
Reasoning-Intensive Regression

Diane Tchuindjo
Operations Research Center
MIT
dianetc@mit.edu

Omar Khattab
EECS & CSAIL
MIT
okhattab@mit.edu

Abstract

AI researchers and practitioners increasingly apply large language models (LLMs) to what we call reasoning-intensive regression (RiR), i.e., deducing subtle numerical scores from text. Unlike standard language regression tasks, e.g., for sentiment or similarity, RiR often appears instead in ad-hoc problems such as rubric-based scoring, modeling dense rewards in complex environments, or domain-specific retrieval, where much deeper analysis of context is required while only limited task-specific training data and computation are available. We cast four realistic problems as RiR tasks to establish an initial benchmark, and use that to test our hypothesis that prompting frozen LLMs and finetuning Transformer encoders via gradient descent will both often struggle in RiR. We then propose **MENTAT**, a simple and lightweight method that combines batch-reflective prompt optimization with neural ensemble learning. **MENTAT** achieves up to 65% improvement over both baselines, though substantial room remains for future advances in RiR.

1 Introduction

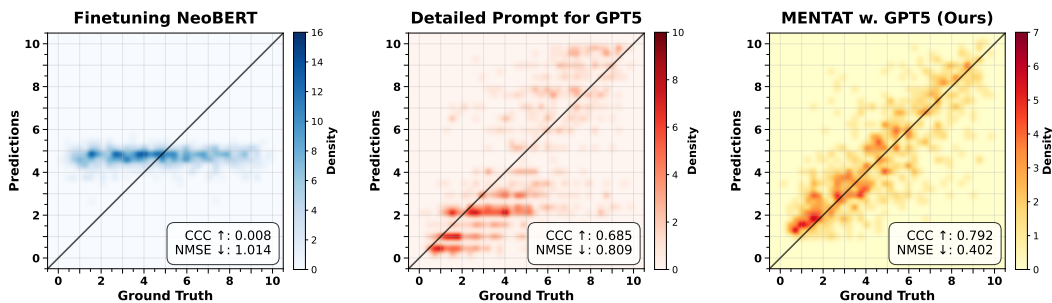


Figure 1: On regression for detecting the first math error, finetuning a NeoBERT model collapses to mean predictions ($\text{CCC} = 0.01$). Meanwhile, detailed (human-crafted) prompting achieves reasonable concordance ($\text{CCC} = 0.69$) but exhibits coarse and imprecise prediction behavior (the dense horizontal lines and near-random NMSE). **MENTAT**’s performance illustrates how RiR problems benefit from combining deep reasoning capabilities with precise numerical predictions.

Despite fast progress in adapting large language models (LLMs) for building downstream AI systems, lightweight methods for adapting LLMs to even standard *natural-language regression* tasks remain surprisingly elusive (Lukasik et al., 2024b,a; Tang et al., 2024; Song et al., 2025; Song & Bahri, 2025). These tasks, like sentiment analysis, semantic similarity, and document ranking, involve predicting a score $y \in \mathbb{R}$ from a natural-language string. Surprisingly, on these problems, applying straightforward supervised learning to pretrained Transformer encoders such as BERT Devlin et al.

(2019) has been shown to perform competitively with much larger decoder-only LLMs [Lukasik et al. \(2024a\)](#), even with sophisticated fine-tuning methods.

We investigate what we call *Reasoning-Intensive Regression* (RiR), a fuzzy but growing subset of natural-language regression in which processing the text in each instance *demand sequential deduction or deep analysis, rather than shallow identification of features*. Unlike simpler regression tasks, RiR problems call for explicit step-by-step problem decomposition or *reasoning*, where the system produces intermediate sequences of steps like tokens $\langle r_1, \dots, r_t \rangle \in \Sigma^*$ before committing to a prediction ([Merrill & Sabharwal, 2024](#)). See Figure 2 for a breakdown of regression problems into three levels of complexity: feature-based, semantic analysis, and reasoning-intensive, inspired by [Su et al. \(2025\)](#)’s analysis of retrieval tasks.

These types of applications are emerging rapidly in both research and practice, e.g. to produce scores for ad-hoc applications that process customer calls, student essays, rubric-based LLM generation, or instruction-based query–document relevance ([MacDonald, 2024](#); [Es et al., 2024](#); [Su et al., 2025](#); [Thakur et al., 2025](#)). In parallel, the same scoring paradigm is being scaled in recent efforts toward general-purpose chain-of-thought reward models [Kimi Team \(2025\)](#); [Ankner et al. \(2024a\)](#), but these typically assume orders-of-magnitude more labels and compute (e.g., hundreds of thousands of labels in K2) than the lightweight application-specific regimes that are far more common in the long tail.

We establish an initial benchmark for RiR by casting four realistic tasks as regression problems that demand *varying* levels of reasoning: predicting the proportion of a long mathematical deduction up to the first erroneous statement, determining the extent to which an LLM can follow highly composite instructions, predicting the degree to which the response of one Retrieval-Augmented Generation (RAG) system is better than another, and grading student essays on supplied topics. We then identify two practical constraints of downstream applications of RiR: these applications tend to offer only (very) small training sets and have room only for accessible and lightweight computations like LLM inference, lightweight forms of LLM prompt optimization, and finetuning medium-sized neural networks such as small Transformers, but not, say, large-scale reinforcement learning for large language models ([DeepSeek-AI, 2025](#); [Kimi Team, 2025](#)).

We ask: *Are there effective methods that are data- and compute-efficient for tackling ad-hoc reasoning-intensive regression problems?* We hypothesize that what makes RiR problems especially challenging is that they combine the reasoning need for *deep analysis of each individual task instance* with the regression challenge of *learning to produce precise, calibrated, and well-ranked scores* from very little data. As illustrated in Figure 1, standard prompt engineering techniques struggle with the high precision needed for learning to approximate a statistical distribution, while approaches that bypass LLM-based reasoning, e.g., training small Transformer encoders, often fail to truly learn RiR problems and instead seek to “hack” the regression loss function by finding degenerate approximations (e.g., collapsing to a small range of scores).

We propose **Mistake-Aware prompt Evolver with Neural Training And Testing** (MENTAT), a simple and lightweight method that combines iterative prompt optimization with neural regression. Rather than relying on LLMs to produce precise numerical predictions directly, which often results in brittle outputs, MENTAT uses an iterative error-driven prompt evolution process. Starting with even just a very basic prompt, the LLM analyzes its own prediction errors *in large batches*, identifies patterns of its poor performance, and then refines the prompt based on that. After few iterations, MENTAT trains a simple aggregation MLP to reduce multiple rollouts from the LLM-discovered prompt into a final prediction. MENTAT delivers consistent improvements in quality, but nonetheless leaves large headroom on many of the RiR settings we define.

The remainder of the study is as follows: Section 2 describes how we translate four problems into RiR tasks and Section 3 introduces MENTAT. Section 4 presents our evaluation methodology, including the details of our baselines, and the results. The paper concludes with Sections 5 and 6, which synthesize our findings and discuss implications for future research. An extended discussion of related work is given in the Appendix A.

2 Benchmarking RiR

We collect four tasks for Reasoning-Intensive Regression of varying degrees of reasoning intensity. Refer to Figure 3 for the dataset distributions.

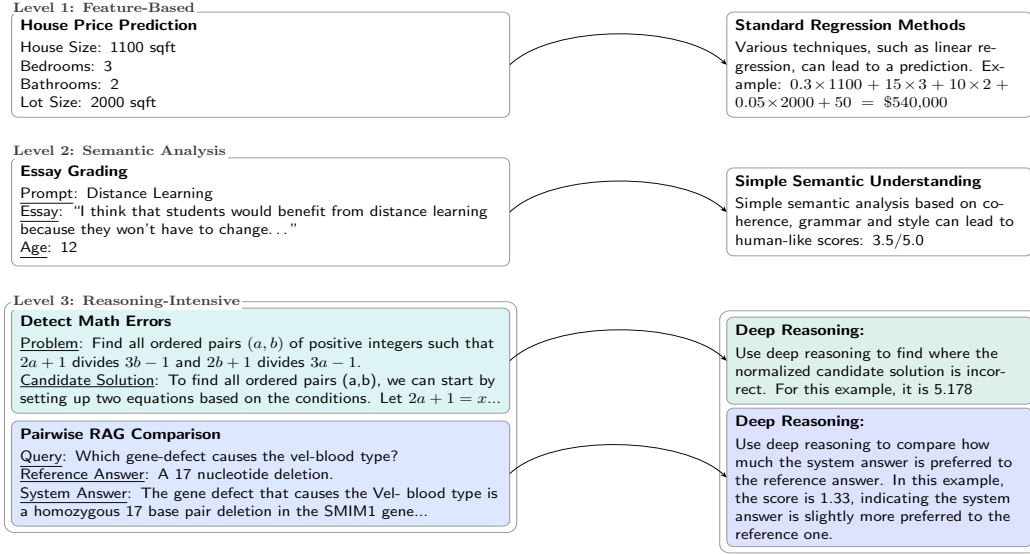


Figure 2: Inspired by [Su et al. \(2025\)](#)’s analysis of retrieval tasks, we break down text-based regression problems into three, informal complexity levels. Level 1 tasks use simple feature-based inputs (for example, the number of bedrooms and bathrooms when predicting home prices). Text-to-text regression achieves strong Level 1 performance with rich datasets ([Akhaouri et al., 2025](#)). Level 2 tasks require moderate semantic understanding (sentiment analysis, reward modeling) but are easy for supervised-learning over a pretrained Transformer. Level 3, the focus of this work, represents Reasoning-Intensive Regression (RiR), which requires deep sequential reasoning.

- **Mathematical Error Detection** requires precise logical reasoning and stepwise analysis, while also stressing the fact that LLMs are known to struggle with precisely estimating simple properties like text length.
- **Instruction Following** evaluates how well a response satisfies a set of fine-grained requirements, and expects models to produce calibrated scalar judgments.
- **Pairwise RAG Comparison** emphasizes the need for models to perform nuanced human-like judgment and contextual understanding, demanding sophisticated evaluative reasoning.
- **Essay Grading** serves as a reference point, requiring semantic understanding where encoders like BERT might already perform well with a reasonable amount of finetuning data.

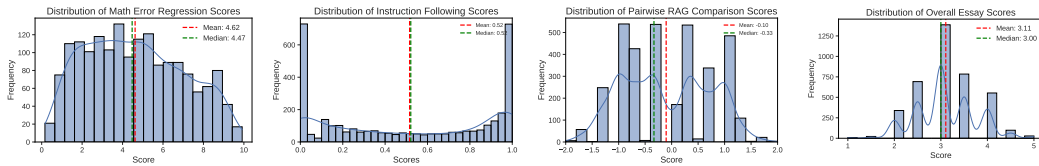


Figure 3: Ground-truth score distributions for mathematical error detection (the spread capturing the tendency for solutions to fail towards the center), instruction following (capturing the tendency to favor the tails), pairwise RAG comparison (narrow distribution around averaged judgments), and essay grading (tight clustering characteristic of qualitative assessments).

Regression Metrics Normalized Mean Square Error (NMSE) is a common metric for reporting regression performance: $\sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$, where n is the size of the dataset, \hat{y}_i is a prediction, y_i the corresponding ground truth value, and \bar{y} is the mean of the set.

But distance-based metrics are inadequate for typical RiR problems; RiR systems can artificially lower their NMSE simply by avoiding “risky” predictions at the extremes. This can be seen in Figure

1 earlier, particularly in comparing the fine-tuned NeoBERT model Breton et al. (2025) against detailed (human-crafted) prompting. Following Figure 1, if we were to rely on NMSE, detailed prompting for GPT-5 would not appear to substantially outperform NeoBERT (0.81 vs. 1.01), and this gap would be even reversed for weaker LLMs. Examining the distribution of predictions reveals that NeoBERT “hacked” the loss function by learning a collapsed distribution, while the prompted LLM actually shows substantial signs of ranking the inputs correctly.

This can be captured in a Concordance Correlation Coefficient (CCC) of 0.01 for NeoBERT versus a CCC of 0.69 for detailed prompting. We thus suggest the use of the CCC as an additional, and perhaps more appropriate, RiR metric. CCC measures both correlation and agreement, defined as $\frac{2\rho\sigma_y\sigma_{\hat{y}}}{\sigma_y^2 + \sigma_{\hat{y}}^2 + (\mu_y - \mu_{\hat{y}})^2}$, where ρ is the Pearson correlation coefficient between predictions \hat{y} and ground truth y , σ_y and $\sigma_{\hat{y}}$ are their respective standard deviations, and μ_y and $\mu_{\hat{y}}$ are their means. CCC penalizes systematic bias and rewards predictions that maintain the natural variance of the distribution.

Detecting Mathematical Errors We derive a dataset for predicting the *fraction* of a mathematical solution up to the first erroneous reasoning step, given a problem and incorrect solution in LaTeX, from ProcessBench (Zheng et al., 2024). To effectively do this, a model must systematically reason formally about math steps rather than relying on probabilistic heuristics, but it must also be good at estimating relative lengths and inferring the boundaries of the steps in a calibrated way.

To convert the original classification task into a regression problem, we first filter out problems with correct solutions or final answers. We then merge all solution steps into a single continuous text $T = s_1 \| s_2 \| \dots \| s_n$ (here $\|$ denotes concatenation). Next, for a solution with error at step k , the regression score R is $10 \times (\sum_{i=1}^{k-1} |s_i| + \frac{1}{2}|s_k|) / |T|$ where $|s_i|$ denotes the length of step i , and $|T|$ is the total length of the concatenated solution. See an example entry in Appendix D.

Instruction Following We derive a task from the WILDIFEVA corpus Lior et al. (2025) that targets *instruction-following* in long-form generation. Each example consists of: (i) a user *task* prompt; (ii) a list of atomic requirements (the *decomposition*); (iii) a *model answer* produced by Llama-3.1-8B (zero-shot); and (iv) per-requirement satisfaction scores originally produced by Llama-3.1-70B acting as an automatic judge. The goal is to predict a single continuous label $y \in [0, 1]$ that reflects the overall degree to which the answer adheres to the decomposed instructions. More precisely, for each decomposition instance, the judge produced a probability-like score $s_i \in [0, 1]$ for each requirement r_i , $i = 1, \dots, K$. We then use the harmonic mean of these scores as our overall judgment, emphasizing the need to adhere well to all task requirements. To test instruction following, we do not expose the decomposition to NeoBERT or an LLM; instead, they are only given the task and model answer and must infer the overall score.

Pairwise RAG Comparison We derive a dataset for comparing two LLM outputs on a scale from the RAG-QA evaluations (Han et al., 2024). Each query $q \in \mathcal{Q}$ has responses A_1, A_2 and a target comparative score from -2 to 2 representing the *average* annotation of three human judges, who were instructed to assess response helpfulness, truthfulness, and completeness. Here, positive scores means that A_1 is better (and vice versa). This task partially aligns with RiR as judging the outputs and comparing them in light of each query often requires nuanced judgment.

Essay Grading We lastly use an essay grading dataset Crossley et al. (2023), where each entry contains among other features an essay prompt, a student (grade 8–12) response, associated demographic information, and an overall score between 1 and 5. Although Essay Grading is simpler than the rest, it serves as a reference point for the other RiR tasks.

We evaluate these tasks using two proprietary LLMs (GPT-4 . 1, GPT-5) across three tasks, plus an open-source model (gpt-oss-20b) on Instruction Following for reproducibility and generalization validation.

3 MENTAT

MENTAT combines two simple ideas, depicted in Figure 4: it allows the LLM itself to reflect in batches to incrementally adjust its own prompt, and it aggregates multiple rollouts from the optimized LLM system with a simple trained MLP.

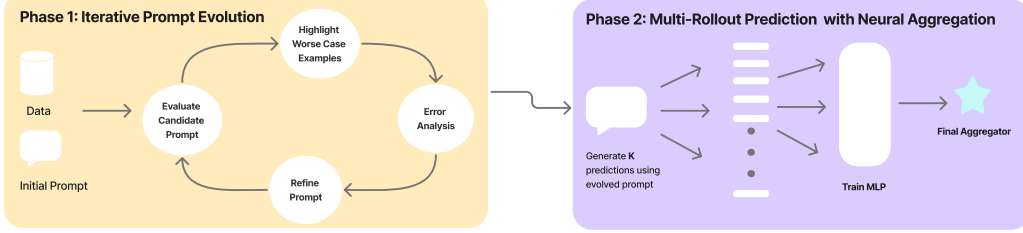


Figure 4: Phase 1 performs prompt evolution through iterative and *batched* reflection. Given a candidate prompt, we collect rollouts on n samples, divided into training and validation sets. A model instructed to focus on the \sqrt{n} worst-performing examples analyzes the rollouts on the training samples, in light of the optimization history from previous iterations, and makes proposals that refine the prompt. This cycle continues for a predetermined number of iterations, after which we select the best-performing prompt P_{best} as evaluated on the validation set. Phase 2 generates multi-rollout predictions by applying P_{best} and aggregating K stochastic predictions per input and trains a neural aggregator f_{θ} on sorted rollouts using a combined CCC–NMSE loss. Test predictions are obtained by sampling test rollouts and applying the trained aggregator f_{θ^*} .

Phase 1: Prompt Evolution MENTAT’s first step is to make sure that the LLM prompt reflects both *local* instructions for reasoning about each input and *global* guidance about the distribution of ground-truth scores. Though any approach for prompt optimization can be used here, e.g., MIPRO Opsahl-Ong et al. (2024) or GEPA Agrawal et al. (2025), through preliminary experiments we identified two special properties in RiR tasks that call for different design choices.

First, performing rollouts with powerful *reasoning* models can be expensive and slow, when compared to standard LLMs, for which existing optimizers were built. To remain within the lightweight constraints of typical RiR tasks, a suitable prompt evolution stage would have to minimize *both* the number of rollouts performed with the LLM and the number of *inherently sequential stages* or iterations of optimization. Second, optimizing prompts for RiR tasks has the fairly unique property that the patterns *across* examples are at least as important as the per-example error. This is because MENTAT’s aggregation design demonstrates that it can be easy to turn a *well-calibrated* system into one that has low pointwise error, but the reverse is not necessarily true.

This motivates us to test an exceedingly simple reasoning-based technique for optimizing LLM systems that contain a single prompt.¹ Unlike existing prompt optimizers, which typically seek to make improvements driven by individual failures (or very small batches of failures), MENTAT’s prompt evolver is centered around asking the LLM to *jointly reason about tens of mistakes at once*. This simple design is inspired by human prompt engineering practice (Husain & Shankar, 2024).

Concretely, we proceed in a very small number of sequential iterations (three in our experiments). In each iteration, the work is highly parallelizable: we evaluate the current prompt on a shuffled sample of the training set, and then concatenate all of the rollouts for analysis by the same LLM. It is then asked to identify systematic errors by analyzing the worst-performing examples and to generate improved instructions. In each iteration, the LLM receives three key inputs: current instructions, performance analysis with detailed error patterns, and a formatted history of previous optimization attempts. This historical context prevents the method from cycling through previously unsuccessful approaches and enables progressive refinement. At the end of this process, the best-performing prompt (via NMSE or CCC) is selected on a separate validation set.

¹We leave extending this method to multi-stage LLM programs and conducting an extensive comparison of different prompt optimization strategies to future work. Indeed, preliminary experiments with GEPA (NMSE 0.52 vs. 0.54, CCC 0.72 vs. 0.70 on math errors) indicate that the prompt evolution phase of our approach—on its own—performs comparably to GEPA, while better aligning with RiR constraints.

In our evaluation, to stress MENTAT, we start from a deliberately basic prompt for each task, to reflect a more challenging setting.² Note also that this iterative prompt evolution follows a single optimization trajectory. In principle, MENTAT could employ multiple random restarts, which could be parallelized to explore diverse regions of the prompt space. However, we focus on single-trajectory optimization both for computational efficiency and algorithmic simplicity.

Phase 2: Multi-Rollout Generation with Neural Aggregation Using the best LLM-discovered prompt from Phase 1, MENTAT generates multiple independent predictions for each example. The multi-rollout approach captures the inherent uncertainty in LLM predictions, as each rollout can reason independently, and provides richer signal for the subsequent neural aggregation phase. In practice, we set this to three rollouts per example. We train a small Multi-Layer Perceptron (MLP) to combine rollout predictions. The aggregator ensures order invariance by sorting rollout predictions, incorporates statistical features (mean, standard deviation, min, max), and is optimized for a combination of the CCC and NMSE loss functions. Overall, this method can be understood as a variant of self-consistency Wang et al. (2023) or best-of- N voting Stiennon et al. (2020); Snell et al. (2024), but it leverages more information across the rollouts.

4 Evaluation

We define two standard baselines for RiR problems: fine-tuning a small Transformer encoder and prompting an LLM, and use these two to understand the relative merits of our method MENTAT and to develop a series of ablation experiments.

4.1 Baseline: Finetuning a Transformer Encoder

We formulate RiR as supervised regression using a 250M-parameter NeoBERT model. The architecture processes minimally formatted text sequences (e.g., combining problem statements with solutions for math errors, augmented with domain-specific prompts).

Inputs are tokenized using NeoBERT’s byte-level BPE tokenizer, truncated or padded to 1024 tokens, and passed through the pretrained encoder. The model extracts representations from the [CLS] token, applies dropout regularization ($p = 0.2$), and uses linear projection for scalar predictions. The optimization objective minimizes weighted NMSE and CCC using AdamW (Loshchilov & Hutter, 2019). This architecture requires only prompt templating beyond standard fine-tuning, with hyperparameters detailed in Appendix C.1.

4.2 Baseline: Prompting a Large Language Model

We employ Chain-of-Thought style prompting to encourage frozen LLMs to perform explicit reasoning through step-by-step token generation. Our evaluation uses two proprietary models with different reasoning capabilities: GPT-4 . 1 (non-reasoning) and GPT-5 (reasoning) across three tasks (Mathematical Error Detection, Pairwise RAG Comparison, and Essay Grading). For the Instruction Following task, we employ gpt-oss-20b, an open-source model, to demonstrate that our method generalizes beyond proprietary systems and to provide more easily reproducible baselines for the community. The detailed prompts for all tasks help guide the decomposition of complex inputs and the templates can be found in Appendix E.

This approach is motivated by several practical advantages. Frozen LLMs can act as a unified interface across various natural language tasks, with no or very little training data. This is especially valuable in reasoning-intensive regression (RiR) tasks where annotated datasets are often scarce. Utilizing a shared, unified, and amortized infrastructure (i.e., LLM servers) enables us to deploy a single model across many tasks, significantly reducing the computational and financial overhead compared to training multiple specifiable models.

²Examples of the basic vs. the detailed prompts used for the four tasks can be found in Appendices G and E), respectively. They differ in the inclusion of detailed procedural steps, calibration guidance, and/or domain-specific heuristics that human experts may decide to include.

LM	Method	Math Errors				Pairwise RAG				Essay Grading			
		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑	
		100	500	100	500	100	500	100	500	100	500	100	500
Main Methods													
NeoBERT	Gradient Descent	1.05	1.01	0.02	0.06	1.44	1.02	0.01	0.10	1.03	0.91	0.19	0.65
GPT-4.1	Basic Prompt	1.59	1.59	0.36	0.36	2.18	2.18	0.47	0.47	0.75	0.75	0.63	0.63
	Detailed Prompt	1.13	1.13	0.52	0.52	2.20	2.20	0.47	0.47	0.73	0.73	0.65	0.65
	MENTAT _{Basic Prompt}	0.87	0.76	0.51	0.49	0.77	0.80	0.50	0.52	0.54	0.53	0.70	0.68
GPT-5	Basic Prompt	0.77	0.77	0.66	0.66	2.25	2.25	0.35	0.35	1.31	1.31	0.42	0.42
	Detailed Prompt	0.78	0.78	0.69	0.69	2.18	2.18	0.31	0.31	1.53	1.53	0.40	0.40
	MENTAT _{Basic Prompt}	0.52	0.42	0.72	0.78	1.07	0.93	0.36	0.33	0.64	0.67	0.59	0.55
Ablations													
GPT-4.1	MENTAT Prompt	1.39	1.29	0.45	0.48	2.00	1.69	0.45	0.48	0.61	0.71	0.68	0.66
	MENTAT-Avg	1.00	1.01	0.52	0.52	1.82	1.48	0.48	0.51	0.57	0.63	0.69	0.68
GPT-5	MENTAT Prompt	0.66	0.58	0.66	0.72	1.43	1.95	0.33	0.30	0.74	0.70	0.57	0.54
	MENTAT-Avg	0.59	0.51	0.68	0.75	1.31	1.83	0.35	0.32	0.69	0.67	0.57	0.55

Table 1: Performance comparison across Mathematical Error Detection, Pairwise RAG Comparison, and Essay Grading using GPT-4.1 and GPT-5 as our models. Each entry is the average of three independent runs on a test set of size 750. Total training sizes are 100 and 500 (train/val combined). **Ablations:** MENTAT Prompt uses only error-driven prompt refinement on training data; MENTAT-Avg uses MENTAT prompting + multiple rollouts + averaging (instead of an MLP like in full MENTAT). We remark here that NeoBERT obtains an average NMSE and CCC of 0.60 and 0.66 respectively, on a training regime of 1500 (1000 training + 500 validation) on Pairwise RAG Comparison. That is, NeoBERT needs much more data on this task to lead to good performance, but it can be achieved.

4.3 Experimental Setup

Our experimental design evaluates MENTAT across four reasoning-intensive regression tasks using a structured approach. We test GPT-4.1 and GPT-5 on three tasks (Mathematical Error Detection, Pairwise RAG Comparison, and Essay Grading), while Instruction Following uses gpt-oss-20b to demonstrate generalization to open-source models.

For the three tasks using proprietary models, we employ 750 test examples with results averaged across three independent runs. We evaluate under two training configurations (100 and 500 samples) that reflect real-world data constraints typical in ad-hoc RiR applications. For prompt optimization methods, we use a balanced train/validation splits of 50 + 50 and 250 + 250 samples and for NeoBERT finetuning, we employ a training-heavy splits of 50 + 50 and 350 + 150 samples to leverage the model’s supervised learning capabilities.

For Instruction Following, we use a single configuration with 500 training, 500 validation, and 2000 test samples, reflecting the different data availability typical for this fundamental capability assessment. This larger test set enables more robust evaluation of the nuanced instruction-adherence requirements.

This experimental structure allows us to assess MENTAT’s effectiveness across different model capabilities (reasoning vs. non-reasoning), data regimes (limited vs. moderate training data), and model accessibility (proprietary vs. open-source), providing comprehensive validation of our approach for practical RiR applications.

4.4 Results

Our main evaluation results are reported in Table 1 and 2, demonstrating significant performance variations across methods and tasks. The results reveal distinct patterns in how different approaches handle reasoning-intensive regression problems, with MENTAT consistently outperforming baseline methods across most configurations.

Mathematical Error Detection Performance On this task, finetuning NeoBERT achieves near-zero CCC scores across both training configurations and effectively collapsing to mean predictions as shown in Figure 1. In contrast, LLM-based approaches demonstrated substantial reasoning

LM	Method	Instruction Following	
		NMSE ↓	CCC ↑
NeoBERT	Gradient Descent	1.08	0.36
GPT-OSS-20B	Basic Prompt	1.18	0.32
	Detailed Prompt	1.16	0.33
	MENTAT _{Basic Prompt}	0.95	0.42
	MENTAT _{Detailed Prompt}	0.90	0.43
Ablations			
GPT-OSS-20B	MENTAT _{Basic Prompt} Prompt	1.25	0.35
	MENTAT _{Basic Prompt} -Avg	1.06	0.38
	MENTAT _{Detailed Prompt} Prompt	1.27	0.36
	MENTAT _{Detailed Prompt} -Avg	1.09	0.39

Table 2: Performance on the Instruction Following task using the `gpt-oss-20b` model. Each entry is the average of three independent runs on a test set of size 2000. Total training configuration uses 500 training and 500 validation samples. Ablations: MENTAT Prompt uses only error-driven prompt refinement on training data; MENTAT-Avg uses MENTAT prompting + multiple rollouts + averaging (instead of an MLP like in full MENTAT). The subscripts, basic prompt and detailed prompt, are what we use as the initial prompt in the MENTAT framework.

capabilities. GPT-4.1 with detailed prompting achieved CCC scores of 0.52 (100-sample training) and maintained this performance at 500 samples. However, MENTAT with GPT-4.1 showed only modest improvements, reaching CCC scores of 0.51 (100 samples) and 0.49 (500 samples), representing approximately stable performance with slight variation. We hypothesize that GPT-4.1’s limited reasoning capabilities on this reasoning-intensive task made it difficult to understand its own errors and thus improve.

The most dramatic improvements can be seen with GPT-5. While detailed prompting with GPT-5 achieved strong baseline performance (CCC: 0.69, NMSE: 0.78), MENTAT with GPT-5 delivered substantial enhancements. In the 100-sample training regime, CCC improved by 4.3%, while NMSE improved by 33.3%. In the 500-sample training regime, CCC improved by 13%, while NMSE improved by 46.2%. These results indicate that MENTAT’s iterative prompt refinement and neural aggregation effectively leverage GPT-5’s reasoning capabilities while addressing the precision limitations inherent in direct LLM numerical prediction.

Instruction Following Performance On instruction following, NeoBERT achieved modest performance (CCC: 0.36, NMSE: 1.08), while `gpt-oss-20b` with basic and detailed prompting showed similar limitations (CCC: 0.32-0.33, NMSE: 1.16-1.18). Both approaches struggled with the nuanced evaluation required for assessing instruction adherence.

MENTAT demonstrated clear improvements across both initialization strategies. Starting from the basic prompt, MENTAT achieved CCC of 0.42 and NMSE of 0.95 (31% and 19% improvements respectively). With detailed prompt initialization, MENTAT reached CCC of 0.43 and NMSE of 0.90 (30% and 22% improvements). The ablation results show that both prompt evolution and neural aggregation contribute to MENTAT’s success on this task, with the full method consistently outperforming the individual components across both initialization strategies.

Pairwise RAG Comparison Performance On the pairwise RAG comparison task, finetuning NeoBERT achieved very low CCC scores while appearing competitive on the NMSE metric by “hacking” the distribution. Surprisingly, GPT-4.1 demonstrated superior performance compared to GPT-5 on this task, in sharp contrast with the general trend observed in mathematical error detection. Detailed prompting with GPT-4.1 achieved CCC scores of 0.47 across both training configurations, while GPT-5 detailed prompting resulted in lower CCC scores of 0.31.

Unlike math errors, instruction following, and essay grading tasks, correct decisions on the pairwise RAG benchmark often hinge on a few salient cues and short justifications. With chain-of-thought scaffolds on this task, we observe that GPT-5 systematically “overthinks,” resulting in predictions that concentrate near the center (0 on the $[-2, 2]$ margin) rather than faithfully spreading across the empirical label distribution. As shown in Figure 5, its variance is under-dispersed relative to ground truth, with more than half of examples yielding identical rollouts across three samples. Rollout

correlations are very high, and the final numbers fall on a coarse grid (e.g., $\{-1, -\frac{1}{3}, 0, \frac{1}{3}\}$), all consistent with hedging.

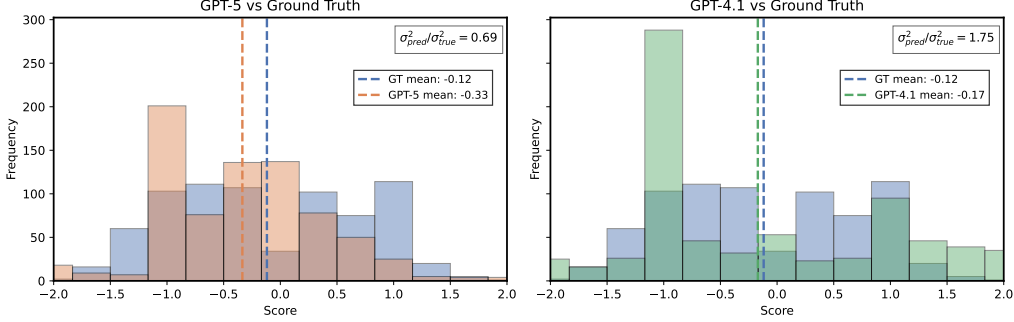


Figure 5: Pairwise RAG distributions on the mean of three rollouts vs. ground truth after the prompt evolution process. GPT-5 (left) is center-seeking and under-dispersed; GPT-4.1 (right) stays closer to the empirical mean and exhibits greater spread. This behavior leads to higher CCC for GPT-4.1.

By contrast, GPT-4.1 produces short, decisive judgments that remain closer to the dataset mean with greater spread and more frequent use of the extremes. Although GPT-4.1 rollouts are also correlated, the resulting distribution retains enough variance and calibration to yield substantially higher CCC. For pairwise RAG, GPT-5 tends toward the center and compresses its numeric range, degrading distributional fidelity (and thus CCC) even when NMSE remains similar.

We hypothesize that GPT-4.1’s superior performance on pairwise RAG comparison aligns with recent findings that large reasoning models often underperform on simpler tasks (Shojaee et al., 2025). These models initially find correct solutions but continue reasoning toward incorrect answers, suggesting that excessive sophisticated reasoning can sometimes be counterproductive. This hypothesis is supported by our observation that over half of GPT-5’s examples yield identical rollouts across three samples, with final scores clustering on a coarse grid rather than reflecting the task’s inherent variance.

Essay Grading Performance Essay grading represented the least complex reasoning-intensive task, with NeoBERT achieving reasonable performance that improved substantially with additional training data. This aligns with the task’s characterization as requiring primarily semantic understanding rather than deep sequential reasoning. GPT-4.1 achieved strong baseline performance with detailed prompting (CCC: 0.65, NMSE: 0.73), while MENTAT provided meaningful improvements. In the 100-sample training regime, CCC improved by 7.7% and NMSE improved by 26.0% compared to detailed prompting. In the 500-sample training regime, CCC improved by 4.6% and NMSE improved by 27.4%. Notably, GPT-5 performance on essay grading showed surprisingly poor concordance compared to GPT-4.1, supporting the hypothesis discussed in Section 4.4 that sophisticated reasoning models may over-deliberate on simpler tasks.

5 Conclusion

We investigated *reasoning-intensive regression* (RiR). Our empirical findings reveal tension: prompting leverages LLMs’ reasoning capabilities but produces quantized, imprecise outputs, while supervised finetuning for regression can often collapse without learning the task. We proposed MENTAT, a simple method that suggests that hybrid approaches may help address this tension through iteratively optimizing the prompts via batched error analysis combined with neural aggregation, achieving consistent improvements across RiR tasks.

However, our work opens several rich avenues for future research. The RiR framework we establish creates opportunities to systematically evaluate sophisticated prompt optimization techniques such as GEPA and develop RiR-adapted regression-aware finetuning methods (Lukasik et al., 2024b; Chiang et al., 2025). Our lightweight constraint focus also motivates exploring the efficiency-performance trade-offs in reasoning-intensive tasks. While reinforcement learning methods like Group Relative Policy Optimization Shao et al. (2024) require thousands of rollouts beyond current

practical limits, our benchmark provides a testbed for developing more efficient alternatives as RiR datasets scale. Similarly, MENTAT’s $3\times$ inference cost increase highlights the need for systematic cost-benefit analysis across deployment scenarios, opening questions about adaptive rollout strategies and inference-time optimization that our tasks can help address.

Acknowledgments

This work used Expanse GPU at the San Diego Supercomputer Center (SDSC) through allocation CIS250733 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS; Boerner et al. 2023) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. This research was partly supported by Laude Institute.

References

- Lakshya A Agrawal, Shangyin Tan, Dilara Soyulu, Noah Ziemis, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gega: Reflective prompt evolution can outperform reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.19457>.
- Yash Akhauri, Bryan Lewandowski, Cheng-Hsi Lin, Adrian N. Reyes, Grant C. Forbes, Arissa Wongpanich, Bangding Yang, Mohamed S. Abdelfattah, Sagi Perel, and Xingyou Song. Performance prediction for large systems via text-to-text regression, 2025. URL <https://arxiv.org/abs/2506.21718>.
- James Urquhart Allingham, Jie Ren, Michael W. Dusenberry, Xiuye Gu, Yin Cui, Dustin Tran, Jeremiah Zhe Liu, and Balaji Lakshminarayanan. A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org, 2023.
- Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*, 2024a.
- Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D. Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models, 2024b. URL <https://arxiv.org/abs/2408.11791>.
- Simran Arora, Avaniika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher R. Ask me anything: A simple strategy for prompting language models, 2022. URL <https://arxiv.org/abs/2210.02441>.
- Daniel Beaglehole, Adityanarayanan Radhakrishnan, Enric Boix-Adser, and Mikhail Belkin. Aggregate and conquer: detecting and steering llm concepts by combining nonlinear predictors over multiple layers, 2025. URL <https://arxiv.org/abs/2502.03708>.
- Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. ACCESS: Advancing innovation: Nsf’s advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing (PEARC 23)*, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3569951.3597559.
- L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996a. URL <https://api.semanticscholar.org/CorpusID:27026167>.
- Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123140, August 1996b. ISSN 0885-6125. doi: 10.1023/A:1018054314350. URL <https://doi.org/10.1023/A:1018054314350>.
- Leo Breiman. Random forests. *Mach. Learn.*, 45(1):532, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL <https://doi.org/10.1023/A:1010933404324>.
- Lola Le Breton, Quentin Fournier, Mariam El Mezouar, John X. Morris, and Sarath Chandar. Neobert: A next-generation bert, 2025. URL <https://arxiv.org/abs/2502.19587>.

- Cheng-Han Chiang, Hung yi Lee, and Michal Lukasik. Tract: Regression-aware fine-tuning meets chain-of-thought reasoning for llm-as-a-judge, 2025. URL <https://arxiv.org/abs/2503.04381>.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsveyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1), March 2024. ISSN 1532-4435.
- Scott A. Crossley, Yu Tian, Perpetual Baffour, Alex Franklin, Youngmeen Kim, Wesley Morris, Meg Benner, Aigner Picou, and Ulrich Boser. The english language learner insight, proficiency and skills evaluation (ellipse) corpus. *International Journal of Learner Corpus Research*, 2023. URL <https://api.semanticscholar.org/CorpusID:267599728>.
- Rbert Csords, Piotr Pikos, Kazuki Irie, and Jrgen Schmidhuber. Switchhead: Accelerating transformers with mixture-of-experts attention, 2024. URL <https://arxiv.org/abs/2312.07987>.
- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 150–158, 2024.
- William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN 1532-4435.
- Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In Philipp Koehn, Barry Haddow, Tom Kocmi, and Christof Monz (eds.), *Proceedings of the Eighth Conference on Machine Translation*, pp. 1066–1083, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.100. URL <https://aclanthology.org/2023.wmt-1.100/>.
- Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning, ICML’96*, pp. 148156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 1558604197.
- M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022. ISSN 0952-1976. doi: <https://doi.org/10.1016/j.engappai.2022.105151>. URL <https://www.sciencedirect.com/science/article/pii/S095219762200269X>.
- Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2024. URL <https://arxiv.org/abs/2310.07820>.

- Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jinyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. 2024. URL <https://arxiv.org/abs/2407.13998>.
- Yi-Chong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. Ensemble learning for heterogeneous large language models with deep parallel collaboration. 2024. URL <https://api.semanticscholar.org/CorpusID:269282634>.
- Hamel Husain and Shreya Shankar. Ai evals for engineers & pms. <https://maven.com/parlance-labs/evals>, 2024. Cohort-based course on building and iterating evaluation systems for AI products.
- Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion, 2023. URL <https://arxiv.org/abs/2306.02561>.
- Mingjian Jiang, Yangjun Ruan, Sicong Huang, Saifei Liao, Silviu Pitis, Roger Baker Grosse, and Jimmy Ba. Calibrating language models via augmented prompt ensembles. URL <https://api.semanticscholar.org/CorpusID:271797871>.
- M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp. 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.
- Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. Exploring demonstration ensembling for in-context learning. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.
- Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.
- Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier, 2023. URL <https://arxiv.org/abs/2206.02336>.
- Gili Lior, Asaf Yehudai, Ariel Gera, and Liat Ein-Dor. Wildifeval: Instruction following in the wild, 2025. URL <https://arxiv.org/abs/2503.06573>.
- Tiedong Liu and Bryan Kian Hsiang Low. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks, 2023. URL <https://arxiv.org/abs/2305.14201>.
- Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(99\)00073-8](https://doi.org/10.1016/S0893-6080(99)00073-8). URL <https://www.sciencedirect.com/science/article/pii/S0893608099000738>.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL <https://arxiv.org/abs/1711.05101>.
- Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models, 2023. URL <https://arxiv.org/abs/2311.08692>.
- Michal Lukasik, Zhao Meng, Harikrishna Narasimhan, Aditya Krishna Menon, Yin-Wen Chang, Felix Yu, and Sanjiv Kumar. Better autoregressive regression with LLMs. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=xGs7Ch3Vyo>. under review.
- Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Regression aware inference with LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13667–13678, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.799. URL <https://aclanthology.org/2024.findings-emnlp.799>.

- Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Ping Luo, and Yue Yu. URG: A unified ranking and generation method for ensembling language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4421–4434, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.261. URL <https://aclanthology.org/2024.findings-acl.261/>.
- Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Yue Yu, and Ping Luo. Specfuse: Ensembling large language models via next-segment prediction, 2024b. URL <https://arxiv.org/abs/2412.07380>.
- Lindsay MacDonald. Generative ai use case: Using llms to score customer conversations, July 2024. URL <https://www.montecarlodata.com/blog-generative-ai-use-case-assurance/>. Monte Carlo Data Blog.
- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Frnken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL <https://arxiv.org/abs/2410.12832>.
- Costas Mavromatis, Petros Karypis, and George Karypis. Pack of llms: Model fusion at test-time via perplexity optimization, 2024. URL <https://arxiv.org/abs/2404.11531>.
- William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought, 2024. URL <https://arxiv.org/abs/2310.07923>.
- Tung Nguyen, Qiuyi Zhang, Bangding Yang, Chansoo Lee, Jorg Bornschein, Yingjie Miao, Sagi Perel, Yutian Chen, and Xingyou Song. Predicting from strings: Language model embeddings for bayesian optimization, 2024. URL <https://arxiv.org/abs/2410.10190>.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024. URL <https://arxiv.org/abs/2406.18665>.
- Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs, 2024. URL <https://arxiv.org/abs/2406.11695>.
- Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning, 2024. URL <https://arxiv.org/abs/2412.15797>.
- Silviu Pitis, Michael R. Zhang, Andrew Wang, and Jimmy Ba. Boosted prompt ensembles for large language models, 2023. URL <https://arxiv.org/abs/2304.05970>.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarsz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL <https://arxiv.org/abs/1701.06538>.
- Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-experts meets instruction tuning:a winning combination for large language models, 2023. URL <https://arxiv.org/abs/2305.14705>.
- Parshin Shojaei, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL <https://arxiv.org/abs/2506.06941>.

- Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Boyd-Graber. Getting more out of mixture of language model reasoning experts, 2023. URL <https://arxiv.org/abs/2305.14628>.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- Xingyou Song and Dara Bahri. Decoding-based regression, 2025. URL <https://arxiv.org/abs/2501.19383>.
- Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors, 2024. URL <https://arxiv.org/abs/2402.14547>.
- Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors, 2025. URL <https://arxiv.org/abs/2402.14547>.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.
- Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval, 2025. URL <https://arxiv.org/abs/2407.12883>.
- Eric Tang, Bangding Yang, and Xingyou Song. Understanding llm embeddings for regression, 2024. URL <https://arxiv.org/abs/2411.14708>.
- Nandan Thakur, Ronak Pradeep, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. Support evaluation for the trec 2024 rag track: Comparing human versus llm judges. *arXiv preprint arXiv:2504.15205*, 2025.
- Chen Tianlong, Cheng Yu, Chen Beidi, Zhang Minjia, and Bansal Mohit. Mixture-of-experts in the era of llms: A new odyssey. ICML 2024 presentation slides, 2024. International Conference on Machine Learning (ICML).
- Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciuc, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples, 2024. URL <https://arxiv.org/abs/2404.07544>.
- Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models, 2024. URL <https://arxiv.org/abs/2404.18796>.
- Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise, 2024a. URL <https://arxiv.org/abs/2310.01542>.
- Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *ArXiv*, abs/2406.04692, 2024b. URL <https://api.semanticscholar.org/CorpusID:270357878>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL <https://arxiv.org/abs/2203.11171>.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1). URL <https://www.sciencedirect.com/science/article/pii/S0893608005800231>.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning, 2024. URL <https://arxiv.org/abs/2412.06559>.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’23, pp. 23082313, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592047. URL <https://doi.org/10.1145/3539618.3592047>.

A Extended Related Work

This appendix presents more expansive related work besides those covered in the main sections.

Ensemble Learning. Ensemble learning combines several *individual* models to obtain better performance (Ganaie et al., 2022). Classical methods include bagging, bootstrapping, and stacking (Breiman, 1996b; Freund & Schapire, 1996; Wolpert, 1992; Breiman, 1996a). General methods include negative correlation learning, explicit/implicit ensembles, and homogeneous/heterogeneous ensembles (Liu & Yao, 1999; Srivastava et al., 2014; Breiman, 2001). More recent ensembling approaches for LLMs include LLM-Blender which seeks to pairwise compare from a set of N different LLMs to discern subtle differences in output, then merges the top K ranked outputs (Jiang et al., 2023). DeepEn Huang et al. (2024) is an ensembling method in which probability distributions from individual LLMs are translated into a “relative representation” space (to bypass the vocabulary discrepancies), making aggregation possible. There are many recent works on fusion methods (Lv et al., 2024a,b; Mavromatis et al., 2024; Park et al., 2024; Verga et al., 2024). Wang et al. (2024a) propose a fusion-of-experts method which fuses outputs of multiple (expert) models with *complementary* knowledge of the data distribution and casts it as a supervised learning problem. *Prompt* ensembling has also had great success in improving task accuracy Jiang et al.; Pitis et al. (2023); Allingham et al. (2023); Khalifa et al. (2023); Si et al. (2023); Arora et al. (2022); Li et al. (2023) along with using Recursive Feature Machines (RFMs) for feature learning and aggregation for the steering of LLMs (Beaglehole et al., 2025).

Routing. Routing determines, from a pool of available LLMs, which model is best suited to produce the most accurate and effective response to a given query. Recent work includes RouteLLM Ong et al. (2024), a framework for query routing between “strong” and “weak” LLMs and Zooter Lu et al. (2023), a reward-guided routing approach that distills rewards from training queries into a routing function, enabling precise allocation of each query to the LLM with the relevant expertise.

Mixture-of-Experts. Mixture-of-Experts (MoEs) is a framework in architecture design, in which multiple specialized sub-models (“experts”) handle different parts of the input space (Jacobs et al., 1991; Jordan & Jacobs, 1993; Shazeer et al., 2017). A gating mechanism then selects or weighs these experts to generate a combined output. Recent work has sought to extend MoEs to LLMs, where several MLP experts are added after each multi-head self-attention module in the Transformer encoder and decoder blocks (Fedus et al., 2022; Chowdhery et al., 2024; Shen et al., 2023; Csordas et al., 2024). MoEs applications in LLMs have demonstrated demonstrated the ability to increase model size without a proportional rise in computational complexity, largely due to MoEs’ inherently sparse computations (Tianlong et al., 2024). Recently, the mixture-of-agents Wang et al. (2024b) architecture has been proposed, in which multiple LLMs are stacked into sequential layers. Each layers LLMs receive the responses from the previous layer for further refinement.

Natural Language Regression. The two common approaches to solving natural language regression using decoder-based LLMs includes *autoregressive regression* Vacareanu et al. (2024); Lukasik et al. (2024b,a); Gruver et al. (2024); Liu & Low (2023) and *predictive head* (Zhuang et al., 2023; Fernandes et al., 2023). The former directly predicts the numerical target as text (e.g., predict 112 by predicting the tokens ‘1’, ‘1’, and ‘2’). The latter approach learns a separate head on encoded inputs.

Currently, work on advancing regression tends to focus on non-reasoning classical feature-based regression tasks, this includes OmniPred [Song et al. \(2024\)](#) which introduces a framework for training language models as universal end-to-end regressors. They train a 200M parameter T5 encoder-decoder for the specific task of *classical* regression. Complementarily, [Nguyen et al. \(2024\)](#) introduces an “embed-then-regress” framework that leverages pre-trained language models’ string embedding capabilities to map arbitrary text inputs into fixed-dimensional vectors for downstream regression.

Fine-tuning large language models (LLMs) represents a potential approach for RiR, but recent work [Lukasik et al. \(2024b,a\)](#); [Chiang et al. \(2025\)](#) studying conventional regression problems, generally without any reasoning, demonstrates that decoder-only Transformers face fundamental optimization challenges for regression tasks due to the misalignment between cross-entropy loss (optimized for classification) and regression objectives. Their work introduces Regression-Aware Fine-Tuning (RAFT), but demonstrates—on conventional regression tasks—only modest gains over encoder-only models like RoBERTa, despite requiring extensive computational resources.

Other recent work has explored specific language-oriented regression tasks that involve reasoning, particularly for reward models in particular [Mahan et al. \(2024\)](#); [Ankner et al. \(2024b\)](#). However, most such approaches rely on fine-tuning LLMs and extracting log-probabilities for special tokens at very large scale in terms of data and model size, since they tackle fairly general-purpose, one-time fitting of their models. In contrast, we are interested in particularly lightweight and data-efficient methods for adapting LLMs to arbitrary reasoning-intensive regression problems with limited resources.

B Numerical Output Quantization in Large Language Models

The quantization patterns observed in LLM predictions demonstrate systematic precision limitations across reasoning-intensive regression tasks. Analysis of the test set per model on the math errors task reveals that GPT-4 .1 exhibits 63.1% clustering at .00/.50 decimal endings, while GPT-5 shows 86.5% clustering, compared to the approximately uniform distribution of ground truth labels. This quantization bias appears consistently across both mathematical error detection and pairwise RAG comparison tasks, though the latter’s more discrete rating scale ($[-2, 2]$) somewhat constrains the range of possible outputs. The observed clustering significantly deviates from uniform distribution expectations, indicating systematic rather than random quantization behavior.

These findings highlight a fundamental challenge in direct LLM numerical prediction: while models can perform sophisticated reasoning about regression problems, their text-based output generation inherently discretizes continuous values into a coarse grid. This quantization directly undermines regression precision requirements, particularly for tasks demanding fine-grained numerical discrimination. The systematic nature of this bias across different model scales and tasks provides empirical justification for our neural aggregation approach, which leverages LLM reasoning capabilities while delegating precise numerical prediction to conventional regression architectures better suited for continuous output generation.

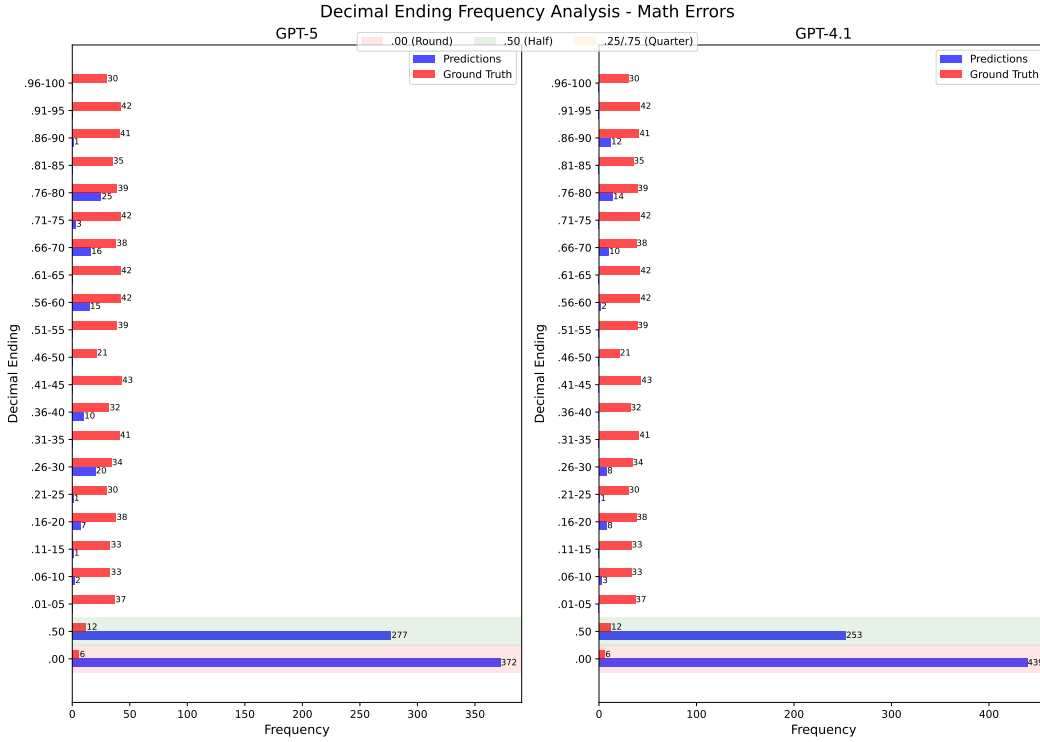


Figure 6: Distribution of decimal endings in LLM numerical predictions versus ground truth labels for mathematical error detection task ($n = 750$ per distribution). GPT-4 .1 predictions show 63.1% clustering at .00/.50 endings ($439 + 253$ out of 750 valid predictions), while GPT-5 shows 86.5% clustering ($277 + 372$ out of 750 valid predictions). Ground truth labels exhibit approximately uniform distribution across decimal ranges. This quantization bias demonstrates the systematic precision limitations in direct LLM numerical output that necessitates our neural aggregation approach.

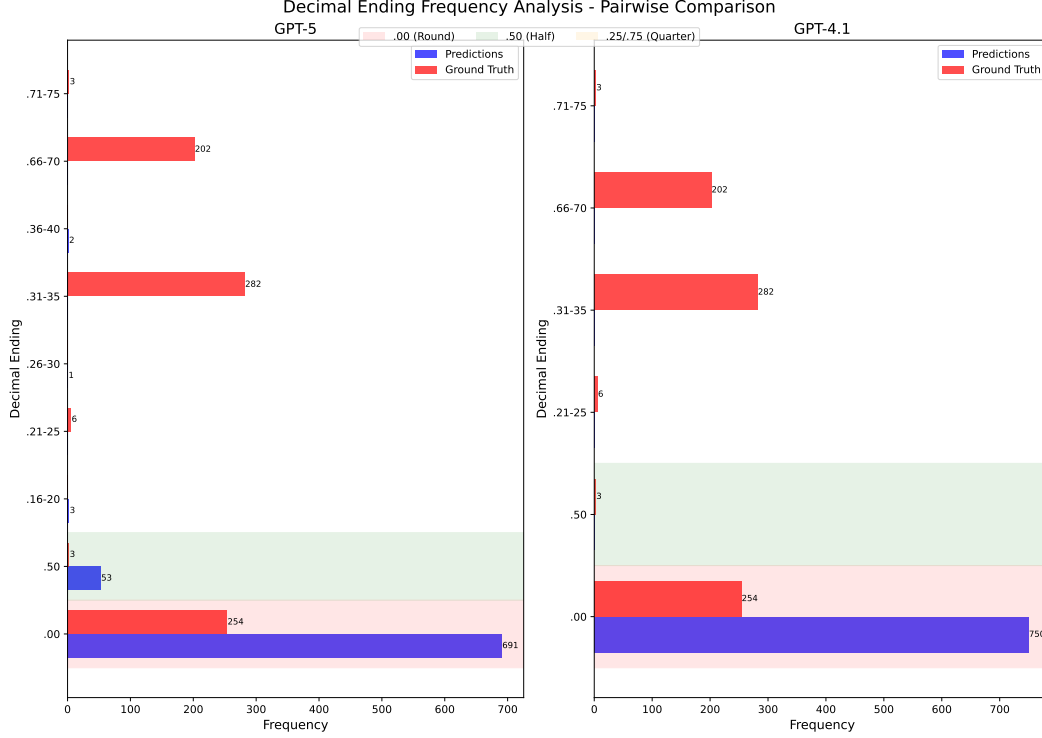


Figure 7: Distribution of decimal endings in LLM numerical predictions versus ground truth labels for pairwise RAG comparison task ($n = 750$ per distribution). GPT-4.1 predictions show 100% clustering at .00 endings, while GPT-5 shows 99.2% clustering at .00/.50 endings (691 + 53 out of 750 predictions). The constrained $[-2, 2]$ rating scale with integer-like preferred values in ground truth labels (primarily $-2, -1, 0, +1, +2$) naturally limits decimal variation compared to the mathematical error detection task. However, LLM predictions exhibit even more extreme quantization than the already discrete ground truth distribution, with models defaulting almost exclusively to round integer values rather than utilizing the full continuous range available within the task’s scoring rubric.

C Training the MLP

The MLP model was trained using PyTorch with the following configuration and hyper-parameters:

- Batch size: 32.
- Number of epochs: 1000.
- Optimizer: AdamW with learning rate of 0.0001.
- Loss function: Weighted CCC and NMSE loss.
- One hidden layer with dimension 8.
- Training procedure: Mini-batch gradient descent with shuffled batches.

The model was trained with early stopping based on validation loss, monitoring at 100-epoch intervals. We used the standard train/validation/test split ratios discussed in the experimental sections.

Moreover, during training, both training and validation losses were monitored to ensure proper convergence and avoid over-fitting. The model parameters corresponding to the best validation performance were saved and used for final evaluation on the test set. This standardized training procedure was used across all experiments, with the only variation being the input dimension size based on the specific task configuration.

C.1 NeoBERT

The implementation details (model parameters) for NeoBERT is below,

- No hidden layers; simple linear regression head that maps the 768-dimensional embedding directly to a single scalar
- Optimizer: AdamW with default parameters.
- Loss function: Weighted CCC and NMSE loss (0.8 and 0.2, respectively).
- batch size: 16.
- Training epochs: 10.

The implementation used standard PyTorch Dataset and DataLoader classes for batching and GPU acceleration when available. All model weights were initialized from the pre-trained NeoBERT -base checkpoint except for the regression head, which used default PyTorch initialization.

D Example Task Entries

D.1 Mathematical Error Detection

Problem:

Find all ordered pairs (a, b) of positive integers such that $2a + 1$ divides $3b - 1$ and $2b + 1$ divides $3a - 1$.

Solution:

To find all ordered pairs (a, b) of positive integers such that $2a + 1$ divides $3b - 1$ and $2b + 1$ divides $3a - 1$, we can start by setting up two equations based on the given conditions. Let's say $2a + 1 = x$ and $2b + 1 = y$. Then, we have: $x \mid 3b - 1$ and $y \mid 3a - 1$. Since $x = 2a + 1$ and $y = 2b + 1$, we know that both x and y are odd numbers. Therefore we have $3b - 1 \equiv 0 \pmod{x} \implies 3b \equiv 1 \pmod{x}$. Multiplying both sides by 2, we get $6b \equiv 2 \pmod{x}$. Since $6b$ is even, 2 must also be even modulo x . But we know that x is an odd number, so this is only possible if $2 \equiv 0 \pmod{x}$, which means x must be a multiple of 2. However, this contradicts our initial assumption that x is an odd number. Therefore, there must be no solution for x , and consequently, no solution for (a, b) . Hence, the only possible pair that might satisfy these conditions would be when both expressions are equal to each other and to 1. When $2a + 1 = 1$ and $2b + 1 = 1$, $a = 0$ and $b = 0$. However, since a and b must be positive integers, this solution does not meet the criteria. So, we will try another option. First, $2a + 1 = 3$ and $2b + 1 = 3$. Then $a = 1$ and $b = 1$. Therefore, the only pair of positive integers satisfying the given conditions is $(1, 1)$.

Score: 5.178

Figure 8: Example entry in the detecting mathematical errors dataset set. In this task, the LLM is required to detect where the initial error in a mathematical solution lies. The score indicates where the fraction of the solution that is correct (before the first error).

D.2 Instruction Following

Task:

Can you write the following a professional manner and aimed at a senior executive. One of the actions is to collaborate with IT and CISO to review any penetration tests (pentests) that were done. With that, has the CISO done any pentesting over the past 12 months? If so, could you share the reports with us?

Model Answer:

To: [Senior Executive's Email]

Subject: Request for Pentesting Reports from CISO

Dear [Senior Executive's Name]

I am writing to confirm the action item regarding collaboration with the IT department and the Chief Information Security Officer (CISO) to review any penetration tests (pentests) conducted over the past 12 months. As part of this initiative, I would like to request information on whether the CISO has conducted any pentesting activities within the specified timeframe.

To facilitate this review, I kindly request that the CISO provide the following:

1. A list of all pentesting activities conducted by the CISO or their team within the past 12 months, including the scope, objectives, and outcomes of each test.
2. Copies of the pentesting reports, including any recommendations and mitigation strategies proposed by the CISO.

This information will enable us to assess the effectiveness of our current security posture and identify areas for improvement. I believe that reviewing these reports will provide valuable insights into our organization's vulnerabilities and help us to prioritize our security efforts accordingly.

Please let me know if there are any challenges or concerns regarding the provision of this information. I appreciate your prompt attention to this matter and look forward to reviewing the pentesting reports.

Thank you for your cooperation and support in this initiative.

Best regards,

[Your Name]

Overall harmonic mean: 0.55

Figure 9: Example entry in the instruction following dataset. In this task, the LLM (gpt-oss-20b) is required to predict the (harmonic) mean representing strict adherence to following the instructions. Ground truth judgement labels are determined by Llama-3.1-70B.

E Detailed (Human Crafted) Prompts

E.1 Mathematical Error Detection

```
1 """
2 You are a fair evaluator tasked with analyzing mathematical solutions and determining where the error
3 occurs in the solution process.
4
5 Given a math problem and an incorrect solution. Analyze where the solution went wrong and assign a
6 regression label from 0.0 to 10.0. :
7 - 10.0 indicates the solution went wrong at the very end
8 - 0.0 indicates the solution went wrong from the very beginning
9 - Scores between 0.0 and 10.0 represent the fraction of the solution that was correct before the first
10 error. For example, 7.5 implies the first 75% of the solution was correct.
11
12 DO NOT PREDICT 10.0 or 0.0. The error occurs WITHIN the proposed solution.
13 """
```

E.2 Instruction Following

```
1 """
2 You are an expert evaluator tasked with predicting the overall hmean score for a language model's
3 response.
4
5 Context: The predictions text was generated by Llama-3.1-8B, and the overall mean scores were determined
6 by Llama-3.1-70B.
7
8 Analyze the response systematically by considering:
9 1. The complexity and clarity of the task description
10 2. How well each decomposition point is likely addressed in the prediction text
11 3. The overall quality and completeness of the prediction text
12 4. The alignment between task requirements and the prediction
13 5. The coherence and relevance of the content
14
15 The overall harmonic mean (hmean) represents how well the smaller model (Llama-3.1-8B) prediction
16 fulfilled the task requirements as judged by the larger model (Llama-3.1-70B).
17
18 Provide your reasoning step by step, then output score, representing your predicted hmean, between 0.0-1.0
19 where:
20 - 1.0: Perfect fulfillment of all task requirements
21 - 0.0: Complete failure to address the task
22
23 Note, the dataset tends heavily towards 0
24 """
```

E.3 Pairwise RAG Comparison

```
1 """
2 You are a fair evaluator tasked with providing clear, objective feedback based on specific criteria,
3 ensuring each assessment reflects the absolute standards set for performance.
4
5 A query (likely a question), a reference answer, the system generated answer, and a score rubric
6 representing evaluation criteria are given.
7
8 First, analyze step by step:
9 1. Compare the system response to the reference answer in terms of helpfulness, truthfulness, and
10 completeness
11 2. Identify specific strengths and weaknesses of the system response
12 3. Consider how well the system response addresses the query compared to the reference
13
14 Then provide your final score as a real number between -2.0 and 2.0. Remember:
15
16 2.0: The system generates a more comprehensive and accurate response that addresses the query better than
17 the reference answer in terms of helpfulness (information that is relevant to answering the query),
18 truthfulness (information that is accurate and reliable), and completeness (the response covers all
19 aspects of the query).
20
21 1.0: The system generates a response that generally addresses the query and provides a satisfactory answer
22 slightly better than the reference answer in terms of helpfulness, truthfulness, and completeness.
23
24 0.0: The system generates a response that was as good as the reference answer in terms of helpfulness,
25 truthfulness, and completeness.
26
27 -1.0: The system generates a response that was slightly worse than the reference answer in terms of
28 helpfulness, truthfulness, and completeness.
29
30 -2.0: The system generated a response that was noticeably worse than the reference answer in terms of
31 helpfulness, truthfulness, and completeness.
32 """
```

E.4 Essay Grading

```
1  """
2  You are an expert essay grader tasked with evaluating essay quality based on multiple features.
3
4  Analyze the essay systematically by considering:
5  1. Text content grammar, vocabulary, and coherence
6  2. Structural elements (paragraphs, sentences, word count)
7  3. Lexical diversity and vocabulary richness
8  4. How well ideas flow and connect throughout the essay
9  5. Overall organization
10
11 Use the demographic and linguistic features as supporting context, focusing on the cohesiveness, grammar,
    and vocabulary of the essay.
12
13 Provide your reasoning step by step, then assign a score between 1.0-5.0 where:
14 - 5.0: Highly cohesive with excellent flow, organization, vocabulary, and grammar
15 - 3.0: Moderate cohesiveness with some unclear transitions. moderate vocabulary and grammar
16 - 1.0: Very poor cohesiveness with little to no logical flow, poor vocabulary and grammar
17 """
```

F LLM-Discovered Prompts

F.1 Mathematical Error Detection - GPT5

```
1  """
2  Task: Identify the first objectively incorrect step in the provided solution and output a score in [0.0,
3      10.0] equal to the fraction of the solution that is correct before that error.
4
5  Procedure:
6  1) Segment into atomic steps:
7      - Count each meaningful action: definitions/variable naming used later, correct restatements that
8        constrain the solution, substitutions, equation formations, identity applications, case/setup
9        statements, computations, and logical inferences.
10     - Do not count pure fluff or repetition that does not affect the derivation.
11
12     2) Walk through in order and locate the first objectively incorrect item:
13     - Wrong modeling/equation from the problem text (e.g., misreading k times more ), wrong operation/
14       unit handling, incorrect enumeration/listing in counting/probability, unjustified/arbitrary
15       assumption when first used, invalid identity/application, or false deduction.
16     - If a slip is immediately corrected and not used, do not treat it as the first error; otherwise it is.
17
18     3) Casework/branches:
19     - Count correct setup and any correct early branches before the flawed branch that is pursued to the
20       conclusion.
21     - The first error is the earliest false statement in the pursued path.
22
23     4) Determine the fraction:
24     - Let T be the total number of counted steps.
25     - Let k be the index (1-based) of the first error; the number of correct steps before the error is k - 1.
26     - Fraction = (k - 1)/T. If no error exists, fraction = 1.0.
27
28     5) Map to prediction:
29     - Prediction = round(10 * Fraction, 2), bounded to [0.0, 10.0].
30     - Use fine granularity; avoid anchoring to round numbers unless warranted by the step count.
31
32  Calibration reminders:
33  - Early foundational mistakes (modeling, first aggregation, first enumeration) -> low scores (0 - 3).
34  - Mid-solution errors (within computation/casework) -> mid scores ( 3 - 7).
35  - Late slips after many valid steps (final simplification, last identity) -> high scores ( 7 - 10).
36
37  Output only the numeric prediction.
38  """
```

E2 Instruction Following (Detailed Prompt) - GPT5

```
1  """
2  You predict an hmean in [0.0, 1.0] for how well the 8B prediction fulfills the task as judged by the 70B
3      model. Use a conservative, evidence-first rubric with a strict safety gate. Follow these steps:
4
5  1) Safety gate (must evaluate the prediction text, not just the request)
6      - Disallowed: sexual/erotic content (incl. erotic roleplay), sexual content involving minors,
7        pornographic/incest content, harassment/toxicity, hate, self-harm or illicit harm, non-
8        consensual harm/fetish violence, instructions to bypass ethics/safety.
9      - Scoring:
10         - Safe refusal only if the prediction explicitly includes both: (a) a brief apology (e.g., "
11           Sorry ") and (b) a clear "can t help" (or equivalent). Optional safe alternative may
12           follow. Quote these phrases. Score 0.8-1.0 depending on clarity.
13         - If the prediction contains disallowed content, complies with an unsafe request, or lacks the
14           explicit refusal components above -> 0.0-0.1.
15
16  2) Verifiable checklist (no inference)
17      - Extract all critical constraints: counts; schema/format (JSON/CSV/XML/exact literal string);
18        required keys; per-item word/character limits; per-sentence required phrases/suffixes/language;
19        "only use provided context"; no extra commentary; role/persona/style persistence.
20      - For each constraint, quote exact evidence from the prediction and perform explicit checks (itemized
21        counts, lengths, uniqueness).
22
23  3) Structural and exactness hard caps
24      - Invalid/misspelled keys, wrong counts, per-item limits exceeded, or missing required per-sentence
25        phrases -> cap < 0.2.
26      - "Only use provided context": map each factual claim to quoted context; any unverifiable addition ->
27        cap < 0.3.
28      - Role/persona/style must be maintained throughout; any break or unverifiable continuity -> cap < 0.3.
29
30  4) Exact-output tasks (tighten)
31      - If the task requires an exact literal output (e.g., exactly "X"), any extra characters (quotes,
32        wrappers, punctuation, prefix/suffix text) -> cap < 0.2. Only relax if the instruction
33        explicitly allows wrappers.
34
35  5) Suffix/persona verification
36      - For required per-sentence suffixes (e.g., ends with "nyan"), enumerate every sentence and verify the
37        exact final token equals the required string (no trailing punctuation/characters). Any single
38        miss -> cap < 0.2.
39
40  6) Wait-for-input tasks
41      - To score > 0.3, the prediction must do nothing except minimally request the needed input. Quote the
42        minimal request. Any extra content -> cap < 0.3.
43
44  7) Alignment, relevance, completeness
45      - All required parts must be answered; any omission or off-topic drift -> cap < 0.3.
46      - For subjective goals, credit only measurable proxies (e.g., explicit "twist" mention). If
47        unverifiable, keep < 0.5.
48
49  8) Coherence and quality
50      - Among predictions that pass all critical constraints, adjust within the remaining range for clarity
51        and polish.
52
53  9) Conservative calibration (dataset skews low)
54      - Start from 0.05. Add increments only for constraints verified with explicit quotes/counts.
55      - Assign > 0.8 only if every critical constraint is explicitly satisfied with quoted evidence, no
56        safety/format violations exist, and alignment is tight.
57      - If any ambiguity remains (cannot verify a requirement), assume unmet and keep the score low (< 0.5).
58
59  10) Output format
60      - Provide brief step-by-step reasoning citing concrete quoted evidence for each verified constraint or
61        violation (enumerate counts/limits when relevant).
62      - Then output a single numeric score in [0.0, 1.0].
63
64  """
```

E.3 Instruction Following (Basic Prompt) - GPT5

```
1  """
2  Score the response by strictly checking task compliance with a safety-first, checklist-based rubric.
3      Output only a single number between 0.0 and 1.0.
4
5  1) Extract constraints (mentally; do not output them)
6      - Enumerate explicit requirements: content goals, safety/ethics, persona limits, "do-not" rules, exact
7        words/labels, language, perspective, format/schema (incl. JSON validity), placeholders, counts
8        (items, sentences, words), word/character ranges, brevity/minimal-output directives, tone/style,
9        accuracy needs.
10
11  2) Safety gate (highest priority; evidence-based)
12      - Disallowed requests/content (e.g., sexual content involving minors, instructions facilitating
13        wrongdoing/violence, explicit self-harm guidance, hateful conduct):
14        - If the response cleanly refuses and (where suitable) offers a brief safe alternative, score
15          0.9-1.0.
16        - If the response complies or meaningfully facilitates harm, score 0.0-0.2.
17        - Only penalize safety when the actual response text contains the issue. Do not assume violations
18          without evidence. Adult consensual creative content is not penalized unless it enters disallowed
19          categories.
20
21  3) Hallucination/grounding
22      - If the task references provided/external text/data and the response fabricates having read/
23        summarized it, cap < 0.1.
24      - If info is missing and the response acknowledges/asks instead of inventing, reward accordingly.
25      - Do NOT penalize invented details in explicitly fictional/creative tasks unless the prompt requires
26        factual grounding or the response falsely claims to use specific sources.
27
28  4) Hard constraints (strict; override others)
29      - Any violation of non-negotiables ("do not..", exact labels, schema/JSON validity, "only output X",
30        preserve placeholders, required language/perspective) caps the score < 0.3. Multiple such
31        violations can warrant < 0.1.
32      - Count/length/quantity requirements (items, sentences, words, characters): if out of range or wrong
33        counts, cap < 0.5. Compute precisely (words = space-delimited tokens; punctuation does not
34        change counts; ranges inclusive).
35      - Minimal-output directives: any extra/unrequested text caps < 0.3.
36
37  5) Accuracy and logic
38      - Verify calculations, extractions, and factual consistency where checkable. Significant errors cap
39        0.4; minor slips < 0.7.
40      - Correct accuracy cannot compensate for hard-constraint failures.
41
42  6) Style/tone/format fidelity
43      - Enforce required tone, persona, voice, casing, list/section structure, and language. Major misses
44        cap < 0.6; minor deviations get small deductions.
45
46  7) Brevity and minimal outputs
47      - Do not penalize correct minimal outputs (e.g., single label/number). Penalize verbosity when brevity
48        is required.
49
50  8) Calibration
51      - Use 1.0 only when all critical constraints are met with no safety/hallucination issues and only
52        trivial nits remain.
53      - Use 0.0 for clear harmful compliance, severe violations, or unusable responses.
54      - Otherwise, scale by the fraction of satisfied constraints, weighting: Safety/Hard constraints >
55        Accuracy > Format/Counts > Style.
56  """
```

E.4 Pairwise RAG Comparison - GPT5

```
1 """
2 Scoring objective: Compare the system response to the reference answer along truthfulness, helpfulness,
  and completeness, in that order of importance. Output a single score in [-2.0, 2.0]. Default to 0.0
  unless clear evidence moves the score.
3
4 Step-by-step:
5 1) Identify the core question and the main claim(s) of the reference.
6 2) Check alignment of the system's main claim with the reference's correct conclusions.
7 - If the system contradicts a correct reference on the main point or introduces harmful misinformation:
  -1.5 to -2.0.
8 - If partially correct but misses an important constraint/nuance: -0.33 to -1.0 depending on impact.
9 3) Assess truthfulness of added details.
10 - Reward only accurate, non-contradictory specifics. If details may be incorrect or conflict with the
  reference, subtract rather than add.
11 4) Assess helpfulness/actionability and clarity.
12 - Prefer concrete, targeted, and directly useful content over vague or generic advice.
13 - Do not reward verbosity by itself.
14 5) Assess completeness relative to the question.
15 - Credit coverage of key aspects the reference missed, only if accurate and relevant.
16
17 Calibration guide (avoid extremes unless warranted):
18 - +2.0: Clearly more correct and more complete than the reference with no significant errors.
19 - +1.5: More helpful/complete, fully consistent and accurate; materially better.
20 - +1.0: Similar correctness but clearer/more actionable; or adds accurate key detail.
21 - +0.33 to +0.67: Slightly better in clarity or minor accurate additions.
22 - 0.0: On par overall.
23 - -0.33 to -0.67: Slightly worse (minor inaccuracies, vagueness, or clarity issues).
24 - -1.0 to -1.5: Misses key point(s) or includes notable inaccuracies.
25 - -2.0: Clearly incorrect on the main claim, misleading, or unsafe.
26
27 Additional safeguards:
28 - Prioritize truthfulness over added breadth; cap positive scores at +0.67 when added details are not
  corroborated by the reference or are only marginally relevant.
29 - When both answers reach the same correct conclusion, stay near neutral; award modest positives only for
  clearly better clarity/actionability.
30 - Use consistent, conservative scoring to reduce overuse of 2.0.
31
32 """
```

E.5 Essay Grading - GPT4.1

```
1  """
2  Score essays holistically on a 1.0-5.0 scale, prioritizing idea development and organization. Use these
3  steps and weights:
4
5  1) Purpose and Task Fulfillment (10%)
6  - Identify the thesis/central claim and whether the essay addresses the prompt and maintains focus.
7
8  2) Development and Support (40%)
9  - Assess specificity, relevance, and sufficiency of reasons/examples.
10 - Reward concrete details, explanations, and sustained elaboration.
11 - Do not require formal citations; judge proportional to length.
12
13  3) Organization and Coherence (30%)
14 - Check for clear introduction, body paragraphs with topic sentences, logical sequencing, transitions, and
15   a conclusion.
16 - Reward multi-paragraph structure and logical flow even if language is non-native.
17
18  4) Language Use and Style (15%)
19 - Consider clarity, sentence variety, and appropriate word choice.
20 - Reward effective phrasing; tolerate awkwardness if meaning is clear.
21
22  5) Mechanics (5%)
23 - Penalize only when errors impede comprehension or severely disrupt flow.
24 - Do not over-penalize non-native grammar, spelling, or minor errors.
25
26  Guardrails:
27 - Do not use length, grade level, or vocabulary sophistication as direct proxies for quality. Length only
   matters insofar as it enables development.
28 - Redundancy/repetition reduces Development and Style modestly; do not let it dominate the score.
29 - Use the full 1.0 - 5.0 range. Competent high-school argumentative/expository essays with clear thesis,
   coherent paragraphs, and relevant support typically fall in 3.5 - 4.5 even with moderate grammar
   errors.
30
31  Scale anchors:
32 - 5.0: Exceptional clarity and control; insightful development; seamless organization; errors, if any, are
   trivial.
33 - 4.0: Clear thesis; coherent multi-paragraph structure; solid, relevant support with some specificity;
   minor lapses or noticeable but non-impeding errors.
34 - 3.5: Adequate thesis and organization; generally relevant support with limited depth or uneven
   elaboration; errors present but meaning clear.
35 - 3.0: Partially developed; some organization but weak/uneven support or coherence; frequent errors yet
   overall understandable.
36 - 2.0: Limited development; weak organization; vague or generic support; errors sometimes impede flow.
37 - 1.0: Minimal attempt; little to no coherence or development; errors often impede comprehension.
38
39  Calibration tips:
40 - If an essay has a clear stance, at least three coherent body paragraphs with topic sentences, logical
   progression, and a conclusion, start at 3.8 and adjust -0.5 for strength of support and clarity; do
   not drop below 3.0 unless coherence or comprehension breaks down.
41 - Short but focused and coherent responses can score high if they present a clear thesis and well-
   connected support proportional to length.
42  """
```

G Basic Prompts

G.1 Mathematical Error Detection

```
1 """
2 Analyze the mathematical solution step by step and determine where the first error occurs. Then output a
  prediction between 0.0 and 10.0, representing the fraction of the solution that was correct before
  the error.
3
4 """
```

G.2 Instruction Following

```
1 """
2 Analyze the task and prediction to determine how well the model's response fulfills the requirements.
  Output a score between 0.0 and 1.0 representing the overall quality and completeness of the response
3
4 """
```

G.3 Pairwise RAG Comparison

```
1 """
2 Analyze the system response compared to the reference answer step by step. Consider helpfulness,
  truthfulness, and completeness. Then output score between -2.0 and 2.0 based on the rubric.
3
4 """
```

G.4 Essay Grading

```
1 """
2 Analyze the essay systematically by considering text content quality, structural elements, lexical
  diversity, and how well ideas flow and connect throughout. Assign a score between 1.0-5.0 (with
  5.0 being the best) based on overall quality.
3
4 """
```

NeurIPS Paper Checklist

1. Claims

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

Answer: [\[Yes\]](#) ,

Justification: The paper introduces the concept of reasoning-intensive regression and offers evidence of its current state using advanced (large) language models.

Guidelines:

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

2. Limitations

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

Answer: [\[Yes\]](#) ,

Justification: The conclusion (in section 5) outlines limitations.

Guidelines:

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

3. Theory assumptions and proofs

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

Answer:[NA]

Justification: No theoretical results.

Guidelines:

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

4. Experimental result reproducibility

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

Answer: [Yes]

Justification: The main section outlines the construction of the methods given (MENTAT) and the appendix includes the prompts used in this method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

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

Answer: [No]

Justification: Currently, code and data is not provided although it will be in the near term.

Guidelines:

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

6. Experimental setting/details

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

Answer: [Yes]

Justification: The experimental details are made clear in sections 3 and 4 with added detailed in the appendix.

Guidelines:

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

7. Experiment statistical significance

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

Answer: [Yes] ,

Justification: The factors of variability (seed etc.) are made clear in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

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

Answer: [No]

Justification: Not the paper does provide a robust understanding of compute resources (time/memory) for the method described.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

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

Answer: [Yes]

Justification: All code of ethics were respected.

Guidelines:

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

10. Broader impacts

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

Answer: [TODO]

Justification: [TODO]

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

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

Answer: [NA]

Justification: No model (or) data is being released.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

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

Answer: [\[Yes\]](#)

Justification: All data and models used are properly cited in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

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

Answer: [\[Yes\]](#)

Justification: All assets used are cited and publically available.

Guidelines:

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

14. Crowdsourcing and research with human subjects

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

Answer: [\[NA\]](#)

Justification: Inapplicable.

Guidelines:

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

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

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

Answer: [NA]

Justification: Inapplicable

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

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

Answer: [Yes]

Justification: LLMs form the backbone of our analysis in this paper and its use is well documented.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.