REINFORCEMENT LEARNING OVER PREDICTIVE DISTRIBUTIONS FOR LLM REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have emerged as flexible regressors capable of predicting real-valued quantities from heterogeneous inputs. Yet most LLM regression objectives optimize predictions independently, often yielding poor calibration. We introduce *Distribution-Aware Reward* (DAR), an on-policy reinforcement learning objective that instead jointly evaluates the empirical predictive distribution formed by multiple predictions for the same input. To translate this distribution-level objective into rollout-level rewards, we assign each prediction credit based on its leave-one-out contribution to the quality of the overall predictive distribution. This encourages predictions that are well-centered and appropriately dispersed around the target. We evaluate on three regression settings: a synthetic task probing interpolation and extrapolation, and two real-world scientific tasks involving code and molecular data. Across tasks, DAR produces better-calibrated uncertainty estimates while consistently reducing prediction error and improving ranking quality over supervised fine-tuning and pointwise reinforcement learning. Together, these results highlight the benefits of distribution-aware training for LLM regression.

1 INTRODUCTION

Many prediction problems in science and engineering require estimating real-valued quantities from heterogeneous inputs, including experimental protocols, source code, and molecular representations (Wu et al., 2018; Park et al., 2025; Akhauri et al., 2025). Large language models (LLMs) provide a flexible approach to such problems when inputs can be represented as sequences (Requeima et al., 2024; Song et al., 2024; Tang et al., 2024). Recent work has improved LLM regression through in-context learning (Vacareanu et al., 2024), supervised fine-tuning (SFT) (Song et al., 2024; Lukasik et al., 2025), and reinforcement learning (RL) with numeric rewards (Chen et al., 2025).

Beyond producing a single prediction, repeated sampling from an LLM naturally induces a predictive distribution over possible values. This distribution can be aggregated for point prediction while also providing information about predictive uncertainty. Yet existing training objectives largely ignore this distributional structure. They typically score each decoded prediction independently according to its distance from the target (Chen et al., 2025), treating rollouts as separate predictions rather than samples from a shared predictive distribution (Figure 1 (left)). We refer to these as *pointwise objectives*. While effective for encouraging individual predictions toward the target, pointwise squared-error rewards also penalize variation across samples, which can suppress predictive variation even when uncertainty is warranted. Ideally, an LLM regressor should produce both accurate aggregate predictions and predictive distributions that support calibrated uncertainty estimates (Gneiting et al., 2007; Gneiting & Raftery, 2007; Levi et al., 2022).

We therefore introduce *Distribution-Aware Reward* (DAR), an RL objective that optimizes the predictive distribution formed by multiple decoded samples. The key idea is to shift the unit of optimization from individual predictions to the distribution they collectively form. As illustrated in Figure 1 (middle), rather than rewarding each rollout solely for being close to the target, DAR assigns credit based on how each rollout contributes to the quality of the predictive distribution as a whole.

DAR treats multiple rollouts as an empirical predictive distribution rather than scoring each one in isolation. We measure the quality of this distribution using the Continuous Ranked Probability Score (CRPS), a scoring rule that rewards distributions that place probability mass near the target while maintaining appropriate spread (Matheson & Winkler, 1976; Gneiting & Raftery, 2007). Thus, DAR

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

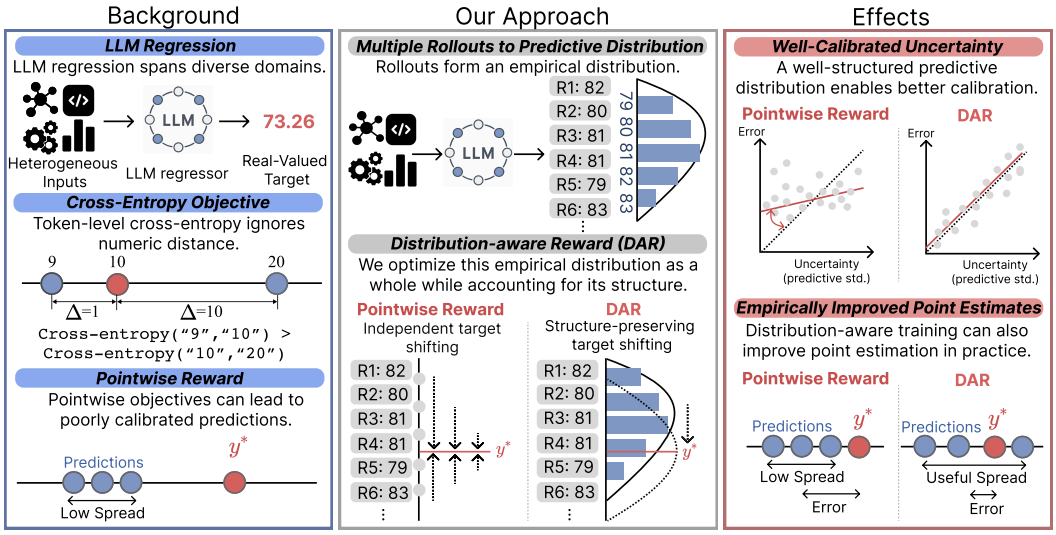


Figure 1: **Distribution-Aware Reward. Left:** Standard objectives optimize individual predictions without explicitly modeling their distribution. **Middle:** Multiple LLM rollouts form an empirical predictive distribution. Pointwise rewards score these rollouts independently, whereas DAR rewards each rollout according to its contribution to the distribution as a whole, encouraging accurate predictions while preserving useful spread. **Right:** Distribution-aware training produces better-calibrated predictive distributions, while empirically also yielding stronger aggregate point estimates.

does not encourage dispersion for its own sake, but balances proximity to the target with dispersion across predictions. We then use leave-one-out credit assignment, rewarding each rollout according to how much its inclusion improves the overall distribution. This provides a joint training signal that accounts for interactions among sampled predictions.

We evaluate DAR on three regression settings: a synthetic task with multimodal target distributions (Fishkov et al., 2025), code performance prediction (Akhauri et al., 2025), and molecular property prediction (Wu et al., 2018). Across these tasks, DAR produces predictive distributions with improved calibration and uncertainty estimates that better align with prediction error. We further analyze the learned predictive distributions and rollout behavior during training to characterize how distribution-aware optimization changes the model's sampling behavior.

Importantly, these distributional benefits do not come at the expense of predictive performance. DAR consistently reduces prediction error and improves rank correlation over supervised fine-tuning and pointwise RL baselines. The gains are particularly strong for code performance prediction, where DAR improves Spearman correlation by up to 6 points. It also achieves the strongest rank correlation on the synthetic task and remains competitive with specialized graph and 3D molecular models for molecular property prediction, despite using only SMILES, a text-based molecular representation. These improvements come with negligible computational overhead over standard RL training. Together, our results show that optimizing LLM predictions at the distribution level can support calibrated uncertainty while also improving aggregate point prediction and ranking.

2 DAR: Distribution-Aware Reward for LLM Regression

We introduce DAR, a distribution-aware reward for on-policy RL that evaluates multiple predictions for the same input jointly rather than independently. DAR can be used within standard rollout-based RL frameworks such as GRPO (Shao et al., 2024) by replacing the pointwise reward with a distribution-aware reward assigned to each rollout. We first review standard training objectives for LLM regression and then derive the distribution-level objective and its rollout-level credit assignment.

2.1 Background: Training Large Language Models for Regression

LLMs have recently emerged as flexible regressors capable of handling heterogeneous inputs. In this setting, given an input x and target y, an autoregressive language model π_θ generates a sequence of output tokens z. For regression, this output is mapped to a scalar prediction $p \in \mathbb{R}$ using a deterministic parsing function g, such that $p = g(z)$. A common approach to training LLMs for regression is supervised fine-tuning (SFT) (Song et al., 2024; Akhauri et al., 2025; Lukasik et al., 2025). Given an input–target pair (x, y), we represent the numeric target y as a sequence $z^\star$, corresponding to its string representation. SFT then minimizes the negative log-likelihood of this target sequence:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\log \pi_\theta(z^\star \mid x). \tag{1}$$

Although effective, this objective optimizes token-level prediction of the target representation rather than the quality of the resulting numeric prediction. One alternative is to formulate regression as classification over a discretized set of numeric values (Lukasik et al., 2025). The model predicts a probability distribution over these values, which is then aggregated to produce a continuous prediction. While this formulation retains standard cross-entropy training, it requires the numeric output space to be discretized in advance, which can be restrictive when target ranges vary substantially across regression tasks.

Reinforcement learning (RL) provides another alternative by defining rewards directly on decoded numeric predictions. Unlike SFT, RL can sample multiple outputs for the same input and assign rewards based on their resulting numeric values. For the same input-target pair (x, y), we sample K textual rollouts from the current policy,

$$z_k \sim \pi_\theta(\cdot \mid x), \qquad k = 1, \ldots, K, \tag{2}$$

$$p_k = g(z_k), \tag{3}$$

where $p_k \in \mathbb{R}$ is the numeric prediction extracted from the k-th rollout. This yields a set of predictions $\{p_k\}_{k=1}^K$ for the same target y. A standard pointwise reward evaluates each rollout independently. For example, a mean squared error (MSE)-based reward (Koa et al., 2025; Chen et al., 2025) can be expressed as:

$$R_k^{\mathrm{MSE}} = -(p_k - y)^2. \tag{4}$$

This directly optimizes the decoded numeric output rather than its token-level representation, but evaluates each rollout independently of the other predictions sampled for the same input. As a result, pointwise rewards do not account for the distribution collectively formed by multiple rollouts. Our goal is not to encourage greater dispersion for its own sake, but to optimize the quality of this predictive distribution by accounting for both accuracy and uncertainty. These considerations motivate our design of DAR.

2.2 Rewarding Predictive Distribution Quality

To move beyond pointwise rewards, we propose a distribution-aware reward (DAR) that jointly evaluates the K rollouts as an empirical predictive distribution rather than rewarding each prediction in isolation. We seek an objective that captures both proximity to the target and the dispersion of the sampled predictions, without imposing a parametric assumption on their distribution. The Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976; Gneiting & Raftery, 2007) satisfies these properties as a proper scoring rule that evaluates a predictive distribution against a realized numerical outcome, with its expected score optimized when the predictive distribution matches the true outcome distribution. Since CRPS is conventionally minimized, we define the distribution-level reward over the entire rollout set $\{p_k\}_{k=1}^K$ using its negative empirical form:

$$R^{\mathrm{dist}} = \underbrace{-\frac{1}{K}\sum_{k=1}^{K}|p_k - y|}_{\text{distance to target}} + \underbrace{\frac{1}{2K^2}\sum_{k=1}^{K}\sum_{\ell=1}^{K}|p_k - p_\ell|}_{\text{spread among predictions}}. \tag{5}$$

The first term is the negative mean absolute error (MAE) of the sampled predictions, rewarding proximity to the target. In isolation, it encourages every rollout to move toward y, irrespective of the other $K-1$ predictions, suppressing variation across the sampled distribution. The second term counterbalances this tendency by rewarding pairwise separation among the sampled predictions. Importantly, the second term does not by itself encourage predictions to concentrate around the target. Rather, the two terms jointly trade off proximity to the observed target against dispersion among predictions, discouraging both excessive concentration and excessive spread.

All K predictions for the same input x jointly determine R^{dist}. To use this distribution-level objective for on-policy RL optimization, we need to assign a scalar reward to each rollout. The key challenge is thus to attribute the quality of the sampled distribution back to the individual prediction that constitutes it. We address this credit-assignment problem using a leave-one-out marginal contribution. Let R^{dist}_{-k} denote the distribution-level reward computed after removing the kth rollout p_k from the sampled set. We define the rollout-level DAR reward as follows, using R^{dist} and R^{dist}_{-k}, both computed over sets of sampled rollouts:

$$R^{\text{DAR}}_k = R^{\text{dist}} - R^{\text{dist}}_{-k}. \tag{6}$$

Intuitively, if removing p_k lowers the distribution-level reward, then $R^{\text{DAR}}_k > 0$, indicating that the rollout improves the overall predictive distribution. If removing it increases the reward, then $R^{\text{DAR}}_k < 0$, indicating that this particular rollout degrades the predictive distribution. Although this leave-one-out construction makes each rollout's reward depend on the other sampled predictions, it introduces no additional bias into the policy-gradient estimator; we provide a formal analysis in Appendix C.

Motivating Example: MSE vs. DAR. We illustrate that DAR can upweight a rollout with larger pointwise error when it improves the *collective* predictive distribution. Consider a single input with target $y = 5$ and three rollouts producing predictions $[3,\ 4,\ 8]$. A pointwise MSE-based reward scores rollouts independently and therefore favors 4 (lowest error) and 3, while penalizing 8, encouraging concentration around near-target values.

DAR instead scores the *set* of rollouts via CRPS and assigns credit by marginal contribution. Here, 3 and 4 lie on the same side of the target, while 8 provides a prediction on the other side. Although 8 has larger pointwise error, removing it worsens CRPS more than removing 3. DAR therefore assigns higher marginal credit to 8 than to the more redundant prediction 3.

3 Experimental Setup

We evaluate whether DAR improves predictive performance and uncertainty estimation across diverse regression settings. Our evaluation spans three regression settings with different input modalities and prediction challenges: a synthetic regression task designed to probe interpolation and extrapolation, code performance prediction, and molecular property prediction.

3.1 Tasks

We select three regression settings that span distinct input representations and prediction challenges, including both controlled and established real-world benchmarks. Together, they allow us to evaluate DAR in a controlled distributional setting as well as on real-world scientific regression problems involving code and molecular representations. Further details are provided in Appendix D.

Synthetic Distributional Regression. We use the synthetic regression benchmark introduced by Fishkov et al. (2025), where the target is sampled from a known heteroscedastic two-component Gaussian mixture conditioned on a scalar input x. This controlled setting provides access to the ground-truth conditional distribution, making it useful for evaluating both predictive accuracy and uncertainty behavior under interpolation and extrapolation. We train on 1,200 samples from the ground-truth process and evaluate on disjoint test regions with 600 samples covering both interpolation and extrapolation. We report RMSE and Spearman correlation.

Coding Performance Prediction. We evaluate on two code performance prediction benchmarks introduced by Akhauri et al. (2025): Triton kernel latency prediction (KBSS) and Python peak-memory prediction (APPS). These tasks require predicting continuous execution properties directly from source code and provide established benchmarks for evaluating LLM-based regression on complex sequential inputs. We use the dataset splits from prior work, with 6K/1K/1K examples for training, validation, and testing, and report Spearman correlation and MAE.

Molecular Property Prediction. We evaluate molecular property prediction on three established MoleculeNet benchmarks (Wu et al., 2018): ESOL, FreeSolv, and Lipophilicity. Each benchmark defines a separate regression task over molecules represented as SMILES strings (Weininger, 1988): ESOL predicts aqueous solubility, FreeSolv predicts hydration free energy, and Lipophilicity predicts a logD-based measure of hydrophobicity. These datasets are widely used benchmarks for molecular property prediction and allow us to compare LLM-based regression against established molecular models that use graph or 3D representations. We use the standard scaffold splits following Zhou et al. (2023) and report RMSE.

3.2 Implementation Details

We follow a standard on-policy RL fine-tuning recipe for sequence models (Schulman et al., 2017; Ouyang et al., 2022). Unless otherwise stated, we use a learning rate of 1×10^{-6}, train for up to 300 steps, and sample $K = 12$ rollouts per input with temperature 1.0 for exploration. All RL and SFT experiments are implemented in `verl` (Sheng et al., 2024), with KL regularization applied in the loss against the fixed reference policy, initialized from the same checkpoint used to start RL training. We select the checkpoint with the best validation performance and report its test results.

For the synthetic experiment, we use Qwen2.5-0.5B-Instruct (Qwen Team, 2024); for real regression tasks, we use Qwen3-4B-Instruct as the default backbone (Yang et al., 2025). We disable explicit reasoning by default for efficiency and analyze its effect in Appendix H. During training, each rollout is parsed into a scalar prediction using a strict output format. Invalid rollouts receive method-specific penalties, such as the minimum batch reward for the negative-MSE baseline. At evaluation time, we generate 32 samples and average them as the final prediction, following standard LLM regression practice (Lukasik et al., 2024; 2025; Akhauri et al., 2025). We repeat evaluation with 5 random seeds and report averaged metrics (Chen et al., 2025).

3.3 Baselines

We compare against baselines that cover the main ways LLMs can be used for regression. At the prompting level, we evaluate both proprietary frontier LLMs and the open-weight base model in *ICL* settings (Vacareanu et al., 2024), capturing off-the-shelf inference without task-specific training. We also evaluate domain-specific LLMs, such as ChemLLM (Zhang et al., 2024; Yu et al., 2024), in the zero-shot setting.

At the training level, we consider a value-head regression baseline trained with MSE (Tang et al., 2024) and an SFT baseline that fine-tunes the model to generate a single numeric output (Lukasik et al., 2025). Finally, in the on-policy RL setting, we compare conventional pointwise reward optimization based on negative MSE (Chen et al., 2025) against DAR, which directly optimizes the predictive distribution induced by multiple rollouts.

4 Experimental Results

We evaluate DAR on point estimation and ranking across the three regression settings. We analyze uncertainty estimation and calibration separately in Section 5.1.

4.1 Synthetic Distributional Regression

Figure 2 compares SFT, MSE reward shaping, a Neural Gaussian Ensemble (Fishkov et al., 2025), and DAR on the synthetic 1D Gaussian mixture task. DAR achieves the highest Spearman correlation among the LLM-based methods and better follows the target structure, particularly in the extrapolation region $(6, 10)$, while SFT and MSE show larger deviations.

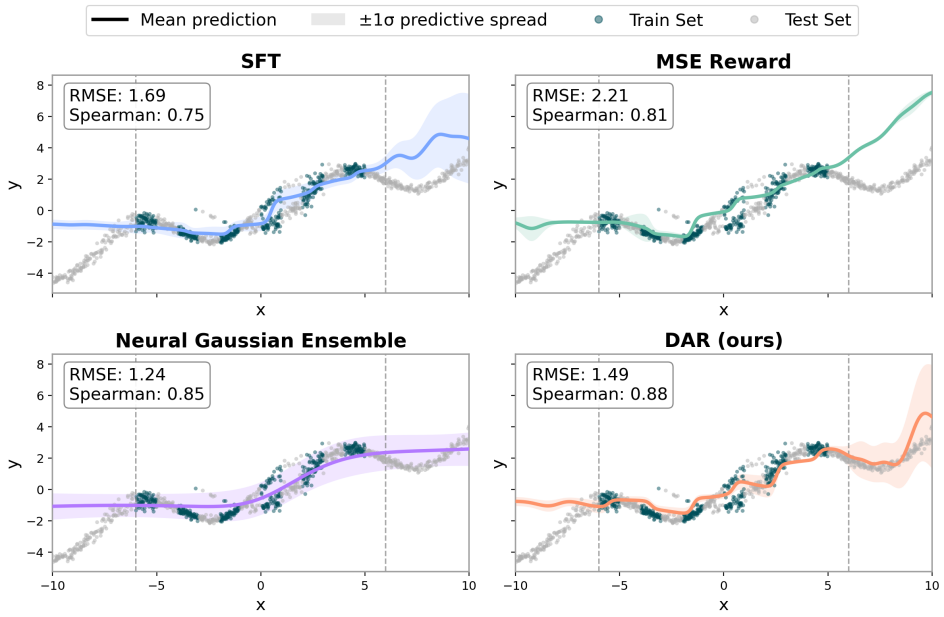


Figure 2: **Synthetic distributional regression.** Mean predictions and $\pm 1\sigma$ predictive spread on the 1D Gaussian mixture task. Dashed vertical lines at $x = \pm 6$ mark the boundary between interpolation and extrapolation regions.

Model	Method	KBSS (latency)		APPS (memory)	
		ρ (↑)	MAE (↓)	ρ (↑)	MAE (↓)
Qwen3-30B-A3B	ICL (Vacareanu et al., 2024)	0.32	0.027	0.38	1.10e4
RLM-300M	SFT (Akhauri et al., 2025)	0.59	0.013	0.89	31.0
RLM-4B	SFT (Akhauri et al., 2025)	0.60	0.013	0.91	29.5
Qwen3-4B	ICL (Vacareanu et al., 2024)	0.23	0.019	0.27	3.46e8
Qwen3-4B	Value Head (Tang et al., 2024)	0.23	0.025	0.65	306
Qwen3-4B	SFT (Lukasik et al., 2025)	0.61	0.013	0.93	29.5
Qwen3-4B	MSE (Chen et al., 2025)	0.60	0.012	0.92	29.8
Qwen3-4B	DAR (ours)	**0.66**†	**0.011**	**0.96**†	**29.1**†

Table 1: **Code performance prediction.** Results on KBSS latency prediction and APPS peak-memory prediction. MAE is reported in milliseconds for KBSS and in 10^3 bytes for APPS. † denotes statistically significant improvement over the strongest baseline under paired bootstrap testing ($p < 0.05$).

The Neural Gaussian Ensemble achieves lower RMSE, as expected from its task-matched Gaussian inductive bias. In contrast, DAR achieves strong rank correlation without task-specific inductive biases or parametric assumptions. This controlled experiment therefore illustrates how distribution-aware optimization can improve learned prediction structure rather than demonstrating specialization to Gaussian mixtures.

4.2 Performance Prediction

Table 1 reports code-to-performance prediction results, where DAR achieves the strongest overall performance. In particular, DAR obtains the best rank correlation and competitive or lower MAE, showing that distribution-aware reward optimization improves code performance prediction beyond standard SFT and pointwise RL rewards. The gains are especially clear in Spearman correlation, suggesting better sensitivity to input-dependent variation and more reliable ranking of examples.

Model	Method	ESOL	FreeSolv	Lipo
		RMSE (↓)	RMSE (↓)	RMSE (↓)
D-MPNN	GNN (Yang et al., 2019)	1.050	2.082	0.683
AttentiveFP	Graph attention (Xiong et al., 2020)	0.877	2.073	0.721
Uni-Mol	3D molecular representation (Zhou et al., 2023)	0.788	1.480	0.603
Qwen3-30B-A3B	ICL (Vacareanu et al., 2024)	1.886	4.160	1.427
Mol-Instructions	Molecular instruction (Fang et al., 2023)	2.271	6.891	1.691
ChemLLM	Chemical LLM (Zhang et al., 2024)	1.946	10.452	1.797
LlaSMol	Molecular LLM (Yu et al., 2024)	1.150	7.150	1.010
Qwen3-4B	ICL (Vacareanu et al., 2024)	2.133	4.368	1.284
Qwen3-4B	Value Head (Tang et al., 2024)	0.893	3.460	0.946
Qwen3-4B	SFT (Lukasik et al., 2025)	0.881	2.755	0.820
Qwen3-4B	MSE (Chen et al., 2025)	0.878	2.561	0.822
Qwen3-4B	DAR (ours)	**0.871**	**2.360**†	**0.776**†

Table 2: **Molecular property prediction.** RMSE on MoleculeNet regression tasks. ESOL is measured in log molar solubility, FreeSolv in kcal/mol hydration free energy, and Lipophilicity in logD. Graph and molecular ML baselines use task-specific molecular representations, while the LLM methods use only SMILES strings. † denotes statistically significant improvement over the strongest LLM baseline under paired bootstrap testing ($p < 0.05$).

Off-the-shelf ICL performs poorly, suggesting that general-purpose LLMs struggle to infer precise continuous performance metrics from source code without task-specific adaptation. The value-head baseline also underperforms, indicating that a regression head alone is insufficient for these long-context tasks. SFT substantially improves performance with task-specific supervision, while DAR further improves over SFT by directly optimizing the predictive distribution. Pointwise MSE reward shaping provides limited improvement over SFT, whereas DAR achieves stronger performance, particularly in rank correlation.

4.3 Molecule Property Prediction

Table 2 reports MoleculeNet property prediction results. Graph-based and 3D molecular models remain strong baselines because they use chemistry-aware representations, including atom-bond graph structure, curated molecular features, or geometric information (Yang et al., 2019; Xiong et al., 2020; Zhou et al., 2023). In contrast, the LLM methods operate only on SMILES strings, which provide a textual representation of molecular structure without explicit graph or 3D inductive biases.

Despite this disadvantage, regression fine-tuning substantially improves LLM performance. DAR consistently outperforms other training strategies, with statistically significant gains on FreeSolv and Lipophilicity. While MSE reward shaping sometimes improves over SFT, DAR achieves the strongest overall results, supporting the benefit of distribution-level optimization over pointwise objectives. ICL (Vacareanu et al., 2024) and value-head fine-tuning (Tang et al., 2024) remain less effective, likely due to limited exposure to training examples in ICL and the difficulty of learning randomly initialized numeric prediction modules.

5 Analysis

We analyze the effects of distribution-aware reward shaping through predictive distribution quality and calibration, rollout behavior during training, and comparisons with alternative objectives.

5.1 Predictive Distribution Quality and Calibration

Rollout Dispersion as Predictive Uncertainty. We first examine whether variation across rollouts provides meaningful example-level uncertainty. Following common practice in probabilistic regression (Rasmussen & Williams, 2006; Lakshminarayanan et al., 2017; Kendall & Gal, 2017), we use the standard deviation across sampled predictions as the uncertainty estimate and compare it with the

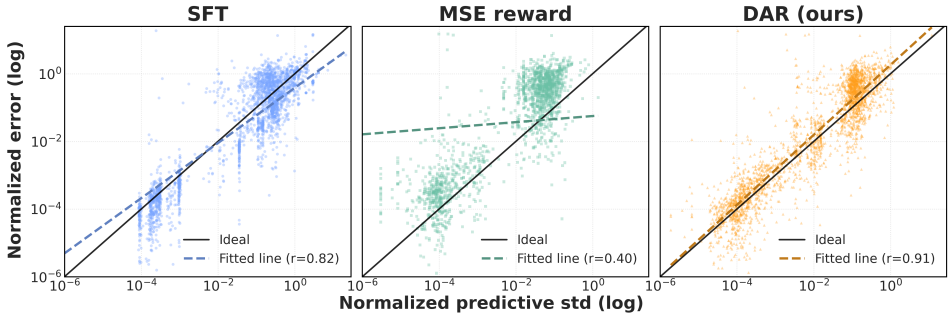


Figure 3: **Distribution-level calibration and uncertainty diagnostics.** We compare predictive distributions from SFT, MSE, and DAR. The plots show normalized predictive standard deviation versus normalized prediction error on a log-log scale, along with fitted trends and the ideal diagonal.

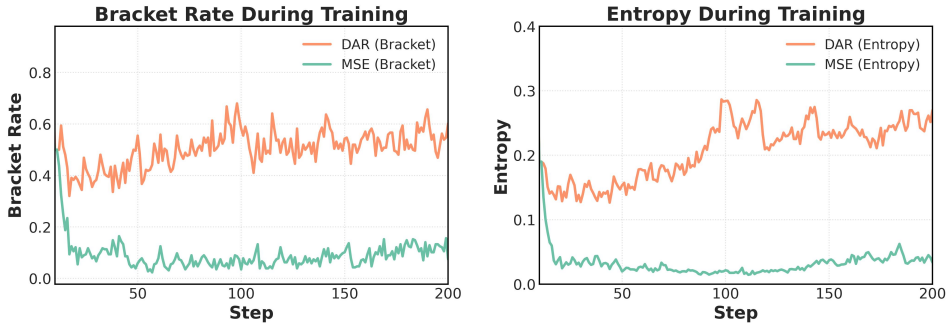


Figure 5: **Training dynamics during RL under DAR (ours) and MSE.** Bracket rate measures the fraction of examples whose rollout set contains predictions on both sides of the ground truth, while entropy measures prediction diversity, with decreasing entropy indicating potential mode collapse.

corresponding prediction error. Figure 3 shows that DAR exhibits the strongest alignment between predictive standard deviation and realized error, suggesting that its rollout dispersion better reflects prediction difficulty than SFT or MSE reward shaping.

Predictive Distribution Calibration. We next evaluate predictive distribution quality using 90% prediction-interval coverage error. Coverage error measures calibration by comparing empirical prediction-interval coverage with the nominal level (Kuleshov et al., 2018). In particular, a well-calibrated 90% prediction interval should contain the true target approximately 90% of the time, corresponding to a coverage error near zero (the lower the better). We apply the same post-hoc variance recalibration procedure (Laves et al., 2020) to all methods and report macro-averaged performance across the five datasets. DAR reduces the macro-average 90% coverage error from 0.15 for SFT and 0.42 for MSE to 0.03 (Figure 4), indicating substantially better agreement with nominal coverage. We report additional distributional metrics results in Appendix E. Together, these results show that distribution-aware training improves both the quality and calibration of the resulting predictive distributions.

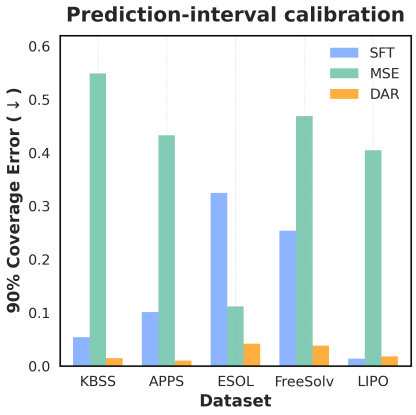


Figure 4: **Prediction-interval calibration.** 90% coverage error after post-hoc variance recalibration.

5.2 Distribution of Rollouts During Training

To examine how distribution-aware reward shaping affects GRPO (Shao et al., 2024) training, we analyze rollout diversity, motivated by prior work showing that RL fine-tuning can reduce output

diversity or policy entropy (Kirk et al., 2024; Cui et al., 2025). For each input x with rollout predictions $p_{1:K}$ and target y, we measure the bracket rate, $\mathbb{I}\left(\min_k p_k \leq y \leq \max_k p_k\right)$, indicating whether the predictions span the ground-truth value. We also track token-level policy entropy during training. Together, these metrics capture target-covering diversity and policy-level stochasticity.

Figure 5 shows the KBSS results. Compared with MSE reward shaping, DAR achieves a higher bracket rate and maintains higher policy entropy throughout training. These results show that distribution-aware reward shaping maintains useful variation across rollouts and improves target coverage, providing a complementary view of how DAR changes sampling behavior during training.

5.3 Alternative Objectives

We study whether simply increasing diversity or using distributional objectives can reproduce the gains of DAR (Table 3). Adding variance regularization slightly improves on LIPO but not KBSS, while entropy regularization provides no consistent benefit. Standard value-head-based distributional objectives, including Gaussian NLL, CRPS, and quantile loss, also underperform MSE across these tasks. In contrast, DAR achieves the best performance on both tasks, suggesting that its gains cannot be explained simply by increased diversity or conventional distributional regression. Full results and component ablations are reported in Appendix F.

Method	**KBSS** ↑	**LIPO** ↓
MSE	0.60	0.822
+ Variance	0.59	0.807
+ Entropy	0.52	0.828
Gaussian NLL	0.57	0.910
Gaussian CRPS	0.58	0.907
Quantile / Pinball	0.58	0.920
DAR	**0.66**	**0.776**

Table 3: **Alternative Objectives.**

6 Related Work

LLM Regression. Recent work has explored LLMs as regressors through direct numeric prediction, in-context learning, embedding-based approaches, and regression-specific inference or fine-tuning (Song et al., 2024; Vacareanu et al., 2024; Nguyen et al., 2024; Tang et al., 2024; Yuchi et al., 2026; Lukasik et al., 2024; 2025; Chiang et al., 2025; Chen et al., 2025). Probabilistic approaches further elicit predictive distributions through quantile or numerical distribution prediction (Vedula et al., 2025; Piskorz et al., 2026), while recent benchmarks evaluate LLM regression across code, text, and model-performance prediction (Akhauri et al., 2025; Tchuindjo & Khattab, 2025; Park et al., 2025). We extend this direction with on-policy RL using a distribution-aware reward.

RL and Distribution-Aware Reward Design. RL for LLM-based numeric prediction has primarily focused on forecasting, optimizing reasoning, accuracy, or calibration (Luo et al., 2025; Koa et al., 2025; Turtel et al., 2025; Chandak et al., 2025). Related work also considers uncertainty, calibration, and preserving output diversity during fine-tuning (Bani-Harouni et al., 2026; Damani et al., 2026; Petrenko et al., 2026; He et al., 2025; Xiao et al., 2025; Anschel et al., 2025). In contrast, we study non-temporal scalar regression and optimize the empirical predictive distribution over numeric rollouts. Our leave-one-out credit assignment differs from RLOO (Kool et al., 2019; Ahmadian et al., 2024): rather than using leave-one-out rewards as an advantage baseline, we score each rollout's contribution to distribution quality. Our approach also differs from ranking objectives (Burges et al., 2005; Joachims, 2002), which optimize relative ordering rather than numeric predictive distributions.

7 Conclusion

We introduced DAR, a distribution-aware RL objective that optimizes the predictive distribution formed by multiple LLM rollouts. Using CRPS with leave-one-out credit assignment, DAR translates distribution-level quality into rollout-level rewards for on-policy RL. Across synthetic and real-world scientific regression tasks, DAR improves point estimation and ranking while producing better-calibrated uncertainty estimates. Our analyses further show that distribution-aware training maintains meaningful variation across rollouts without explicitly maximizing dispersion. Together, these results highlight distribution-aware optimization as a promising direction for LLM regression with both accurate predictions and informative uncertainty.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. URL https://aclanthology.org/2024.acl-long.662/.

Yash Akhauri, Xingyou Song, Arissa Wongpanich, Bryan Lewandowski, and Mohamed S. Abdelfattah. Regression language models for code. *arXiv preprint arXiv:2509.26476*, 2025. URL https://arxiv.org/abs/2509.26476.

Oron Anschel, Alon Shoshan, Adam Botach, Shunit Haviv Hakimi, Asaf Gendler, Emanuel Ben Baruch, Nadav Bhonker, Igor Kviatkovsky, Manoj Aggarwal, and Gerard Medioni. Group-aware reinforcement learning for output diversity in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32394–32415, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1649.

David Bani-Harouni, Chantal Pellegrini, Paul Stangel, Ege Özsoy, Kamilia Zaripova, Nassir Navab, and Matthias Keicher. Rewarding doubt: A reinforcement learning approach to calibrated confidence expression of large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yResLmrVO1.

Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96, 2005.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Scaling open-ended reasoning to predict the future. *arXiv preprint arXiv:2512.25070*, 2025. URL https://arxiv.org/abs/2512.25070.

Ming Chen, Sheng Tang, Rong-Xi Tan, Ziniu Li, Jiacheng Chen, Ke Xue, and Chao Qian. Beyond token-level supervision: Unlocking the potential of decoding-based regression via reinforcement learning. *arXiv preprint arXiv:2512.06533*, 2025.

Cheng-Han Chiang, Hung-yi Lee, and Michal Lukasik. TRACT: Regression-aware fine-tuning meets chain-of-thought reasoning for LLM-as-a-judge. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2934–2952, Vienna, Austria, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.147/.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025. URL https://arxiv.org/abs/2505.22617.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty. In *International Conference on Learning Representations*, 2026.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. *arXiv preprint arXiv:2306.08018*, 2023.

Alexander Fishkov, Kajetan Schweighofer, Mykyta Ielanskyi, Nikita Kotelevskii, Mohsen Guizani, and Maxim Panov. Uncertainty quantification for regression using proper scoring rules. *arXiv preprint arXiv:2509.26610*, 2025.

Tilmann Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011. doi: 10.1198/jasa.2011.r10138.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2): 243–268, 2007.

Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting grpo beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25559–25571, 2025.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133–142, 2002.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PXD3FAVHJT.

Kelvin JL Koa, Jan Chen, Yunshan Ma, Huanhuan Zheng, and Tat-Seng Chua. Reasoning on time-series for financial technical analysis. *arXiv preprint arXiv:2511.08616*, 2025. URL https://arxiv.org/abs/2511.08616.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *Deep Reinforcement Learning Meets Structured Prediction Workshop at ICLR*, 2019. URL https://openreview.net/forum?id=r1lgTGL5DE.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2796–2804. PMLR, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Max-Heinrich Laves, Sontje Ihler, Jacob F. Fast, Lüder A. Kahrs, and Tobias Ortmaier. Well-calibrated regression uncertainty in medical imaging with deep learning. In *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, volume 121 of *Proceedings of Machine Learning Research*, pp. 393–412. PMLR, 2020.

Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. Evaluating and calibrating uncertainty prediction in regression tasks. *Sensors*, 22(15):5540, 2022. doi: 10.3390/s22155540.

Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Regression aware inference with llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13667–13678, 2024. doi: 10.18653/v1/2024.findings-emnlp.799. URL https://aclanthology.org/2024.findings-emnlp.799/.

Michal Lukasik, Zhao Meng, Harikrishna Narasimhan, Yin-Wen Chang, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Better autoregressive regression with LLMs via regression-aware fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xGs7Ch3Vyo.

Yucong Luo, Yitong Zhou, Mingyue Cheng, Jiahao Wang, Daoyu Wang, Tingyue Pan, and Jintao Zhang. Time series forecasting as reasoning: A slow-thinking approach with reinforced llms. *arXiv preprint arXiv:2506.10630*, 2025. URL https://arxiv.org/abs/2506.10630.

James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

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. *arXiv preprint arXiv:2410.10190*, 2024. URL https://arxiv.org/abs/2410.10190.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL https://arxiv.org/abs/2203.02155.

Jungsoo Park, Ethan Mendes, Gabriel Stanovsky, and Alan Ritter. Anticipatory evaluation of language models. *arXiv preprint arXiv:2509.20645*, 2025. doi: 10.48550/arXiv.2509.20645. URL https://arxiv.org/abs/2509.20645.

Aleksei Petrenko, Ben Lipkin, Kevin Chen, Erik Wijmans, Marco Cusumano-Towner, Raja Giryes, and Philipp Krähenbühl. Entropy-preserving reinforcement learning. *arXiv preprint arXiv:2603.11682*, 2026.

Julianna Piskorz, Katarzyna Kobalczyk, and Mihaela van der Schaar. Eliciting numerical predictive distributions of LLMs without auto-regression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SsuBd46twl.

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006. ISBN 026218253X.

James Requeima, John Bronskill, Dami Choi, Richard Turner, and David K. Duvenaud. Llm processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 109609–109671, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024. doi: 10.48550/arXiv.2409.19256. URL https://arxiv.org/abs/2409.19256.

Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors. *arXiv preprint arXiv:2402.14547*, 2024. URL https://arxiv.org/abs/2402.14547.

Eric Tang, Bangding Yang, and Xingyou Song. Understanding llm embeddings for regression. *arXiv preprint arXiv:2411.14708*, 2024. URL https://arxiv.org/abs/2411.14708.

Diane Tchuindjo and Omar Khattab. Reasoning-intensive regression. *arXiv preprint arXiv:2508.21762*, 2025. URL https://arxiv.org/abs/2508.21762.

Benjamin Turtel, Danny Franklin, Kris Skotheim, Luke Hewitt, and Philipp Schoenegger. Outcome-based reinforcement learning to predict the future. *arXiv preprint arXiv:2505.17989*, 2025. URL https://arxiv.org/abs/2505.17989.

Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. *CoRR*, 2024.

Nikhita Vedula, Dushyanta Dhyani, Laleh Jalali, Boris N. Oreshkin, Mohsen Bayati, and Shervin Malmasi. Quantile regression with large language models for price prediction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12396–12415, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.641.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988. doi: 10.1021/ci00057a005.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. doi: 10.1039/C7SC02664A.

Jiancong Xiao, Bojian Hou, Zhanliang Wang, Ruochen Jin, Qi Long, Weijie J. Su, and Li Shen. Restoring calibration for aligned large language models: A calibration-aware fine-tuning approach. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68364–68390. PMLR, 2025.

Zhaoping Xiong, Dingyan Wang, Xin Liu, Feisheng Zhong, Xiaohong Wan, Xutong Li, Zhaojun Li, Xiaoqin Luo, Kun Chen, Hualiang Jiang, and Mingyue Zheng. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, 63(16):8749–8760, 2020. doi: 10.1021/acs.jmedchem.9b00959.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philip Eiden, Hua Gao, Angelica Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Alex Palmer, Volker Settels, Tommi Jaakkola, Klavs F. Jensen, and Regina Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019. doi: 10.1021/acs.jcim.9b00237.

Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*, 2024.

Fengting Yuchi, Li Du, and Jason Eisner. Llms know more about numbers than they can say. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 659–673, 2026.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Dongzhan Zhou, Shufei Zhang, Mao Su, Hansen Zhong, Yuqiang Li, and Wanli Ouyang. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024. URL https://arxiv.org/abs/2402.06852.

Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=6K2RM6wVqKu.

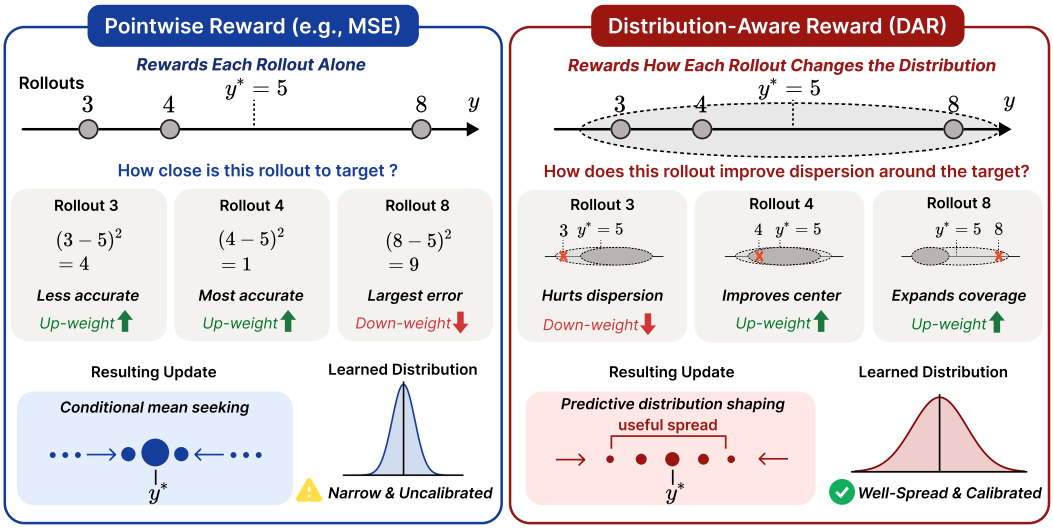


Figure 6: **Pointwise versus distribution-aware reward shaping.** Pointwise MSE reward scores each rollout independently, encouraging predictions to collapse toward the mean. DAR instead scores each rollout by its leave-one-out contribution to the full predictive distribution, rewarding both accuracy and useful spread around the ground truth.

A Proofs for Pointwise Squared Loss and CRPS

In this section, we formalize the difference between pointwise squared loss and CRPS at the population level. The key distinction is that pointwise squared loss evaluates a single prediction in isolation, whereas CRPS evaluates an entire predictive distribution.

A.1 Theoretical Comparison of Pointwise Squared Loss and CRPS

The two objectives induce different optimal solutions: squared loss is mean-seeking, while CRPS is distribution-seeking (Gneiting, 2011; Gneiting & Raftery, 2007; Matheson & Winkler, 1976). These results are standard consequences of the Bayes optimality of squared loss for the conditional mean and the strict propriety of CRPS, but we restate them here to clarify the optimization bias induced by each objective in our multi-rollout regression setting.

Standard Result 1: Pointwise squared loss is mean-seeking. Fix an input x, and let $Y \mid X = x \sim P_x$ with finite second moment. Define the population pointwise squared-loss objective

$$\mathcal{L}_{\mathrm{MSE}}(a; x) = \mathbb{E}\big[(a - Y)^2 \mid X = x\big], \qquad a \in \mathbb{R}. \tag{7}$$

Then $\mathcal{L}_{\mathrm{MSE}}(a; x)$ is uniquely minimized at

$$a^\star(x) = \mu_x := \mathbb{E}[Y \mid X = x]. \tag{8}$$

Equivalently, among predictive distributions over scalar predictions, the minimizer of expected pointwise squared loss is the degenerate predictor δ_{μ_x} concentrated at the conditional mean (Gneiting, 2011).

Proof. Let $\mu_x = \mathbb{E}[Y \mid X = x]$. Expanding the square around μ_x gives

$$\mathcal{L}_{\mathrm{MSE}}(a;x) = \mathbb{E}\big[(a-Y)^2 \mid X=x\big] \tag{9}$$
$$= \mathbb{E}\big[(a-\mu_x+\mu_x-Y)^2 \mid X=x\big] \tag{10}$$
$$= (a-\mu_x)^2 + 2(a-\mu_x)\mathbb{E}[\mu_x - Y \mid X=x] + \mathbb{E}\big[(\mu_x-Y)^2 \mid X=x\big]. \tag{11}$$

Since $\mathbb{E}[\mu_x - Y \mid X=x] = 0$, this simplifies to

$$\mathcal{L}_{\mathrm{MSE}}(a;x) = (a-\mu_x)^2 + \mathrm{Var}(Y \mid X=x). \tag{12}$$

The variance term does not depend on a, so the unique minimizer is $a = \mu_x$.

Now consider a predictive distribution F over scalar predictions $\hat{Y}$. If the model is evaluated by pointwise squared loss on a single sampled prediction $\hat{Y} \sim F$, then the population objective becomes

$$\mathbb{E}_{\hat{Y}\sim F}\,\mathbb{E}\big[(\hat{Y}-Y)^2 \mid X=x\big] = \mathbb{E}_{\hat{Y}\sim F}\left[(\hat{Y}-\mu_x)^2 + \mathrm{Var}(Y \mid X=x)\right] \tag{13}$$
$$= \mathbb{E}_{\hat{Y}\sim F}[(\hat{Y}-\mu_x)^2] + \mathrm{Var}(Y \mid X=x). \tag{14}$$

This is minimized by placing all mass at μ_x, namely $F = \delta_{\mu_x}$. Therefore, under pointwise squared loss, any predictive spread away from the conditional mean strictly increases the objective. This establishes that pointwise squared loss is inherently mean-seeking and favors collapse to a point mass, which is empirically verified in our analysis.

Standard Result 2: CRPS is distribution-seeking. Fix an input x, and let $Y \mid X = x \sim P_x$. For any predictive distribution F on $\mathbb{R}$ with finite first moment, define

$$\mathcal{L}_{\mathrm{CRPS}}(F;x) = \mathbb{E}_{Y\sim P_x}\big[\mathrm{CRPS}(F,Y)\big]. \tag{15}$$

Then $\mathcal{L}_{\mathrm{CRPS}}(F;x)$ is minimized at $F = P_x$. In particular, if P_x is non-degenerate, then for any point mass δ_a,

$$\mathcal{L}_{\mathrm{CRPS}}(P_x;x) < \mathcal{L}_{\mathrm{CRPS}}(\delta_a;x). \tag{16}$$

Hence, when $\mathrm{Var}(Y \mid X=x) > 0$, collapse to any single point, including the conditional mean, is suboptimal under CRPS (Matheson & Winkler, 1976; Gneiting & Raftery, 2007).

Proof. CRPS is a strictly proper scoring rule for univariate predictive distributions (Matheson & Winkler, 1976; Gneiting & Raftery, 2007). Therefore, for any true conditional distribution P_x and any predictive distribution F with finite first moment,

$$\mathbb{E}_{Y\sim P_x}\big[\mathrm{CRPS}(P_x,Y)\big] \le \mathbb{E}_{Y\sim P_x}\big[\mathrm{CRPS}(F,Y)\big], \tag{17}$$

with equality if and only if $F = P_x$.

This immediately proves the first claim: the population CRPS objective is minimized by the true conditional predictive distribution.

For the second claim, suppose P_x is non-degenerate. Then $P_x \neq \delta_a$ for every $a \in \mathbb{R}$. By strict propriety of CRPS, any predictive distribution different from P_x must incur strictly larger expected CRPS. In particular,

$$\mathcal{L}_{\mathrm{CRPS}}(P_x;x) < \mathcal{L}_{\mathrm{CRPS}}(\delta_a;x) \qquad \text{for all } a \in \mathbb{R}. \tag{18}$$

Applying this to $a = \mu_x = \mathbb{E}[Y \mid X = x]$ shows that the mean-collapsed predictor δ_{μ_x} is strictly suboptimal whenever the conditional target distribution has nonzero spread.

Thus, unlike pointwise squared loss, CRPS does not generally favor collapse to the conditional mean. Rather, its population optimum matches the true conditional distribution: it remains non-degenerate when the conditional target distribution is non-degenerate, and reduces to a point mass at the ground-truth target when the conditional target is deterministic. As discussed in Section B, the benefit of DAR in the deterministic setting therefore arises not from changing the population optimum, but from providing a distribution-aware learning signal over finite stochastic rollouts.

A.2 Implication

The two standard results above formalize the contrast between pointwise and distributional regression objectives at the population level. Pointwise squared loss treats each prediction independently and is optimized by the conditional mean, so it provides no incentive to represent predictive uncertainty. In contrast, CRPS is a strictly proper scoring rule for predictive distributions and is optimized by the true conditional predictive law when that law is observed through repeated samples.

This population-level result motivates CRPS as a distribution-aware scoring rule. In our finite-sample LLM setting, however, DAR does not directly observe the full conditional distribution for each input. Instead, it uses the empirical negated CRPS against the available scalar target as a reward over multiple sampled rollouts. Thus, DAR should be viewed as a practical distribution-aware credit-assignment objective that encourages well-centered and appropriately dispersed rollout sets, rather than as a guarantee of recovering the true conditional distribution.

B Distribution-Aware Rewards for Deterministic Regression

Even when each input has a single deterministic target, an LLM naturally induces a predictive distribution through multiple numerical rollouts due to finite data, model approximation, and stochastic generation. This raises a natural question: why should optimizing this distribution help when the underlying regression target itself is deterministic?

A conventional pointwise reward such as MSE evaluates each rollout independently, encouraging sampled predictions to concentrate around a single point. In contrast, DAR evaluates each rollout by how it changes the quality of the empirical predictive distribution formed by all sampled rollouts. As illustrated in Figure 1, a rollout can therefore receive useful credit not only for being individually close to the target, but also for improving the distribution as a whole—for example, by shifting its center toward the target or placing predictions on both sides of the target. Conversely, rollouts that introduce unhelpful dispersion are penalized. Thus, DAR encourages a predictive distribution that is accurate and appropriately dispersed, rather than diversity for its own sake.

This distributional supervision can also improve the resulting point estimate. Our final prediction is obtained by averaging multiple sampled outputs. When the sampled predictions fall on both sides of the target, positive and negative errors can partially cancel under averaging, reducing the error of the aggregate prediction. Consistent with this mechanism, DAR increases the bracketing rate—the fraction of examples for which the rollout set contains predictions on both sides of the ground-truth target—from 37.7% to 53.6% (Figure 5). This provides empirical evidence for how distribution-level supervision can improve point-prediction accuracy even when the target is deterministic.

Importantly, DAR does not encourage predictive spread for its own sake. CRPS jointly accounts for accuracy and dispersion, so additional spread is beneficial only when it improves the predictive distribution relative to the observed target. As predictions approach the ground truth, unnecessary dispersion is increasingly penalized. The role of DAR is therefore to provide a richer optimization signal during learning, rather than to maintain persistent uncertainty at convergence.

This intuition does not require stochastic or repeated labels. CRPS is a strictly proper scoring rule that can be evaluated against a single observed target (Gneiting & Raftery, 2007). For a predictive distribution F and target y,

$$\mathrm{CRPS}(F, y) = \mathbb{E}|X - y| - \frac{1}{2}\mathbb{E}|X - X'|, \qquad X, X' \sim F. \tag{19}$$

By the triangle inequality,

$$\frac{1}{2}\mathbb{E}|X - X'| \leq \mathbb{E}|X - y|, \tag{20}$$

and therefore

$$\mathrm{CRPS}(F, y) \geq 0. \tag{21}$$

Equality is attained when $F = \delta_y$, such that all predictive mass is concentrated at the correct target. Hence, for deterministic regression, the global optimum under CRPS remains a point mass at the ground-truth target. DAR therefore does not alter the desired solution; rather, it changes the optimization signal and trajectory used to reach it.

C Estimator Properties of Leave-One-Out-CRPS

Our method uses LOO-CRPS to assign a distribution-aware reward to each rollout based on its contribution to the empirical predictive distribution. Because this reward is computed from a finite set of sampled rollouts, two theoretical questions naturally arise: how the leave-one-out (LOO) credit assignment affects the resulting policy-gradient estimator, and whether empirical CRPS introduces finite-sample bias. We analyze both properties below.

Leave-One-Out credit assignment. LOO credit assignment does not introduce additional bias into the policy-gradient estimator. For a group of K rollouts $\{x_1, \ldots, x_K\}$, the LOO baseline for rollout x_k is computed exclusively from the remaining $K - 1$ rollouts. Conditional on the input, this baseline is therefore independent of the stochastic action corresponding to rollout x_k. As in standard multi-sample LOO estimators (Kool et al., 2019), subtracting such a baseline does not change the expected REINFORCE gradient:

$$\mathbb{E}_{x_k \sim \pi_\theta}\left[b_{-k} \nabla_\theta \log \pi_\theta(x_k)\right] = b_{-k}\mathbb{E}_{x_k \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(x_k)\right] = 0, \tag{22}$$

where b_{-k} denotes a quantity computed only from the other $K - 1$ rollouts. LOO credit assignment can therefore provide rollout-specific credit while preserving the expected policy-gradient estimator.

Finite-sample properties of empirical CRPS. The empirical CRPS computed from a finite rollout set is a biased but consistent estimator of the population CRPS. For K i.i.d. samples $X_1, \ldots, X_K \sim F$, the empirical CRPS can be written as

$$\widehat{\mathrm{CRPS}}_K = \frac{1}{K}\sum_{k=1}^{K}|X_k - y| - \frac{1}{2K^2}\sum_{k=1}^{K}\sum_{j=1}^{K}|X_k - X_j|. \tag{23}$$

The first term is an unbiased estimator of $\mathbb{E}|X - y|$. For the pairwise dispersion term, the K diagonal terms are zero, leaving $K(K-1)$ nonzero pairs in expectation. Consequently,

$$\mathbb{E}\left[\frac{1}{2K^2}\sum_{k=1}^{K}\sum_{j=1}^{K}|X_k - X_j|\right] = \frac{K-1}{K}\frac{1}{2}\mathbb{E}|X - X'|, \tag{24}$$

where $X, X' \sim F$ are independent. The empirical estimator therefore underestimates the dispersion term by a factor of $(K-1)/K$, resulting in an $O(1/K)$ finite-sample bias.

As $K \to \infty$, this factor converges to one, and the empirical CRPS converges to the population CRPS. Correspondingly, the LOO contribution increasingly approximates the marginal effect of each rollout on the population objective. Importantly, strict propriety is a property of the population CRPS and does not imply that each finite-K LOO reward is itself a proper scoring rule. At finite K, the

empirical objective is slightly conservative with respect to predictive dispersion; this bias vanishes as the number of rollouts increases.

Together, these properties distinguish two aspects of the estimator: LOO credit assignment does not introduce additional policy-gradient bias, while empirical CRPS has a finite-sample $O(1/K)$ bias that vanishes asymptotically.

D Experimental Setup

In this section, we provide additional details on the experimental setup for the synthetic benchmark and the two real-world regression datasets, as well as the implementation details and baseline methods.

D.1 Tasks

Synthetic Distributional Regression. We construct a synthetic 1D distributional regression benchmark following Appendix E.2 of Fishkov et al. (2025). For each input $x \in \mathbb{R}$, the ground-truth conditional distribution is a heteroscedastic two-component mixture:

$$p^*(y \mid x) = \pi(x)P_1(y \mid x) + \big(1 - \pi(x)\big)P_2(y \mid x). \tag{25}$$

The input-dependent mixing weight is defined as

$$\pi(x) = \frac{1}{1 + \exp(1.2x)}, \tag{26}$$

and the two component means are given by

$$\mu_1(x) = \frac{x}{3} + 1.2\sin(0.8x), \tag{27}$$

$$\mu_2(x) = \frac{x}{3} - 1.2\cos(0.8x). \tag{28}$$

The input-dependent noise scale is

$$\sigma(x) = 0.12 + 0.28\left(0.5 + 0.5\sin(0.7x)\right)^2. \tag{29}$$

Each component is Gaussian with shared heteroscedastic variance:

$$P_1(y \mid x) = \mathcal{N}\big(y; \mu_1(x), \sigma^2(x)\big), \tag{30}$$

$$P_2(y \mid x) = \mathcal{N}\big(y; \mu_2(x), \sigma^2(x)\big). \tag{31}$$

Equivalently, one may sample from the distribution via

$$k \sim \text{Bernoulli}(\pi(x)), \tag{32}$$

$$\epsilon \sim \mathcal{N}(0, 1), \tag{33}$$

$$y = \mu_k(x) + \epsilon\,\sigma(x). \tag{34}$$

Following the same setup, we draw $n = 1200$ training pairs (x, y) from $p^*(y \mid x)$. In our implementation, training inputs are sampled from the in-domain interval $x \in [-6, 6]$, while the test set covers both the training region and additional disjoint regions outside $[-6, 6]$ in order to evaluate both interpolation and extrapolation behavior. Figure 7 shows the one-dimensional distribution of the data samples.



Figure 7: Illustration of data samples

Coding Performance Prediction. For code performance prediction, we use the `akhauriyash/Code-Regression` benchmark and restrict attention to the APPS and KBSS portions.[1] Following Akhauri et al. (2025), we construct random train/validation/test splits over these subsets. We adopt this protocol because the original authors do not provide an official fixed split for these data, and thus the random-split setup is consistent with their experimental practice. The task is to predict a continuous execution-related target from code alone, and our evaluation follows the standard regression setting used in prior work. The released dataset card specifies an MIT license for the packaged benchmark.

MoleculeNet Property Prediction. For molecular property prediction, we use the MoleculeNet benchmark (Wu et al., 2018) and focus on the three commonly used physicochemical regression subsets: ESOL, FreeSolv, and Lipophilicity (Lipo). MoleculeNet explicitly includes these regression tasks and recommends scaffold-based evaluation for such chemistry benchmarks, which is also the protocol commonly followed in recent work (Zhou et al., 2023; Yu et al., 2024). In line with prior molecular representation work, including Uni-Mol and subsequent studies, we use the official scaffold split for ESOL, FreeSolv, and Lipo.

Regarding licensing, MoleculeNet is a benchmark that aggregates public molecular datasets, while the DeepChem software commonly used to access these benchmark tasks is released under the MIT License. Since ESOL, FreeSolv, and Lipophilicity originate from public source datasets collected into the MoleculeNet benchmark, usage should also respect the terms of the original underlying sources in addition to the benchmark software license. This benchmark structure is documented in MoleculeNet and related dataset documentation.

D.2 Baselines

For zero-shot evaluation of domain-specific language models, we follow the default generation protocol described in each original paper (Fang et al., 2023; Zhang et al., 2024; Yu et al., 2024). For ICL (Vacareanu et al., 2024), we use 50 in-context examples for MoleculeNet property prediction and 10 for coding regression due to context length constraints. Following the original setup, these examples are randomly sampled from the training set.

For the value-head baseline, we follow the embedding-based regression setup of Tang et al. (2024). Specifically, we keep the language model frozen and train only a lightweight prediction head on top of fixed input representations. For each example, we extract a fixed-dimensional embedding from the `prompt` field using a pretrained Qwen instruction model, and use this embedding as input to a two-hidden-layer MLP regressor with ReLU activations. The regression head is trained with mean squared error (MSE) on normalized target values, where the normalization statistics are computed only from the training split. At evaluation time, predictions are transformed back to the original target scale before computing all regression metrics. We optimize the regressor with AdamW, select the

[1]The Hugging Face dataset card lists the dataset license as MIT.

Category	Value
Framework	`verl`
Training method	GRPO
Train batch size	256
Learning rate	1×10^{-6}
PPO mini-batch size	16
PPO micro-batch size / GPU	4
Entropy coefficient	0
Use KL loss	True
KL loss coefficient	0.001
KL loss type	`low_var_kl`
Use KL in reward	False
Rollout engine	vLLM
Rollouts per prompt (n)	12
Sampling temperature	1.0
Top-p	1.0
Rollout GPU memory utilization	0.8
Rollout log-prob micro-batch size / GPU	8
Reference log-prob micro-batch size / GPU	8
Gradient checkpointing	True
Parameter offload	False
Optimizer offload	False
Critic warmup	0
GPUs	4
Nodes	1

Table 4: **Main RL hyperparameters for DAR training in `verl`.** Our method follows the standard GRPO pipeline and differs mainly in the reward computation.

best checkpoint based on validation performance, and use early stopping with a maximum of 300 training steps. Overall, this setup closely follows the protocol of Tang et al. (2024).

For both SFT and RL training, we use the default training configuration provided in the `verl` repository. We keep the batch size and the number of training steps the same across methods for fair comparison.

D.3 Computing Resources

We use a single node with 4 H100 GPUs for all SFT and RL training runs. Depending on the model scale and the maximum number of training steps, each run takes approximately 1–2 days. Our method does not introduce additional computational overhead relative to the pointwise GRPO baseline.

D.4 Implementation Details

Table 4 shows the detailed setup of the training. We implement both supervised fine-tuning (SFT) and reinforcement learning (RL) using the `verl`[2] framework. For RL, we build on the standard GRPO training pipeline and replace only the reward computation with our distribution-aware reward. Concretely, for each prompt, the policy generates multiple rollouts, and our reward manager computes rollout-level rewards based on each sample's contribution to the predictive distribution induced by the full rollout set. This design requires no additional value head, no auxiliary uncertainty head, and no extra learned scorer beyond the standard policy/reference setup.

Our method introduces only minimal computational overhead relative to standard GRPO. The dominant training cost remains rollout generation and policy optimization, which are unchanged. The additional cost comes only from reward-side aggregation over the sampled rollouts for each prompt. In practice, this is lightweight compared to model forward passes, since it operates on scalar predictions after decoding rather than on token-level hidden states. Thus, the training pipeline, hardware footprint, and optimization procedure remain nearly identical to standard GRPO. We trained

[2]https://github.com/verl-project/verl

each model on each dataset for sufficiently long training, up to a maximum of 300 steps, selected the best checkpoint based on validation performance, and then reported the final results on the test set.

D.5 Computational Overhead

DAR introduces only negligible computational overhead relative to standard negative-MSE RL under matched GRPO-style training settings. Importantly, multiple rollouts are not an additional cost introduced by DAR: both the negative-MSE baseline and DAR operate on the same $K = 12$ rollouts sampled per prompt. The two methods therefore share the same rollout generation, log-probability evaluation, and policy optimization procedures, differing only in how rewards are computed from the sampled predictions.

Metric	Negative-MSE	DAR	Overhead
Reward computation (ms/step)	$\sim$15	83	+56–73 ms
Share of a 207.9 s training step	$<$0.01%	0.04%	+0.03–0.04%
Reward computation at $K = 48$ (ms)	–	302	$<$0.2% of step

Table 5: **Computational overhead analysis.** Computational overhead of DAR relative to negative-MSE RL using Qwen3-4B with $K = 12$ rollouts under matched settings.

As shown in Table 5, the entire reward-processing stage for DAR accounts for only 0.04% of a 207.9 s training step. Relative to negative-MSE RL, DAR adds only 56–73 ms per step, corresponding to approximately 0.03–0.04% additional wall-clock time. We also observe no measurable increase in peak GPU memory.

The additional computation arises from the pairwise CRPS evaluation over the K predictions generated for each prompt. This computation has theoretical complexity $O(K^2)$, but it is restricted to reward evaluation and is negligible compared with rollout generation and policy optimization. Increasing the rollout count from $K = 12$ to $K = 48$ increases the measured reward-computation time from 83 ms to 302 ms, which still accounts for less than 0.2% of a typical training step. Because the measured runtime also includes fixed reward-processing overhead, the observed scaling need not follow the asymptotic pairwise complexity exactly.

Overall, DAR changes the reward computation applied to rollouts that are already required by GRPO-style training, while leaving the dominant computational components unchanged. Its distribution-aware objective therefore incurs negligible practical runtime and memory overhead.

E Distribution Characteristics

Table 6 reports CRPS together with complementary statistics describing the predictive distributions. DAR achieves the best CRPS on four of the five datasets and consistently outperforms the MSE baseline. The only exception is KBSS, where SFT obtains a lower CRPS. Moreover, DAR generally produces less asymmetric predictive distributions than SFT, with skewness values closer to zero across datasets.

On KBSS, DAR achieves better point prediction accuracy than SFT but produces a substantially narrower predictive distribution. This concentration makes DAR more sensitive to a small subset of difficult tail examples, which increases its average CRPS despite better performance on most examples. When the worst tail examples are excluded (2%), the ranking reverses and DAR slightly outperforms SFT. Thus, the KBSS result is best understood as a dataset-specific effect related to the concentration of the predictive distribution, rather than a systematic disadvantage of DAR.

F Reward and Credit Assignment Ablations

We compare DAR against several simpler alternatives, including negative-MSE objectives with variance or entropy bonuses, standard distributional regression objectives, and component ablations that replace either the CRPS objective or LOO credit assignment. As shown in Table 7, DAR achieves the best performance on all five datasets, outperforming both diversity-regularized and conventional

Dataset	Method	CRPS↓	Predictive SD	IQR	Skewness
KBSS	SFT	0.307	0.014	0.016	1.164
	MSE	0.411	0.001	0.001	-0.447
	DAR	0.339	0.005	0.007	0.279
APPS	SFT	0.071	2070.000	1067.250	2.082
	MSE	0.072	0.002	0.002	0.004
	DAR	0.070	303.279	296.402	0.399
ESOL	SFT	0.307	0.171	0.171	0.171
	MSE	0.296	0.157	0.171	-0.300
	DAR	0.279	0.255	0.305	-0.121
FreeSolv	SFT	0.460	0.317	0.371	-0.205
	MSE	0.478	0.179	0.166	0.878
	DAR	0.420	0.311	0.350	0.051
LIPO	SFT	0.505	0.219	0.249	-0.246
	MSE	0.561	0.056	0.049	-0.160
	DAR	0.498	0.128	0.162	-0.045

Table 6: **Predictive distribution quality across datasets.** CRPS is the primary metric, where lower values are better. Predictive SD, IQR, and skewness characterize the spread and shape of the raw predictive samples generated by each checkpoint. CRPS values are normalized within each dataset, while SD and IQR are reported in the original target units. All values are rounded to three decimal places.

Method	KBSS ↑	APPS ↑	FreeSolv ↓	LIPO ↓	ESOL ↓
MSE	0.60	0.92	2.561	0.822	0.878
+ Variance	0.59	0.93	2.388	0.807	0.881
+ Entropy	0.52	0.93	2.424	0.828	0.885
Gaussian NLL	0.57	0.78	3.389	0.910	0.877
Gaussian CRPS	0.58	0.67	2.866	0.907	0.883
Quantile / Pinball	0.58	0.62	2.972	0.920	0.890
CRPS + Group-Mean	0.62	0.94	2.374	0.781	0.875
NLL + LOO	0.63	0.95	2.368	0.797	0.880
DAR	**0.66**	**0.96**	**2.360**	**0.776**	**0.871**

Table 7: **Alternative objectives and component ablations.** We compare DAR with diversity-regularized MSE objectives, standard distributional regression objectives, and ablations of its CRPS objective and LOO credit assignment. Spearman correlation is reported for KBSS and APPS (↑), and RMSE for molecular property prediction tasks (↓). Best results are shown in bold.

distributional baselines as well as either component ablation. These results indicate that the gains do not arise from predictive diversity, distributional training, or either component alone, but from combining empirical distribution-level evaluation with rollout-specific LOO credit assignment.

G Sensitivity to Hyperparameters

We examine the sensitivity of DAR to two sampling hyperparameters: the number of rollouts K and the sampling temperature T. We evaluate rollout counts $K \in \{4, 8, 12, 24\}$ and sampling temperatures $T \in \{0.7, 1.0, 1.3\}$ on KBSS and the synthetic regression task.

Tables 8 and 9 show that DAR is relatively robust to both hyperparameters. Performance generally improves as the number of rollouts increases and stabilizes beyond $K = 8$, with little additional benefit from increasing K to 24. Similarly, varying the sampling temperature from 0.7 to 1.3 results

Dataset	$\mathbf{K = 4}$	$\mathbf{K = 8}$	$\mathbf{K = 12}$	$\mathbf{K = 24}$
KBSS	0.59	0.65	0.66	0.66
Synthetic	1.78	1.40	1.49	1.42

Table 8: **Sensitivity to the number of rollouts K.** We report Spearman correlation (↑) on KBSS and RMSE (↓) on the synthetic task.

Dataset	$\mathbf{T = 0.7}$	$\mathbf{T = 1.0}$	$\mathbf{T = 1.3}$
KBSS	0.62	0.66	0.65
Synthetic	1.55	1.49	1.48

Table 9: **Sensitivity to sampling temperature T.** We report Spearman correlation (↑) on KBSS and RMSE (↓) on the synthetic task.

in only modest performance differences across both tasks. These results indicate that the performance of DAR does not depend strongly on a particular choice of rollout count or sampling temperature.

H Reasoning Mode vs. Non-Reasoning Mode

On-policy RL is naturally compatible with reasoning-oriented models because it can reinforce longer intermediate derivations. We nevertheless disable explicit reasoning by default in our main experiments for efficiency, and instead study its effect through separate ablations in both the zero-shot and RL fine-tuning settings. In these ablations, we explicitly instruct the model in the system prompt to reason before producing a prediction, and set the maximum output length to 8192 during training.

Table 10 presents a controlled comparison on ESOL using Qwen3-4B-Instruct, evaluating both `Reasoning=Yes` and `Reasoning=No` under two settings: (i) zero-shot inference and (ii) DAR fine-tuning on ESOL. The results show that explicitly prompting the model to generate intermediate reasoning (Wei et al., 2022) substantially increases the number of output tokens, but does not improve regression performance in either setting. In the fine-tuning case, reasoning is in fact associated with worse performance. This trend is consistent with recent findings in GRPO training for verifiable tasks, which suggest that RL-based reasoning training can reduce output diversity (He et al., 2025; Petrenko et al., 2026). A similar effect may be present here, as the reasoning-enabled model exhibits lower variance than the non-reasoning counterpart (total variance 3.5 vs. 4.2). Note that this finding is specific to our single-task fine-tuning setup on ESOL with small scale model, where training and evaluation are performed on the same task. It does not rule out the possibility that explicit reasoning could help in broader settings such as multi-task post-training or transfer.

Reward scale. DAR also does not require task-specific reward scaling despite differences in target ranges across datasets. Under GRPO, advantages are normalized by the within-group standard deviation, making the policy-gradient signal invariant to affine rescaling of the reward. Consequently, differences in the absolute magnitude of CRPS across target ranges do not alter the effective reward scale used for policy optimization. Consistent with this property, we use the same reward formulation and KL coefficient across all five datasets without per-task tuning.

I Limitations

Comparison to traditional regression methods. While our results suggest that LLM-based regression can be useful for scientific prediction tasks, this approach also has important limitations relative to traditional nonlinear regression methods. Classical models such as Gaussian processes, kernel methods, gradient-boosted trees, and neural regression models are often more computationally efficient, easier to train, and better suited when the input is already available as structured numerical or tabular features. They also provide clearer inductive biases and, in some cases, more established uncertainty estimates. In contrast, LLM-based regression requires substantially more compute, depends on careful prompting and decoding choices, and may be less sample-efficient when the task does not benefit from language-based representations.

Model	Reasoning	Zero-shot			DAR (ESOL)		
		# Tokens	ρ (↑)	RMSE (↓)	# Tokens	ρ (↑)	RMSE (↓)
Qwen3-4B	No	1072	0.67	1.62	7	0.90	0.73
Qwen3-4B	Yes	2737	0.66	1.68	2584	0.87	1.01

Table 10: **ESOL regression results with and without explicit reasoning.** We report the average number of output tokens, Spearman correlation (ρ), and RMSE for zero-shot and DAR.

Flexibility and sensitivity. The main advantage of LLM-based regression is its flexibility: the same model can consume heterogeneous inputs such as natural-language descriptions, code, SMILES strings, metadata, or task specifications without task-specific feature engineering. This makes it attractive for settings where the input is naturally textual or where constructing specialized features is costly. However, this flexibility also introduces additional sources of variability. Predictions can be sensitive to prompt format, sampling temperature, output parsing, and the choice of backbone model. Moreover, although our distribution-aware objective improves rollout diversity and calibration, it does not eliminate all failure modes of RL training, such as unstable optimization, reward misspecification, or distribution shift between training and evaluation.

Scope of evaluation. Finally, our evaluation focuses on scalar regression tasks and predictive distributions formed from multiple sampled rollouts. Extending the method to higher-dimensional outputs, structured scientific predictions, or settings requiring strict physical constraints remains future work. Our results should therefore be interpreted as evidence that distribution-aware RL can improve LLM-based regression in the studied settings, rather than as a replacement for well-established regression methods in all domains.

J Broader Impact

LLM-based regression has the potential to support scientific domains where prediction targets are continuous quantities and inputs are heterogeneous, such as natural-language descriptions, code, molecular strings, metadata, or experimental conditions. By improving the quality of predictive distributions, our work aims to provide a foundation for using LLMs not only as text generators, but also as flexible regression models for scientific prediction. This could help accelerate early-stage screening, experiment prioritization, and inverse-design workflows, where researchers need to reason about both expected outcomes and uncertainty before committing costly experimental or computational resources.

At the same time, these models should be used as decision-support tools rather than replacements for domain expertise or empirical validation. In scientific settings, inaccurate predictions or poorly calibrated uncertainty estimates could lead to wasted resources or misleading conclusions, especially in high-stakes areas such as materials discovery, drug development, or engineering design. Therefore, outputs from LLM-based regression systems should be interpreted cautiously, validated with domain-specific methods, and accompanied by uncertainty estimates when used for downstream decision making.

More broadly, we view this work as an initial step toward general-purpose scientific regression with language models. While the current study focuses on scalar prediction tasks, future extensions could support more complex inverse-engineering and design problems, where models propose or evaluate candidate systems under uncertainty. Responsible use will require careful benchmarking, transparent reporting of failure modes, and safeguards against over-reliance on model predictions.

K LLM Usage

LLMs are the primary model class studied in this work for regression with sampled predictive distributions. In addition, we used LLM-based tools to assist with grammar checking, wording refinement, and figure-generation code during manuscript preparation.

L TASK PROMPTS

We present the system prompts and user instruction prompts used for each task. For all tasks, the target value is a single real number.

Note. The prompt boxes below show the exact system and instruction templates used in our experiments.

L.1 SYNTHETIC DISTRIBUTIONAL REGRESSION

System Prompt.

```
Given a synthetic regression sample, predict the target value
y.
The input x is generated from a synthetic benchmark.
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

Instruction Prompt Template.

```
Synthetic regression task.
Input feature:
x = {x}

Predict the target value y.
```

Example.

```
Synthetic regression task.
Input feature:
x = 1.23456789

Predict the target value y.
```

L.2 CODING PERFORMANCE PREDICTION

APPS System Prompt.

```
Given an APPS sample, predict the peak memory usage in bytes.
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

KBSS System Prompt.

```
Given a KBSS sample, predict the kernel execution latency in
milliseconds.
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

Instruction Prompt Template. The instruction prompt is taken directly from the `input` field:

```
{code/problem context from the dataset input field}
```

Example Structure.

```
[Problem description, code snippet, or execution context from
the dataset]
```

L.3 MoleculeNet Property Prediction

ESOL System Prompt.

```
Given a SMILES string, predict the aqueous solubility (logS).
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

FreeSolv System Prompt.

```
Given a SMILES string, predict the hydration free energy in
kcal/mol.
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

Lipo System Prompt.

```
Given a SMILES string, predict the octanol/water partition
coefficient.
Output your estimate in \boxed{}.  Only place the numeric
value inside the \boxed{}.
```

Instruction Prompt Template. The user prompt wraps the input molecule as a SMILES string:

```
<SMILES> {smiles_string} </SMILES>
```

Example.

```
<SMILES> CC(=O)OC1=CC=CC=C1C(=O)O </SMILES>
```