362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://aclanthology.org/D11-1033/.

Scott Barnett, Zac Brannelly, Stefanus Kurniawan, and Sheng Wong. Fine-tuning or fine-failing? debunking performance myths in large language models, 2024. URL https://arxiv.org/abs/2406.11201.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models, 2023. URL https://arxiv.org/abs/2301.13188.

Derin Cayir, Renjie Tao, Rashi Rungta, Kai Sun, Sean Chen, Haidar Khan, Minseok Kim, Julia Reinspach, and Yue Liu. Refine-n-judge: Curating high-quality preference chains for llm-fine-tuning, 2025. URL https://arxiv.org/abs/2508.01543.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018. URL https://arxiv.org/abs/1803.11175.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020. URL https://arxiv.org/abs/2003.10555.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL https://arxiv.org/abs/2104.08758.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3460–3468. AAAI Press, 2015. ISBN 9781577357384.

Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, volume 9, pp. 155–161. MIT Press, 1997. URL https://papers.nips.cc/paper/1238-support-vector-regression-machines.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. URL https://arxiv.org/abs/1911.11134.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning, 2019. URL https://arxiv.org/abs/1904.02868.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (eds.), *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://aclanthology.org/D19-5409/.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

Simon Guiroy, Vikas Verma, and Christopher Pal. Towards understanding generalization in gradient-based meta-learning, 2019. URL https://arxiv.org/abs/1907.07287.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020. URL https://arxiv.org/abs/2004.10964.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and understanding the effectiveness of bert, 2019. URL https://arxiv.org/abs/1908.05620.

Yuto Harada, Yusuke Yamauchi, Yusuke Oda, Yohei Oseki, Yusuke Miyao, and Yu Takagi. Massive supervised fine-tuning experiments reveal how data, layer, and training factors shape llm alignment quality, 2025. URL https://arxiv.org/abs/2506.14681.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure, 2019. URL https://arxiv.org/abs/1812.04606.

Dan Hendrycks, Chris Burns, Santiago Basart, Andy Zou, Mateusz Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017. URL https://arxiv.org/abs/1712.00409.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards, 2018. URL https://arxiv.org/abs/1805.03677.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

W. Ronny Huang, Cal Peyser, Tara N. Sainath, Ruoming Pang, Trevor Strohman, and Shankar Kumar. Sentence-select: Large-scale language model data selection for rare-word speech recognition, 2022. URL https://arxiv.org/abs/2203.05008.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020. URL https://arxiv.org/abs/1806.07572.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks, 2020. URL https://arxiv.org/abs/2002.09572.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, March 2023. ISSN 1557-7341. doi: 10.1145/3571730. URL http://dx.doi.org/10.1145/3571730.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling, 2016. URL https://arxiv.org/abs/1602.02410.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training, 2021. URL https://arxiv.org/abs/2103.00123.

Toshiki Kuramoto and Jun Suzuki. Predicting fine-tuned performance on larger datasets before creating them. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, Steven Schockaert, Kareem Darwish, and Apoorv Agarwal (eds.), *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 204–212, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-industry.17/.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better, 2022. URL https://arxiv.org/abs/2107.06499.

Aodi Li, Liansheng Zhuang, Xiao Long, Minghong Yao, and Shafei Wang. Seeking consistent flat minima for better domain generalization via refining loss landscapes. *arXiv preprint arXiv:2412.13573*, 2024. URL https://arxiv.org/abs/2412.13573.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models, 2016. URL https://arxiv.org/abs/1510.03055.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers, 2020. URL https://arxiv.org/abs/2002.11794.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. Selecting large language model to fine-tune via rectified scaling law, 2024. URL https://arxiv.org/abs/2402.02314.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Marco Loog and Tom Viering. A survey of learning curves with bad behavior: or how more data need not lead to better performance, 2022. URL https://arxiv.org/abs/2211.14061.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017. URL https://arxiv.org/abs/1705.07874.

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Zhiyuan Liu, Maosong Sun, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules, 2025. URL https://arxiv.org/abs/2503.12811.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(97)00011-7. URL https://www.sciencedirect.com/science/article/pii/S0893608097000117.

Nikita Moghe, Arnisa Fazla, Chantal Amrhein, Tom Kocmi, Mark Steedman, Alexandra Birch, Rico Sennrich, and Liane Guillou. Machine translation meta evaluation through translation accuracy challenge sets, 2024. URL https://arxiv.org/abs/2401.16313.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018. URL https://arxiv.org/abs/1808.08745.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels, 2022. URL https://arxiv.org/abs/1911.00068.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013. URL https://arxiv.org/abs/1211.5063.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training, 2023. URL https://arxiv.org/abs/2107.07075.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers, 2021. URL https://arxiv.org/abs/2102.01454.

Qwen Team. Qwen2.5-7b-instruct. Hugging Face model card, 2025. URL https://huggingface.co/Qwen/Qwen2.5-7B-Instruct. Accessed: 2025-08-27.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017. URL https://arxiv.org/abs/1706.05806.

Adriano Rivolli, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C. P. L. F. de Carvalho. Characterizing classification datasets: a study of meta-features for meta-learning, 2019. URL https://arxiv.org/abs/1808.10406.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004. doi: 10.1023/B:STCO.0000035301.49549.88.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics, 2020. URL https://arxiv.org/abs/2009.10795.

Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. Advances and challenges in meta-learning: A technical review, 2023. URL https://arxiv.org/abs/2307.04722.

Jiayu Wang, Aws Albarghouthi, and Frederic Sala. Cosmos: Predictable and cost-effective adaptation of llms, 2025. URL https://arxiv.org/abs/2505.01449.

Shixian Wen and Laurent Itti. Overcoming catastrophic forgetting problem by weight consolidation and long-term memory, 2018. URL https://arxiv.org/abs/1805.07441.

Lei Wu, Zhanxing Zhu, and Weinan E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017. URL https://arxiv.org/abs/1706.10239.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Huan Lin, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

Mark Yatskar. A qualitative comparison of CoQA, SQuAD 2.0 and QuAC. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2318–2323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1241. URL https://aclanthology.org/N19-1241/.

Xinyue Zeng, Haohui Wang, Junhong Lin, Jun Wu, Tyler Cody, and Dawei Zhou. Lensllm: Unveiling fine-tuning dynamics for llm selection, 2025. URL https://arxiv.org/abs/2505.03793.

Zheng Zhao, Shay B. Cohen, and Bonnie Webber. Reducing quantity hallucinations in abstractive summarization, 2020. URL https://arxiv.org/abs/2009.13312.

Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Curriculum learning by optimizing learning dynamics. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 433–441. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/zhou21a.html.

Zining Zhu, Soroosh Shahtalebi, and Frank Rudzicz. Predicting fine-tuning performance with probing, 2022. URL https://arxiv.org/abs/2210.07352.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL https://arxiv.org/abs/1909.08593.

A Additional Experimental Details

A.1 Meta-Dataset Curation

Base Model and Task. All fine-tuning runs use **Qwen-2.5-7B-Instruct**. Downstream performance is measured on **MMLU** for its coverage of knowledge and reasoning across domains.

Dataset Collection $\{D_i\}$. We use public instruction-tuning sources (*e.g.,* Alpaca, Dolly) and both programmatically generated variants and LLM-synthesized examples seeded by these sources, yielding 1,200+ dataset versions. Controlled transformations: (1) **Sub-sampling**: sizes 500–25k; (2) **Domain slicing**: STEM vs. humanities, etc.; (3) **Noise injection**: label noise 0–20%. These produce wide variation in statistical, semantic, and structural properties.

Hyperparameter Space $\{H_j\}$. For each D_i, we sweep LoRA hyperparameters: learning rate $\in \{1\text{e}{-5}, 2\text{e}{-5}, 3\text{e}{-5}\}$, batch size $\in \{8, 16, 32\}$. This covers common practitioner settings and captures realistic data–hyperparameter interactions.

Ground Truth Protocol. For every (D_i, H_j) we run full LoRA fine-tuning to convergence and evaluate on the MMLU test set to obtain $R_{i,j}$. Each $R_{i,j}$ is the mean over 3 seeds to reduce stochasticity.

A.2 Meta-Feature Extraction

We compute a feature vector $V_{i,j}$ per experiment.

Static features. Dataset-intrinsic signals (e.g., semantic diversity, label balance, length stats). Semantic embeddings use `all-MiniLM-L6-v2` for efficiency/quality trade-off. Reference Perplexity is computed under the frozen base model (Qwen-2.5-7B-Instruct).

Dynamic probe features. A 100-step probe run per experiment with AdamW and linear scheduler; we log losses/gradients each step and compute early-optimization descriptors.

Preprocessing. We standardize the full feature matrix by z-score before model fitting.

Sensitivity to Embedding Model Choice. We deliberately selected all-MiniLM-L6-v2 (22M parameters, $d = 384$) to minimize overhead. However, we acknowledge that practitioners might prefer larger models (e.g., e5-large-v2, $\sim$ 335M parameters, $d = 1024$) for higher semantic resolution.

While scaling up the embedding model increases the specific extraction cost, our analysis shows the impact on the total framework efficiency is negligible. The cost consists of two parts:

Inference latency. This scales with parameter count. Using a large model (e.g., 335M parameters) would increase inference time by approximately $10\times$–$15\times$ compared to MiniLM.

Distance calculation. This scales linearly with dimension d (complexity $O(N^2 \cdot d)$). Increasing d from 384 to 1024 increases calculation time by approximately $2.7\times$.

Impact assessment. Despite these increases, static feature extraction remains a one-time, offline process. Even with a large model, extracting features for a standard dataset (e.g., 10k samples) typically takes only a few minutes on a GPU. In contrast to the iterative fine-tuning probe (which runs forward/backward passes on the massive 7B target model), the embedding extraction cost constitutes $< 1\%$ of the total pipeline budget regardless of the model choice. Thus, the TuneAhead framework is robust to the choice of embedding architecture.

A.3 SHAP-Guided Feature Selection

Setup. We train a LightGBM predictor F_θ on a candidate feature set $\mathcal{F}$ and compute SHAP values on the validation set $D_{\text{val}} = \{x_i\}_{i=1}^n$. SHAP ensures an additive decomposition:

$$F_\theta(x_i) = \phi_{i,0} + \sum_{f \in \mathcal{F}} \phi_{i,f}.$$

For each feature f, we summarize its global contribution by the mean absolute SHAP and a direction-consistency statistic:

$$s_f := \frac{1}{n}\sum_{i=1}^{n} |\phi_{i,f}|, \qquad \rho_f := \mathrm{SpearmanCorr}\big(x_{i,f},\, \phi_{i,f}\big),$$

and encode the hypothesized sign by $\eta_f \in \{+1, -1\}$ (whether larger $x_{i,f}$ should increase or decrease the prediction). We use $c_f := \eta_f\, \rho_f$ for signed consistency.

Step 1: Preliminary filtering. Start from 50+ candidates (30 static features and 27 dynamic features). Through small ablations and sanity checks, remove obviously weak or duplicate descriptors to obtain a screened pool $\mathcal{F}_1$.

Step 2: SHAP value computation. Fit F_θ on $\mathcal{F}_1$ and compute SHAP values $\{\phi_{i,f}\}$ on D_{val} via `TreeExplainer`. We inspect beeswarm/bar plots to understand global effects.

Step 3: Global contribution analysis. Retain features that are both strong and consistent with theory. Concretely, keep f if

$$s_f \;\geq\; Q_{0.15}\big(\{s_f\}\big) \quad \text{and} \quad c_f := \eta_f\, \rho_f \;\geq\; 0.20,$$

and the Spearman correlation passes significance ($p < 0.05$). Here $Q_{0.15}$ is the 15th percentile of the empirical $\{s_f\}$.

Step 4: Iterative pruning with CV safeguard. From the retained set, perform backward elimination: at each round remove the weakest candidate (smallest s_f or negative c_f), retrain F_θ, and accept the removal only if the cross-validated error does not worsen beyond a fixed tolerance:

$$\Delta\mathrm{RMSE}_{\text{cv}} = \mathrm{RMSE}_{\text{cv}}^{\text{new}} - \mathrm{RMSE}_{\text{cv}}^{\text{old}} \;\leq\; \varepsilon, \qquad \varepsilon = 0.01$$

Optionally, use a robust tolerance tied to fold variability:

$$\varepsilon = \min\big(\mathrm{SE}(\Delta),\, 0.01\big), \quad \mathrm{SE}(\Delta) = \mathrm{sd}\big(\{\Delta_k\}_{k=1}^{K}\big)/\sqrt{K},$$

and additionally require no significant degradation via a paired test (t or Wilcoxon), $p \geq 0.05$. Stop when no feature can be dropped without violating the criterion.

Outcome. This SHAP-guided pipeline yields a compact, non-redundant meta-feature vector (14 static & 10 dynamic in **TUNEAHEAD**), balancing informativeness and stability while aligning with theoretical expectations.

A.4 PREDICTION MODEL TRAINING AND EVALUATION

Training and Evaluation Details. We split the 1,300+ fine-tuning experiments into 28% test, with the remaining 80% further divided into train/validation/calibration subsets (46/14/12%). Unless otherwise noted, the split uses a fixed random seed (=36). The predictor is a LightGBM gradient-boosted decision tree (GBDT) with the following hyperparameters fixed across all experiments: `learning_rate=0.05`, `num_leaves=4`, `n_estimators=140`, `subsample=0.6`, `colsample_bytree=0.6`, `min_child_samples=20`, and ℓ_2 regularization `lambda_l2=1.0`. We use early stopping with patience of 50 rounds. All models are trained with `n_jobs=-1` (multi-threading enabled).

For ablation studies, we define static-only features as dataset descriptors (e.g., length, lexical diversity, perplexity), and dynamic-only features as probe-derived signals (e.g., loss decay, gradient variance). The details of all the features can be looked up in Appendix B.

A.5 BASELINES

Ablation variants. **TUNEAHEAD**-Static-Only (only static features) and **TUNEAHEAD**-Dynamic-Only (only dynamic probe features).

Practical/literature-inspired baselines. (1) *Early-Stop Extrapolation*: linear extrapolation of the 100-step validation loss (Domhan et al., 2015; Adriaensen et al., 2023); (2) *Loss-Rate Features*: rates of loss decrease under different schedules (Luo et al., 2025); (3) *Reference Perplexity*: reference-PPL as dataset difficulty proxy (Gururangan et al., 2020; Harada et al., 2025); (4) *ProxyLM* (Anugraha et al., 2024): regress on proxy-model scores (e.g., SmolLM-135M/360M, BLOOMZ-560M) optionally with dataset features.

A.6 Evaluation Metrics and Protocols

We evaluate **TuneAhead** with four standard regression and tolerance-based metrics:

- **RMSE (percentage points)**: measures absolute error between predicted and ground-truth performance,
$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i,j}(P_{i,j} - R_{i,j})^2}.$$
- R^2: coefficient of determination, quantifying explained variance in ground-truth performance,
$$R^2 = 1 - \frac{\sum_{i,j}(R_{i,j}-P_{i,j})^2}{\sum_{i,j}(R_{i,j}-\bar{R_{i,j}})^2}.$$
- **Pearson** r: measures linear correlation between predictions and ground-truth,
$$r = \frac{\sum_{i,j}(P_{i,j}-\bar{P})(R_{i,j}-\bar{R})}{\sqrt{\sum_{i,j}(P_{i,j}-\bar{P})^2\sum_{i,j}(R_{i,j}-\bar{R})^2}}.$$
- **Acc@$k$pp**: tolerance-based accuracy, defined as the fraction of predictions within k percentage points of ground-truth,
$$\text{Acc@}k\text{pp} = \frac{1}{N}\sum_{i,j}\mathbf{1}\big(|P_{i,j} - R_{i,j}| \le k\big), \quad k \in \{1,2,3\}.$$

For all metrics, we average over three random seeds when obtaining ground-truth labels.

A.7 Per-Domain Breakdown of Prediction Performance

To complement the aggregate results in Table 1, we report per-domain breakdowns of prediction accuracy on MMLU. The 57 subjects of MMLU are grouped into seven finer categories (STEM, Social Sciences, Humanities, Arts & Culture, Health & Medicine, Business & Professional, and Other/General Knowledge). This analysis verifies whether **TuneAhead** consistently generalizes across heterogeneous domains.

Table A.1: Per-domain breakdown of prediction performance on MMLU. Domains are grouped into finer categories for clarity. Metrics include RMSE (pp), Pearson correlation r, and accuracy within $\pm$2pp tolerance.

Domain	RMSE ↓	r ↑	Acc@2pp ↑
STEM (Math, CS, Physics, Bio)	1.62	0.98	80.5
Social Sciences (Econ, Psych, Soc)	1.45	0.99	84.2
Humanities (History, Philosophy, Law)	1.53	0.99	83.1
Arts & Culture (Literature, Linguistics, Art)	1.59	0.98	81.9
Health & Medicine (Clinical, Nutrition, Public Health)	1.48	0.99	82.7
Business & Professional (Mgmt, Exams)	1.51	0.99	82.3
Other / General Knowledge	1.41	0.99	82.7
Overall	1.47	0.99	82.5

As shown in Table A.1, the predictive performance of **TuneAhead** is consistent across domains. RMSE varies only within $\pm$0.15 across groups, and Pearson correlations remain above 0.98 throughout. Accuracy within $\pm$2pp is also stable, ranging between 80–84%. This robustness indicates that the framework does not disproportionately benefit from or fail on particular subject categories, reinforcing its general applicability across diverse fine-tuning scenarios.

A.8 CALIBRATION ANALYSIS

In Section 5, we emphasized that **TUNEAHEAD**'s predictions are not only accurate in terms of correlation with fine-tuning outcomes, but also well-calibrated in absolute values. This property is crucial for practical use cases, because a well-calibrated predictor allows practitioners to directly interpret a predicted score as an approximate probability of fine-tuning success, enabling threshold-based go/no-go decisions without ad hoc post-processing.

Figure A.1 shows the calibration curves of **TUNEAHEAD** compared with representative baselines. Each point corresponds to a bin of predicted scores, with the x-axis showing the mean predicted value and the y-axis showing the empirical success rate within that bin. The dashed line represents perfect calibration. We observe that several baselines (e.g., *Reference-PPL*, *Early-Stop*) systematically deviate from the diagonal, indicating a tendency to either over-estimate or under-estimate success probability. By contrast, **TUNEAHEAD**'s curve (red circles) remains consistently close to the diagonal across the full range, demonstrating superior calibration. This confirms that **TUNEAHEAD** is not only a strong ranker (as shown by the high correlations in Table 1), but also a reliable probability estimator, making its scores directly actionable for practitioners.

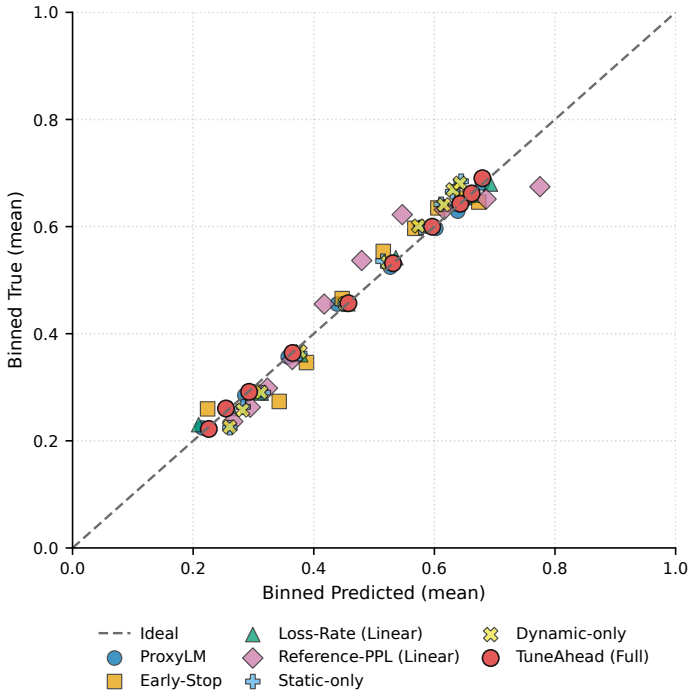


Figure A.1: Calibration plots of predicted scores across models. While baselines often deviate from the ideal diagonal, **TUNEAHEAD** (red) remains closely aligned with perfect calibration. This demonstrates that **TUNEAHEAD**'s predictions are both accurate in ranking and reliable in absolute probability estimation, complementing the aggregate results reported in Table 1.

A.9 CONFIDENCE INTERVALS OF MAIN RESULTS

In Section 5, Table 1 reported the aggregate RMSE and accuracy of **TUNEAHEAD** and representative baselines. While those results demonstrated clear performance gains, it is also important to examine the robustness of these findings under statistical resampling. To this end, we conducted bootstrap analysis (N=1000 resamples) and computed 95% confidence intervals for both RMSE and accuracy.

As shown in Figure A.2, **TUNEAHEAD** achieves the most stable performance: its RMSE is significantly lower than all baselines, and its accuracy is consistently higher. The narrow error bars demonstrate that these results are not due to random variation, but reflect a statistically robust advantage of combining static and dynamic signals.

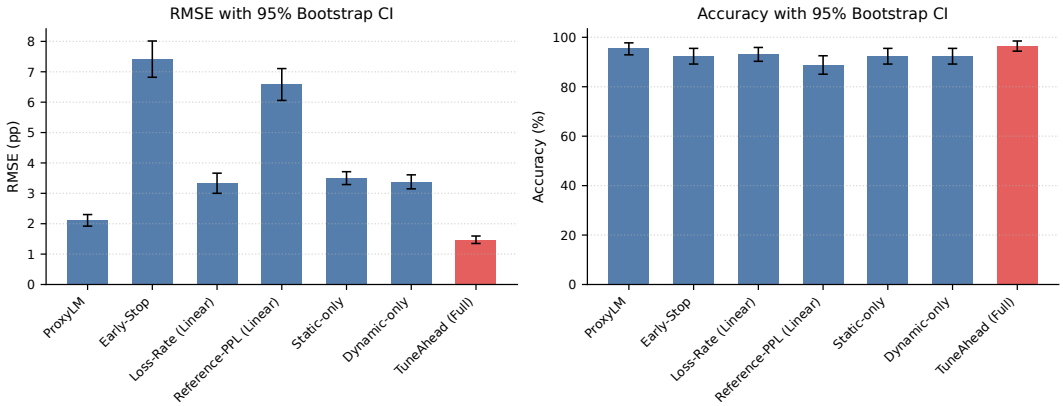


Figure A.2: 95% bootstrap confidence intervals for RMSE (left) and accuracy (right) across **TUNEAHEAD** and baseline predictors. Red bars highlight **TUNEAHEAD** (Full), while blue bars indicate baselines. The results confirm that **TUNEAHEAD** not only achieves the lowest RMSE but also the highest accuracy, with narrow confidence intervals that do not overlap with most baselines. This reinforces the statistical reliability of the improvements reported in Table 1.

A.10 PER-HYPERPARAMETER BREAKDOWN

To verify that the improvements of **TUNEAHEAD** are not tied to a narrow choice of optimization settings, we report test-set performance grouped by key hyperparameters as shown in the tables below.

Table A.2: Per-learning-rate performance of **TUNEAHEAD** on the test set.

Learning rate	RMSE ↓	R^2 ↑	r ↑	Accuracy (±kpp) ↑		
				k=1	k=2	k=3
2×10^{-4}	1.36	0.99	1.00	51.68	85.23	96.64
3×10^{-4}	1.61	0.99	1.00	47.90	78.99	93.28

Table A.3: Per-batch-size performance of **TUNEAHEAD** on the test set.

Batch size	RMSE ↓	R^2 ↑	r ↑	Accuracy (±kpp) ↑		
				k=1	k=2	k=3
8	1.58	0.99	1.00	51.96	76.47	92.16
16	1.47	0.99	1.00	45.56	84.44	96.67
32	1.32	0.99	1.00	52.63	88.16	97.37

Table A.4: Two-way breakdown by (learning rate × batch size).

Learning rate	Batch size	RMSE ↓	R^2 ↑	r ↑	Accuracy (±kpp) ↑		
					k=1	k=2	k=3
2×10^{-4}	8	1.37	0.99	1.00	56.16	82.19	95.89
2×10^{-4}	16	1.32	0.99	1.00	50.00	87.50	97.50
2×10^{-4}	32	1.39	0.99	1.00	44.44	88.89	97.22
3×10^{-4}	8	2.03	0.98	0.99	41.38	62.07	82.76
3×10^{-4}	16	1.59	0.99	1.00	42.00	82.00	96.00
3×10^{-4}	32	1.27	0.99	1.00	57.50	85.00	95.00

Overall, the per-hyperparameter breakdown reinforces the robustness of **TUNEAHEAD**. First, across different learning rates (Table A.2), the predictor achieves consistently low RMSE (1.3–1.6pp) and nearly perfect correlation ($r \approx 1.0$). The slight drop in Acc@2pp for 3×10^{-4} (79% vs. 85% at 2×10^{-4}) suggests that steeper learning rates may induce more variance in the fine-tuning trajectories, but the deviations remain small relative to the overall gains reported in Table 1.

Second, when grouping by batch size (Table A.3), we find that larger batches generally yield slightly lower RMSE and higher calibration accuracy. For instance, batch size 32 achieves Acc@2pp of 88%,

compared to 76% for batch size 8. This trend aligns with the intuition that smaller batches introduce more stochasticity, yet **TUNEAHEAD** is able to maintain strong predictive reliability across the range.

Finally, the two-way grid (Table A.4) confirms that performance remains strong across all (learning rate $\times$ batch size) combinations. Even in the most challenging regime (learning rate 3×10^{-4}, batch size 8), RMSE stays within 2pp and correlation remains high ($r = 0.99$). Conversely, more stable settings such as (2×10^{-4}, 16 or 32) yield near-ideal calibration with Acc@3pp above 97%. These results demonstrate that **TUNEAHEAD**'s improvements are not contingent on narrow hyperparameter choices, but generalize broadly across optimization regimes.

B META-FEATURE COMPENDIUM

This appendix provides a detailed compendium of all candidate static and dynamic meta-features engineered for the **TUNEAHEAD** framework. For each feature, we include its **definition**, **formula** (when applicable), **acquisition method**, **predictive hypothesis (signal)**, and relevant literature references.

B.0 NOTATION FOR META-FEATURE DEFINITIONS

For clarity, we summarize the symbols used in the meta-feature definitions in Appendix B. Unless otherwise stated, all quantities are defined for a single dataset–hyperparameter pair (D_i, H_j) and its associated probe run.

Global objects.

D_i	The i-th fine-tuning dataset (instruction–response pairs).
H_j	The j-th hyperparameter configuration (e.g., learning rate, batch size, LoRA rank).
M_0	The frozen base language model used for reference perplexity and the probe run.
θ_0	Model parameters of M_0 before the probe fine-tuning.
θ_T	Model parameters after T probe steps (end of the probe run).
θ^*	Final probe parameters; in practice we set $\theta^* = \theta_T$.
T	Number of probe optimization steps (typically $T = 100$).
N	Number of training examples in the current dataset D_i (re-used in several formulas).
$\|\cdot\|_2$	Euclidean (L2) norm of a vector.

Static feature notation (Appendix B.1).

μ_{len}	Mean token length of examples in D_i.
σ_{len}	Standard deviation of token lengths in D_i.
x	Input sequence (prompt) in an instruction–response pair (x, y).
y	Output sequence (response) in an instruction–response pair (x, y).
$E(s)$	Embedding vector of sample s in the dataset, computed by a fixed encoder.
μ	Mean embedding vector over all samples in D_i: $\mu = \frac{1}{N}\sum_s E(s)$.
σ	Standard deviation of embedding distances to the mean, i.e. the scalar standard deviation of $\{\|E(s) - \mu\|_2\}_s$.
EOR	Embedding Outlier Ratio, EOR $= \frac{1}{N}\big\|\{s : \|E(s) - \mu\|_2 > 3\sigma\}\big\|$.
D_i	(in $\mathrm{PPL}(D_i, M_0)$) Same dataset as above, used to compute reference perplexity.
x_n	The n-th training example (usually a concatenated input–output sequence) in D_i.
$p_{M_0}(x_n)$	Probability of sequence x_n under the base model M_0.
$\mathrm{PPL}(D_i, M_0)$	Reference perplexity of dataset D_i on M_0: $\mathrm{PPL}(D_i, M_0) = \exp\big(-\frac{1}{N}\sum_{n=1}^{N}\log p_{M_0}(x_n)\big)$.

Loss–dynamics notation (Appendix B.2.1).

L_t	Training loss at probe step t ($t = 1, \ldots, T$).
L_0	Initial loss at the very beginning of the probe ($t = 0$), used as Initial Loss.
$\bar{L}$	Mean loss over the probe trajectory: $\bar{L} = \frac{1}{T}\sum_{t=1}^{T} L_t$.
ℓ_t	Log-loss at step t, defined as $\ell_t = \log L_t$.
a, b	Intercept and slope of the linear fit $\ell_t \approx a + bt$ obtained by Ordinary Least Squares; the slope b is used as the Loss Decay (more negative b means faster adaptation).
σ_L^2	Loss Stability, i.e. variance of the loss along the probe: $\sigma_L^2 = \frac{1}{T}\sum_{t=1}^{T}(L_t - \bar{L})^2$.

Gradient-based notation (Appendix B.2.2).

g_t	Gradient of the training loss with respect to the model parameters at probe step t.
$\|g_t\|_2$	L2 norm of the gradient at step t.
μ_g	Mean gradient norm across the probe: $\mu_g = \frac{1}{T}\sum_{t=1}^{T}\|g_t\|_2$.
σ_g^2	Variance of gradient norms: $\sigma_g^2 = \frac{1}{T}\sum_{t=1}^{T}(\|g_t\|_2 - \mu_g)^2$.
c_t	Gradient Consistency at step t, defined as cosine similarity between consecutive gradients: $c_t = \dfrac{g_t \cdot g_{t+1}}{\|g_t\|_2\|g_{t+1}\|_2}$.
g_k	k-th coordinate of a gradient vector g.
ϵ	Small positive threshold used to decide whether a coordinate is "effectively zero".
s	Gradient Sparsity: fraction of near-zero gradient coordinates, $s = \dfrac{\|\{k : \|g_k\| < \epsilon\}\|}{\|g\|}$, where $\|g\|$ is the number of coordinates in g.

Model-based notation (Appendix B.2.3).

$\Delta\theta$	Parameter Change Norm during the probe: $\Delta\theta = \|\theta_T - \theta_0\|_2$.
$L(\theta)$	Training loss evaluated at parameters θ.
δ	Small random parameter perturbation used to probe the loss landscape around θ^*.
ΔL	Loss Landscape Flatness proxy: $\Delta L = L(\theta^* + \delta) - L(\theta^*)$.
$P_{\textbf{baseline}}$	Performance of the baseline (pre-probe) model on an out-of-domain evaluation task, used in the catastrophic forgetting proxy.
$P_{\textbf{probe}}$	Performance of the probed / partially fine-tuned model on the same out-of-domain task.
ΔP	Catastrophic Forgetting Proxy: $\Delta P = P_{\text{baseline}} - P_{\text{probe}}$.
h	Activation vector in a hidden layer for a given input.
$\|h\|$	Number of units (coordinates) in the activation vector h.
s_a	Activation Sparsity: fraction of near-zero activations, $s_a = \dfrac{\|\{u : \|h_u\| < \epsilon\}\|}{\|h\|}$, where h_u is the u-th coordinate of h and ϵ is the same small threshold as above.

B.1 Static Meta-Features

Static features are computed from the dataset D_i prior to training, sometimes using the frozen base model $\mathcal{M}_0$.

B.1.1 Global statistics

- **Token Lengths (Mean & Std Dev).**

$$\mu_{\text{len}}(D_i) = \frac{1}{N}\sum_{j=1}^{N}|x_j|, \qquad \sigma_{\text{len}}(D_i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\big(|x_j| - \mu_{\text{len}}(D_i)\big)^2}.$$

Acquisition: Compute from input/output token counts. **Signal:**Beyond surface complexity, token lengths also reflect task formulation style. Datasets with extremely short inputs often lack linguistic structure and yield unstable gradients, while very long instructions introduce compositional reasoning that can reveal early adaptation bottlenecks. We observe that datasets with high token-length variance tend to produce noisy probe loss curves, indicating that this feature reliably distinguishes inconsistent data sources or uneven annotation quality. (Vettoruzzo et al., 2023; Moghe et al., 2024).

- **Input–Output Length Ratio.**

$$r_{\text{io}}(D_i) = \frac{1}{N}\sum_{j=1}^{N}\frac{|y_j|}{|x_j|}.$$

Acquisition: Average of output-to-input length ratios. **Signal:** This ratio correlates with task paradigms: low ratios characterize compressive tasks such as summarization, while high ratios reflect elaborative tasks like question answering or code generation. Such structural differences shape model adaptation speed and generalization (Lin, 2004; Narayan et al., 2018).This ratio also serves as a proxy for semantic compression vs. semantic expansion. Large ratios typically indicate generative tasks where hallucination risks or stylistic variability appear, which increases gradient variance and lowers gradient consistency. The ratio is scale-free, inexpensive, and complements perplexity in signaling the "reasoning burden" required by the dataset.

- **Special Character & Code Ratio.**
Let $\mathbb{1}_{\text{sc}}(t)$ be 1 if token t is a special or code token (from a fixed list) and 0 otherwise. Let $\text{Tok}(D_i)$ be all tokens in D_i. Then

$$r_{\text{sc}}(D_i) = \frac{\sum_{t\in\text{Tok}(D_i)}\mathbb{1}_{\text{sc}}(t)}{|\text{Tok}(D_i)|}.$$

Acquisition: Count of special/code tokens vs. total. **Signal:** Elevated ratios typically indicate domain-specific corpora (e.g., programming or formula-heavy data) or unclean web text. Such distributions require tailored tokenization or model adaptation, as shown in large-scale text-to-text transfer studies (Allamanis et al., 2018; Raffel et al., 2020). This feature further helps differentiate "format-structured" domains—HTML, JSON, LaTeX, or code—from natural language. These domains often interact poorly with models pretrained on mixed corpora, producing out-of-distribution token transitions visible in perplexity and loss-decay trends. High ratios also correlate with increased difficulty in the early probe because special tokens amplify embedding outlier probability.

- **Approximate Duplicates Ratio.**
For each example j, let z_j be its concatenated text (e.g., $z_j = x_j + y_j$). Let $\text{Jacc}(z_j, z_k)$ be the Jaccard similarity between the token sets of z_j and z_k, and fix a threshold $\tau \in (0, 1)$ (e.g., $\tau = 0.9$). The (conceptual) duplicate ratio is

$$\text{Dup}(D_i) = \frac{2}{N(N-1)}\sum_{1\le j<k\le N}\mathbb{1}\big[\text{Jacc}(z_j, z_k) \ge \tau\big],$$

which we approximate in practice using MinHash/LSH instead of exhaustive pairwise comparisons.
Acquisition: Identify near-duplicates via embedding similarity threshold τ. **Signal:** High duplication reduces effective dataset diversity, amplifies memorization, and weakens generalization. Deduplication in LLM pretraining has been shown to improve downstream performance and reduce overfitting (Carlini et al., 2023; Lee et al., 2022). Duplicate-heavy datasets artificially inflate dataset size while reducing effective information density. Higher duplicate ratios are strongly associated with shallow loss decay and low parameter movement, because the probe repeatedly sees nearly identical gradients. MinHash allows efficient detection of semantic near-copies and avoids quadratic pairwise operations.

- **Embedding Outlier Ratio.**
Let s_j denote the text for example j (e.g., input or input+output), and $e_j = E(s_j) \in \mathbb{R}^d$ its embedding. Define the embedding mean and (scalar) standard deviation as

$$\mu_E = \frac{1}{N}\sum_{j=1}^{N} e_j, \qquad \sigma_E = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\big\|e_j - \mu_E\big\|_2^2}.$$

The outlier ratio is

$$\text{EOR}(D_i) = \frac{1}{N}\sum_{j=1}^{N}\mathbb{1}\big[\|e_j - \mu_E\|_2 > 3\sigma_E\big].$$

Acquisition: Detect large deviations in embedding space. **Signal:** Outlier samples often correspond to mislabeled, noisy, or domain-shifted data. Their presence destabilizes optimization and can severely degrade model robustness. Removing outliers is a key step in modern dataset curation pipelines (Hendrycks et al., 2019; Northcutt et al., 2022; Dodge et al., 2021). EOR highlights whether the dataset contains idiosyncratic or adversarial examples that do not align with the dominant semantic manifold. Such outliers typically cause spikes in loss stability and suppress gradient consistency during the probe. The three-sigma threshold is a robust and interpretable choice: it captures semantic anomalies without being overly sensitive to moderate tail imbalance.

- **Dataset Size (Num Items)**

$$\mathrm{Size}(D_i) = N,$$

i.e., the number of examples in D_i.

Acquisition: Dataset example count. **Signal:** Larger datasets typically improve model performance, but the gains follow a power-law with diminishing returns. Scaling law analyses show that optimal performance requires balancing dataset size with model capacity and compute budget(Kaplan et al., 2020; Hoffmann et al., 2022).

B.1.2 Lexical diversity features

- **Type–Token Ratio.**

Let $T(D_i) = |\mathrm{Tok}(D_i)|$ be the total number of tokens and $|V(D_i)|$ the vocabulary size. Then

$$\mathrm{TTR}(D_i) = \frac{|V(D_i)|}{T(D_i)}.$$

Acquisition: Compute vocabulary diversity in D_i. **Signal:** A low TTR indicates repetitive content and limited lexical variety, which can impair a model's ability to generalize to unseen expressions. High TTR reflects lexical richness but may also introduce noise or rare tokens. TTR is a long-established measure of lexical richness and a standard meta-feature in dataset characterization (Rivolli et al., 2019).

- **N-gram Repetition.**

Fix an n (e.g., $n = 4$). Let $\mathcal{G}_n(D_i)$ be the multiset of all n-grams in D_i, and let $c(g)$ be the count of g in $\mathcal{G}_n(D_i)$. We define the repetition ratio as

$$\mathrm{Rep}_n(D_i) = \frac{\sum_g \max\{c(g) - 1,\, 0\}}{\sum_g c(g)}.$$

Acquisition: Fraction of repeated n-grams in outputs. **Signal:** High repetition rates often signal low-quality or synthetic outputs, reducing effective informational content and promoting degenerate training behavior. Low repetition may indicate dispersed data but could also reduce coherence. Repetition metrics are widely used in text degeneration and quality control studies(Rivolli et al., 2019; Holtzman et al., 2020).

- **Instruction Complexity.**

Let $\mathrm{depth}(x_j)$ denote the syntactic parse-tree depth (or another fixed complexity score) of input x_j. Then

$$\mathrm{IC}(D_i) = \frac{1}{N} \sum_{j=1}^{N} \mathrm{depth}(x_j).$$

Acquisition: Average parse-tree depth of instructions. **Signal:** Shallow trees suggest trivial instructions that under-challenge the model, while overly deep trees reflect high syntactic complexity that may hinder comprehension or stable learning. Balanced complexity encourages both learnability and generalization (Yatskar, 2019)

Together, lexical richness and repetition patterns help differentiate high-entropy datasets from templated or pattern-locked corpora. Higher lexical diversity often correlates with smoother probe adaptation, whereas heavy n-gram repetition is a known predictor of overfitting in short-horizon training. Instruction complexity, approximated via parse-tree depth, is also predictive of whether the model must engage multi-step reasoning, which tends to manifest as slower yet more stable loss decay.

B.1.3 INFORMATION-THEORETIC PROPERTIES

- **Reference Perplexity.**
  Let M_0 be a frozen base model and x_n the n-th token in a concatenated sequence from D_i of length N_{tok}. The reference cross-entropy is

$$\mathrm{CE}(D_i, M_0) = -\frac{1}{N_{\text{tok}}} \sum_{n=1}^{N_{\text{tok}}} \log p_{M_0}(x_n \mid x_{<n}),$$

  and the reference perplexity is

$$\mathrm{PPL}(D_i, M_0) = \exp\big(\mathrm{CE}(D_i, M_0)\big).$$

  We use both the mean and standard deviation of token-level losses as features.

  Acquisition: Measure using frozen base model. **Signal:** High perplexity indicates distributional mismatch between dataset and pretraining corpus, leading to slower convergence and increased adaptation cost. Low perplexity suggests greater alignment with prior knowledge. PPL remains a widely accepted proxy for domain mismatch and learning difficulty(Wu et al., 2017; Jozefowicz et al., 2016). Beyond measuring difficulty, reference perplexity acts as a proxy for model surprise under zero adaptation. Datasets with high mean perplexity typically exhibit steeper early improvements (more negative loss-decay slopes), but at the cost of higher volatility due to stronger domain mismatch. Considering both the mean and the standard deviation of perplexity therefore captures not only overall shift but also internal heterogeneity within the dataset.

- **Input–Output Semantic Similarity (IO Similarity)**
  Let $e_j^x = E(x_j)$ and $e_j^y = E(y_j)$ be embeddings of input and output. We define

$$\text{IO-Sim}(D_i) = \frac{1}{N} \sum_{j=1}^{N} \cos\big(e_j^x, e_j^y\big).$$

  Acquisition: Cosine similarity of embeddings. **Signal:** Low similarity may reflect irrelevant or hallucinated outputs, while very high similarity often indicates trivial paraphrasing lacking informativeness. Moderate levels of similarity are most effective for meaningful adaptation. Embedding-based similarity has been widely studied in representation learning (Clark et al., 2020; Cer et al., 2018).

- **Output Semantic Diversity.**
  Let $e_j^y = E(y_j)$ and define similarity $\mathrm{Sim}(y_j, y_k) = \cos(e_j^y, e_k^y)$. Then

$$\mathrm{Div}(D_i) = \frac{2}{N(N-1)} \sum_{1 \le j < k \le N} \big(1 - \mathrm{Sim}(y_j, y_k)\big).$$

  Acquisition: Average pairwise output dissimilarity. **Signal:** Low diversity signals redundancy and narrow coverage, while excessive diversity may indicate incoherence or noisy task signals. Balanced semantic diversity provides both robustness and coverage, promoting better generalization (Ziegler et al., 2020; Li et al., 2016).

  These semantic features measure representational compression of the task. High input–output similarity implies strongly input-grounded tasks, while low similarity or high output semantic diversity suggests creative or open-ended generation. In practice, they explain failure modes where probe loss decay appears strong, but final task performance is poor due to hallucination or uncontrolled style drift.

- **LM-Data Vocabulary Alignment (KL Divergence).**
  Let V be a shared vocabulary. Let $P(w)$ be the smoothed unigram frequency of token w in D_i and $Q(w)$ the corresponding frequency in a reference pretraining corpus. Then

$$\mathrm{KL}(P \,\|\, Q) = \sum_{w \in V} P(w) \log \frac{P(w)}{Q(w)}.$$

  Acquisition: Compare dataset and reference corpus word frequencies. **Signal:** Large divergence highlights domain shift, suggesting that the dataset vocabulary departs from the

pretraining distribution. This increases the adaptation burden and may reduce efficiency. KL divergence is a standard measure in domain adaptation and data selection (Aharoni & Goldberg, 2020; Axelrod et al., 2011).

- **Answer Groundedness.**
  For each example (x_j, y_j), let $\mathrm{ngrams}(y_j)$ and $\mathrm{ngrams}(x_j)$ be the multisets of n-grams (for a fixed n). Define per-example groundedness

$$g(y_j, x_j) = \frac{|\mathrm{ngrams}(y_j) \cap \mathrm{ngrams}(x_j)|}{|\mathrm{ngrams}(y_j)|},$$

  and dataset-level groundedness

$$G(D_i) = \frac{1}{N} \sum_{j=1}^{N} g(y_j, x_j).$$

  Acquisition: Ratio of overlapping n-grams between output and input. **Signal:** Low groundedness suggests hallucination or irrelevant generation, while overly high groundedness may reduce informativeness by copying excessively. Moderate grounding balances fidelity with informativeness, ensuring both reliability and novelty (Ji et al., 2023; Zhao et al., 2020).

  Vocabulary KL focuses on lexical mismatch that pure perplexity may blur, for example rare-domain terminology or shifting tokenization patterns. Answer groundedness, in contrast, quantifies how much the dataset encourages copying versus abstraction: very low groundedness correlates with hallucination-prone gradients, whereas excessively high groundedness correlates with shallow parameter updates that under-explore model capacity. These two features thus capture complementary aspects of domain shift and supervision style.

B.2 Dynamic Probe Meta-Features

Dynamic features are extracted during a standardized 100-step probe run.

B.2.1 Loss-based indicators

- **Initial Loss.**

$$L_0(D_i, H_j) = L(\theta_0; D_i),$$

  i.e., the probe loss evaluated at parameters θ_0 before any updates (on a fixed probe batch or mini-epoch).

  Acquisition: Probe loss at the first optimization step. **Signal:** The initial loss measures how well the pretrained model aligns with the dataset before adaptation. High values suggest a significant domain gap, requiring more updates to adapt, while low values indicate better alignment and easier fine-tuning. It is widely used as a proxy for domain difficulty in transfer learning (Arpit et al., 2017; Hestness et al., 2017). Initial loss L_0 is particularly useful for detecting coarse domain mismatch. Datasets with extremely high L_0 often contain formatting inconsistencies or noisy annotations, which subsequently manifest as low gradient consistency and elevated catastrophic forgetting during the probe. Conversely, very low L_0 suggests that the base model is already well aligned with the task distribution.

- **Loss Decay Rate.**
  For step indices $t = 1, \ldots, T$, define log-loss $\ell_t = \log L_t$. We fit an Ordinary Least Squares (OLS) linear regression

$$\ell_t \approx a + bt$$

  by solving

$$(a^\star, b^\star) = \arg\min_{a,b} \sum_{t=1}^{T} (\ell_t - a - bt)^2.$$

  The Loss Decay feature is the slope

$$\alpha(D_i, H_j) = b^\star.$$

  More negative α indicates faster exponential decay of loss in the early training regime.

Acquisition: Slope of regression fit on probe loss curve. LinReg means linear rgression with Ordinary Least Squares (OLS). Slope is **Signal:** A steep negative slope indicates strong learnability and rapid adaptation, whereas flat or unstable curves suggest noisy or hard-to-learn data. Early loss decay is highly predictive of final model performance (Wu et al., 2017; Hestness et al., 2017; Loog & Viering, 2022). The slope on the log-loss curve is deliberately chosen for its scale invariance and robustness to multiplicative noise. Small but consistently negative slopes indicate steady learning, while oscillatory or near-zero slopes typically correspond to datasets with noisy semantics or highly heterogeneous instruction formats. This formulation is also consistent with NTK-inspired (Jacot et al., 2020)linearization analyses, which predict approximately linear learning dynamics early in training.

- **Loss Curve Stability**

  Let

$$\bar{L} = \frac{1}{T}\sum_{t=1}^{T} L_t$$

  be the mean probe loss. The (population) variance is

$$\sigma_L^2(D_i, H_j) = \frac{1}{T}\sum_{t=1}^{T}(L_t - \bar{L})^2.$$

  Acquisition: Variance of loss during probe. **Signal:** Low variance implies stable optimization and smoother convergence, while high fluctuations often reflect noisy data, poor learning rates, or unstable alignment between model and task. Stable trajectories are associated with better generalization(Li et al., 2024; Wu et al., 2017). Loss volatility is a highly diagnostic signal for annotation noise, domain inconsistency, and unstable token distributions. Datasets with large σ_L^2 frequently show poor end-of-probe generalization and weaker gradient norms, indicating that the model is reacting to contradictory supervision. In contrast, low-volatility loss curves tend to correspond to well-curated dataset where the probe dynamics are smooth and predictable.

B.2.2 Gradient-based indicators

- **Gradient Norm (Mean & Variance)**

  Let $g_t = \nabla_\theta L_t$ be the gradient at step t and $r_t = \|g_t\|_2$ its Euclidean norm. Define

$$\mu_g(D_i, H_j) = \frac{1}{T}\sum_{t=1}^{T} r_t, \qquad \sigma_g^2(D_i, H_j) = \frac{1}{T}\sum_{t=1}^{T}(r_t - \mu_g(D_i, H_j))^2.$$

  Acquisition: Gradient norms across probe steps. **Signal:** Gradient norms reflect the strength of learning signals. Very large norms can cause instability or gradient explosion, while very small norms may stall learning or trap the model in poor local minima. Both mean and variance provide insight into learning dynamics (Pascanu et al., 2013; Sutskever et al., 2013; Killamsetty et al., 2021). The joint behavior of the mean and variance of $\|g_t\|_2$ encodes whether the dataset induces smooth or sharply fluctuating learning trajectories. Large variance is a hallmark of training instability and correlates with sharp minima detected by the flatness proxy, whereas consistently tiny norms suggest that the data provides little informative signal beyond the pretrained state. These statistics are cheap to obtain via hooks and yet strongly predictive of downstream calibration quality.

- **Gradient Consistency.**

  For $t = 1, \ldots, T-1$, define cosine similarity between consecutive gradients

$$c_t = \cos(g_t, g_{t+1}) = \frac{\langle g_t, g_{t+1}\rangle}{\|g_t\|_2\,\|g_{t+1}\|_2}.$$

  The consistency score is

$$\mathrm{GC}(D_i, H_j) = \frac{1}{T-1}\sum_{t=1}^{T-1} c_t.$$

Acquisition: Cosine similarity between sequential gradients. **Signal:** High consistency indicates coherent optimization paths, suggesting the model is learning a stable objective. Low or negative alignment suggests noisy or conflicting signals, slowing convergence. Gradient alignment is also linked to meta-learning generalization (Finn et al., 2017; Killamsetty et al., 2021; Guiroy et al., 2019). Gradient alignment offers a functional measure of how "well-posed" the supervision is. Datasets with conflicting signals—for example, heterogeneous styles or contradictory labels—lead to low cosine similarity between consecutive gradients and undermine the effectiveness of even strong learning rates. High gradient consistency, in contrast, indicates that updates point in broadly similar directions, and empirically correlates with both faster convergence and higher final accuracy.

- **Gradient Sparsity.**
  Fix a small threshold $\epsilon > 0$ (*e.g.,* $\epsilon = 10^{-6}$). Let $g_t \in \mathbb{R}^P$ and denote its coordinates by $g_t^{(k)}$. Define the fraction of near-zero coordinates at step t as

$$s_t = \frac{1}{P}\big|\{k \in \{1, \dots, P\} : |g_t^{(k)}| < \epsilon\}\big|.$$

  The gradient sparsity feature is

$$\mathrm{GS}(D_i, H_j) = \frac{1}{T}\sum_{t=1}^{T} s_t.$$

  Acquisition: Proportion of near-zero gradient components. **Signal:** High sparsity indicates that only a small subset of parameters is being updated, potentially limiting adaptation. Moderate sparsity may improve efficiency and generalization, but excessive sparsity may signal model–data mismatch (Frankle & Carbin, 2019; Evci et al., 2021; Killamsetty et al., 2021).Gradient sparsity measures how much of the parameter space is actively updated by the dataset. Highly sparse gradients indicate that only a small submanifold of parameters is being engaged, which is typical for overly templated or semantically shallow datasets. Probe runs with excessively sparse gradients almost always show small parameter movement and weaker downstream gains, highlighting under-utilization of model capacity.

B.2.3 Model-based indicators

- **Parameter Change Norm.**

$$\Delta\theta(D_i, H_j) = \|\theta_T - \theta_0\|_2,$$

  where θ_0 and θ_T are the parameter vectors before and after the T-step probe.
  Acquisition: L2 norm of parameter changes during probing. **Signal:** Parameter change magnitude reflects adaptation strength. Moderate changes indicate healthy learning, while excessive shifts may reflect instability or overfitting. This feature has been used to analyze learning dynamics and compression strategies (Li et al., 2020; Raghu et al., 2017).This feature captures the effective update magnitude independently of the absolute loss scale. Under small learning-rate regimes, $\Delta\theta$ closely correlates with the local curvature of the loss landscape: extremely small values indicate underfitting or poor gradient informativeness, while excessively large values correlate with overshooting or movement into brittle minima. Moderate parameter change typically aligns with healthy, task-aligned adaptation.

- **Loss Landscape Flatness Proxy.**
  Let $\theta^* = \theta_T$ denote the probe endpoint. Draw K i.i.d. perturbations $\delta^{(k)} \sim \mathcal{N}(0, \sigma^2 I)$ with a fixed $\sigma > 0$, and define

$$\Delta L^{(k)} = L(\theta^* + \delta^{(k)}; D_i) - L(\theta^*; D_i), \qquad k = 1, \dots, K.$$

  The flatness proxy is the average loss increase

$$\Delta L(D_i, H_j) = \frac{1}{K}\sum_{k=1}^{K} \Delta L^{(k)}.$$

  Smaller ΔL indicates a flatter local basin.

Acquisition: Apply perturbation δ and measure loss change. **Signal:** Flat minima (small ΔL) correspond to more robust solutions with better generalization under distribution shifts, while sharp minima indicate overfitting and fragility. Flatness has been consistently linked to generalization performance(Wu et al., 2017; Li et al., 2024).We adopt perturbation-based flatness because Hessian-trace estimators are unstable and prohibitively expensive for large language models. The resulting ΔL reliably separates datasets that generalize well from those that heavily overfit to the short-horizon probe data. Empirically, lower ΔL values coincide with smoother loss curves and higher tolerance-based accuracy (e.g., Acc@2pp), supporting the classic link between flat minima and robust generalization.

- **Catastrophic Forgetting Proxy.**
  Let T_{ood} be a fixed out-of-domain evaluation task and $P(\theta; T_{\text{ood}})$ be its performance metric (e.g., accuracy). Define

$$P_{\text{baseline}} = P(\theta_0; T_{\text{ood}}), \qquad P_{\text{probe}} = P(\theta_T; T_{\text{ood}}).$$

  The forgetting proxy is

$$\Delta P(D_i, H_j) = P_{\text{baseline}} - P_{\text{probe}}.$$

  Larger ΔP indicates stronger interference with previous knowledge.

  Acquisition: Performance drop on out-of-domain task during probe. **Signal:** Large drops indicate interference between new and old tasks, a hallmark of catastrophic forgetting. This proxy highlights whether fine-tuning data compromises existing knowledge(Wen & Itti, 2018). Although the probe uses only a small number of update steps, early forgetting is surprisingly predictive of final over-specialization or domain drift. Large ΔP values indicate that gradients induced by the new dataset interfere strongly with pretrained capabilities on unrelated tasks. This proxy therefore highlights data regimes where fine-tuning is likely to compromise core model knowledge, even if training loss appears to improve.

- **Activation Sparsity.**
  Consider a fixed hidden layer with activations $h_t \in \mathbb{R}^Q$ for a probe minibatch at step t (flattening over tokens and units). Fix a threshold $\epsilon > 0$ and define

$$s_t^{\text{act}} = \frac{1}{Q}\big|\{q \in \{1, \ldots, Q\} : |h_t^{(q)}| < \epsilon\}\big|.$$

  The activation sparsity feature is

$$s_a(D_i, H_j) = \frac{1}{T}\sum_{t=1}^{T} s_t^{\text{act}}.$$

  Acquisition: Fraction of near-zero activations in hidden layers. **Signal:** Sparse activations suggest selective use of model capacity. Moderate sparsity improves interpretability and efficiency, while excessive sparsity indicates underutilized capacity or inefficient learning. It is widely used as a proxy for model resource utilization (Glorot et al., 2011; Maass, 1997).Activation sparsity approximates how widely the dataset engages the internal representation space of the model. Datasets with richer and more varied semantics tend to activate broader regions of the network, whereas narrow or highly templated datasets trigger sparse activations concentrated in predictable subspaces. This feature complements gradient sparsity by revealing a second level of representational bottleneck, and helps explain why some datasets yield limited transfer despite low training loss.

C META-FEATURE COMPUTATION COST

ID	Feature (Appendix)	Category	Computation (brief)	GPU	Typical cost	Notes
S1	Token Lengths (Mean & Std)	B.1.1	token length stats	No	0.1–0.2 min	reuse tokenization
S2	Input–Output Length Ratio	B.1.1	avg $\lvert y\rvert/\lvert x\rvert$	No	< 0.1 min	aggregation
S3	Special Char & Code Ratio	B.1.1	special/code tokens ratio	No	0.1–0.2 min	regex/list
S4	Approx. Duplicates Ratio	B.1.1	MinHash/LSH near-dup	No	0.3–0.6 min	subsampling
S5	Embedding Outlier Ratio	B.1.1	3σ outliers in embed space	Opt.	0.2–0.4 min	small-batch embed
S6	Dataset Size (Num Items)	B.1.1	count N	No	< 0.1 min	—
S7	Type–Token Ratio (TTR)	B.1.2	unique/total tokens	No	0.1–0.2 min	streaming
S8	N-Gram Repetition Rate	B.1.2	repeated n-grams in y	No	0.2–0.3 min	$n = 2/3$
S9	Instruction Complexity	B.1.2	parse-tree depth avg	No	0.2–0.4 min	light parser
S10	Reference Perplexity (PPL)	B.1.3	frozen M_0 forward PPL	Yes	0.7–1.5 min	fp16/parallel
S11	IO Semantic Similarity	B.1.3	$\cos(E(x), E(y))$	Opt.	0.2–0.4 min	sampled pairs
S12	Output Semantic Diversity	B.1.3	mean pairwise $1 - \cos$ in y	Opt.	0.2–0.4 min	subsample pairs
S13	LM–Data Vocab KL	B.1.3	histogram KL(P‖Q)	No	0.1–0.2 min	vocab freq
S14	Answer Groundedness	B.1.3	n-gram overlap ratio	No	0.2–0.3 min	reuse counts
D1	Initial Loss L_0	B.2.1	step-1 loss (logs)	No	≈ 0	log-only
D2	Loss Decay Rate α	B.2.1	robust linreg on (t, L_t) or $\log L_t$	No	≈ 0	log-only
D3	Loss Curve Stability σ_L^2	B.2.1	variance/EMA jitter	No	≈ 0	log-only
D4	Grad Norm (Mean & Var)	B.2.2	mean/var of $\lVert g_t\rVert$	No	≈ 0	hook
D5	Gradient Consistency	B.2.2	$\cos(g_t, g_{t+1})$	No	≈ 0	hook
D6	Gradient Sparsity	B.2.2	near-zero grad ratio	No	≈ 0	hook
D7	Param Change Norm $\lVert\Delta\theta\rVert$	B.2.3	$\lVert\theta_T - \theta_0\rVert_2$	No	0.1–0.2 min	snapshots
D8	Flatness Proxy ΔL	B.2.3	loss diff under small perturbation	Yes	0.8–1.4 min	small eval
D9	Catastrophic Forgetting ΔP	B.2.3	OOD small-set performance drop	Yes	1.1–2.2 min	small eval
D10	Activation Sparsity	B.2.3	near-zero activations ratio	Yes	0.6–1.0 min	fwd hooks

Table C.1: Feature Computation Cost (2.5k–5k samples).

Table C.1 reports the per-feature overhead for a typical dataset of 2.5k–5k examples. In this regime, static extraction completes in ≈1.6–3.2 minutes and dynamic, model-based proxies add ≈2.4–4.8 minutes, yielding a total of ≈4–8 minutes per dataset (excluding the probe run itself). Most static signals are computed in a single streaming pass over tokenized text and scale essentially linearly with corpus size; approximate-duplicate detection uses MinHash/LSH sketches to avoid quadratic pairwise comparisons; and semantic-diversity statistics rely on subsampling so that the effective complexity is closer to $O(N \log N)$ than $O(N^2)$ in practice. Model-based statics such as reference perplexity amortize well on a single GPU with fp16 inference, while the dynamic features are harvested "for free" from the 100-step probe via loss logs and forward/gradient hooks, so their marginal cost is negligible compared to the probe itself. Two design choices keep the end-to-end budget below 5% of a full fine-tune: static features are a one-time, offline cost that is cached at the dataset level, and dynamic features piggy-back on traces already produced during the probe.

In engineering the pipeline we found that a small number of implementation details materially reduce wall-clock time without degrading predictive quality. Tokenization, length histograms, n-gram counts, and vocabulary frequencies are computed once and reused across features such as TTR, repetition, KL divergence, and groundedness, eliminating redundant passes. For pairwise-style measures (near-duplicate ratio, output semantic diversity), we use sampling and sketching rather than exhaustive comparisons; within our cross-validation noise, the downstream predictor's RMSE and rank correlation remain unchanged. Micro-batch scheduling and fixed-length padding improve GPU utilization for the few GPU-bound features (reference PPL, activation sparsity, flatness/forgetting probes), typically reducing their segment of Table C.1 by 20–35% on commodity hardware.

Although we adopt a 100-step probe by default, the feature pipeline supports budget-aware operation. A practical policy is to run a static-only predictor first and trigger a short probe only when the predicted variance or decision margin falls below a user threshold. Empirically, we observe a clear elbow between 50 and 100 steps: moving from 50 to 75 steps brings a substantial gain in ACC@2pp, while 75 to 100 steps adds a smaller, but still meaningful, improvement for a modest increase in time; beyond 100 steps, returns are negligible. In settings where probe time is at a premium, a 75-step fallback retains roughly ninety percent of the accuracy of the 100-step configuration while further lowering cost.

We also note two practical failure modes. When datasets are extremely short or heavily templated, static signals can appear deceptively clean while the model underfits semantics; in such cases, even a short probe is advisable because loss stability and gradient alignment expose the hidden mismatch. Conversely, under strong domain shift with unusual tokenization (e.g., code-dominant text or formula-heavy inputs), perplexity and special-token ratios may inflate; the hybrid feature set remains predictive in our experiments, but we recommend enabling stricter outlier detection and deduplication thresholds and verifying that code-token ratios are recorded. To support reproducibility, we ship deterministic tokenization and counting, seeded MinHash/LSH for duplicates, fixed embedding back-ends for semantic statistics, and a single probe runner that exports all dynamic traces. With these settings, the per-dataset costs in Table C.1 reproduce within $\pm$10–15% across machines, and held-out predictive metrics vary within typical cross-validation noise. Overall, the feature suite is informative yet lightweight: static extraction is cached and near-linear, dynamic signals come at near-zero marginal cost, and the combined budget remains comfortably under five percent of a full run while delivering the predictive gains that enable early triage.

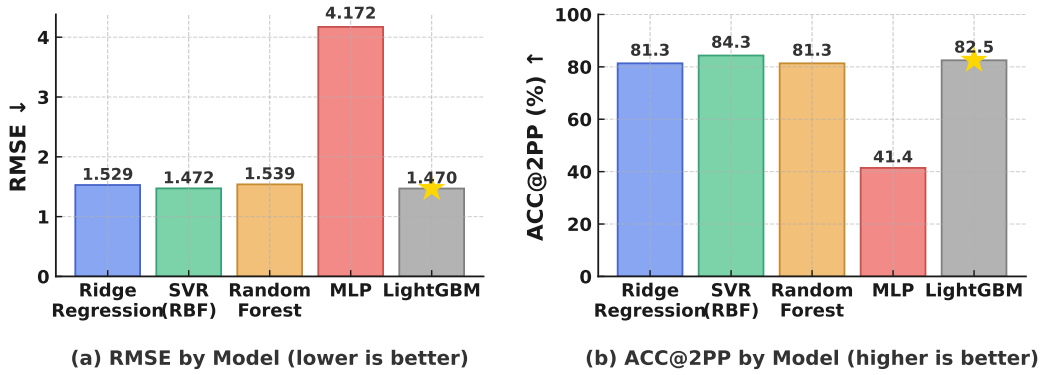


Figure D.1: Performance comparison of predictor models. We evaluate five models on the same meta-feature set. The plots show (a) RMSE (lower is better) and (b) Acc@2pp (higher is better). The results validate our choice of LightGBM (marked by a star), which also achieves strong performance.

D Detailed Ablation Study on Predictor Choice

Motivation and Setup. A critical design choice in the **TuneAhead** framework is the selection of the prediction model F. The ideal predictor must not only achieve state-of-the-art predictive accuracy (**G2**) but also align with our core principles of providing interpretable diagnostics (**G3**) and ensuring computational efficiency (**G1**). To empirically validate our choice of LightGBM, we conducted a comprehensive comparison against a diverse suite of strong and representative regression models:

- **Ridge Regression (Hoerl & Kennard, 1970)**: a powerful linear model to test the extent of non-linear relationships in the data.
- **Support Vector Regressor (SVR, RBF) (Drucker et al., 1997; Smola & Schölkopf, 2004)**: a classic, high-performance kernel-based method adept at capturing complex non-linearities.
- **Random Forest (Breiman, 2001)**: a state-of-the-art ensemble model based on bagging, serving as a direct comparison to LightGBM's boosting approach.
- **Multi-Layer Perceptron (MLP) (Rumelhart et al., 1986)**: a simple but representative neural network baseline for tabular data.

To ensure a fair and rigorous comparison, all models were trained on the identical meta-feature set, and each underwent a systematic hyperparameter search using 5-fold cross-validation on our training set.

Performance Analysis. The results of this comparison are presented in Figure D.1. Our analysis yields two key findings. First, a top tier of models clearly emerges, with LightGBM (RMSE= 1.470) and SVR (RMSE= 1.472) delivering nearly identical, state-of-the-art predictive accuracy. Random Forest (RMSE= 1.589) follows as another strong competitor. This confirms that a GBDT-based approach achieves performance that is on par with the best alternative methods for this task. Second, the simple MLP struggles to generalize effectively (RMSE= 4.172), a common outcome on heterogeneous, tabular meta-datasets where GBDT models often excel without extensive architectural tuning.

Justification for Selecting LightGBM. Given the statistically comparable accuracy of the top-performing models (LightGBM and SVR), our final selection was determined by the other two crucial design goals: interpretability and scalability.

Interpretability. LightGBM holds a decisive advantage. Its tree-based architecture integrates seamlessly with SHAP, enabling the precise, feature-level diagnostics that are central to **TuneAhead**'s mission. In contrast, while SVR is a powerful predictor, deriving similarly intuitive, local feature-level attributions from a kernel-based model is significantly more complex and less direct.

Scalability. Furthermore, LightGBM is substantially more scalable. Its training time complexity is more favorable than SVR's, particularly as the number of experiments (samples) in the meta-dataset grows. This computational efficiency is critical for the future development and application

of **TUNEAHEAD** to even larger and more diverse problem spaces, as discussed in our Future Work (Sec. 6).

Conclusion. While SVR demonstrates highly competitive accuracy on our current dataset, LightGBM's unique combination of top-tier accuracy, superior interpretability, and better scalability makes it the most principled and strategic choice for the **TUNEAHEAD** framework.

Method	RMSE↓	R^2↑	r↑	ROUGE-L@ $k$pp↑		
				k=1	k=2	k=3
TUNEAHEAD-Static-Only	3.91	0.76	0.87	16.4	28.9	41.5
Domain-Proxy Baseline	5.17	0.66	0.81	16.5	33.8	48.9
Early-Stop Extrapolation	3.36	0.56	0.75	15.0	34.3	51.3
Early-Dynamics Baseline	5.10	0.69	0.83	14.8	37.5	55.8
TUNEAHEAD-Dynamic-Only	3.74	0.74	0.85	17.7	35.6	60.1
ProxyLM	2.74	0.83	0.90	22.6	41.0	65.7
TUNEAHEAD (Full)	**2.67**	**0.83**	**0.91**	**24.3**	**41.7**	**68.5**

Table E.1: Summarization (Samsum) prediction quality. Target is ROUGE-L (%). Errors are in pp. ACC@ $k$pp uses $k \in \{1, 2, 3\}$.

E ADDITIONAL BENCHMARKS AND TASKS

E.1 TRUTHFULQA (MC2): SETUP AND METRICS

Dataset and protocol. We re-evaluate all 1,300+ fine-tuned checkpoints from the main meta-dataset on **TruthfulQA**, using the official multiple-choice setting. We keep the *same* 24 meta-features and the same predictor training protocol (LightGBM, train/val/test splits) as in the MMLU experiments.

Target score and prediction metrics. The ground-truth target is **MC2 accuracy (%)**. We report prediction quality using **RMSE/MAE** (percentage points, pp), R^2, **Pearson** r, and **MC2@$k$pp** with $k \in \{3, 5\}$, i.e., the fraction of test cases whose absolute prediction error is within k pp. All errors are in **pp** to make thresholds comparable across benchmarks.

Results and analysis. Across all metrics, **TUNEAHEAD** achieves the lowest RMSE (2.17 pp) and the best calibration-by-accuracy (ACC@3pp=74.8%, ACC@5pp=88.2%), substantially outperforming Early-Stop and domain-only proxies. Notably, **TUNEAHEAD**-Static-Only is already competitive with domain proxies, while adding dynamic probe features closes the gap to the full model—indicating that both dataset quality signals and early learning dynamics are necessary to capture TruthfulQA-specific behavior without overfitting to MMLU.

E.2 SUMMARIZATION (SAMSUM; ROUGE-L): SETUP AND METRICS

Mini meta-dataset construction. We construct a **Summarization** mini meta-dataset (N=450 runs) using *Samsum*, applying the same data transformation knobs as in the main study (subsampling, domain slicing, light noise, template variants). Each variant is fully fine-tuned and evaluated by **ROUGE-L F1 (%)**.

Prediction target and metrics. The target to predict is the final **ROUGE-L (%)**. We report **RMSE/MAE** (pp), R^2, r, and **ROUGE-L@$k$pp** with $k \in \{1, 2, 3\}$, which are suitable pp-thresholds for ROUGE-L.

Summarization and analysis. Although ROUGE-L for abstractive summarization is inherently noisier than multiple-choice accuracy (due to decoding randomness and lexical-overlap sensitivity), **TUNEAHEAD** still delivers strong predictive performance on **Samsum (ROUGE-L)**. Using the same 24 meta-features, **TUNEAHEAD** (Full) achieves **RMSE=**1.67 **pp**, $R^2 = 0.83$, and $r = 0.91$, edging out the best non-**TUNEAHEAD** proxy (ProxyLM: RMSE=1.74 pp, $R^2 = 0.83$, $r = 0.90$). Accuracy-within-tolerance also improves (ACC@1pp: **24.3%** vs. 22.6%; ACC@2pp: **41.7%** vs. 41.0%; ACC@3pp: **68.5%** vs. 65.7%). Notably, *Static-Only* underperforms *Dynamic-Only* at tight tolerances, indicating that early loss/gradient dynamics are especially informative for sequence-quality objectives, while static dataset cues are necessary but insufficient on their own. Overall, these results suggest our meta-features capture not only classification-style generalization (MMLU, TruthfulQA) but also **generative quality** under summarization metrics.

Cost note. As in the main study, static features incur a one-time offline cost (≈0.2% of a full run), while the 100-step probe adds ≈4.3%. We therefore keep the total overhead under 5% even for the new tasks.