WEAVER Augmented: Optimizing Verifier Selection

Will Fang
Department of Computer Science
Stanford University
wfang03@stanford.edu

Amy Guan
Department of Computer Science
Stanford University
amyguan@stanford.edu

Bradley Moon
Department of Computer Science
Stanford University
bradmoon@stanford.edu

Poonam Sahoo
Department of Computer Science
Stanford University
pnsahoo@stanford.edu

Abstract

WEAVER is a framework which addresses the generation-verification gap in LLMs via combining several weak verifiers with weighted aggregation. However, WEAVER's approach relies on up to 30+ different weak verifiers, many of which are highly correlated to each other or expensive. In this paper, we propose an augmented WEAVER architecture to reduce inference compute and improve selection accuracy; we also extend this approach by incorporating difficulty routing, which enables the model to select a more suitable subset of verifiers based on problem difficulty. Our experiments show that augmented WEAVER consistently outperforms the original WEAVER as well as the original author's verifier filtering approach (subsetted WEAVER) across all datasets for small generators such as Llama 8B, and that LLM-as-a-Judge difficulty routing with augmented WEAVER provides additional accuracy gains while preserving efficiency.

1 Introduction

LLMs often generate multiple candidate responses for complex, open-ended tasks, but the act of selecting the correct answer remains a challenging problem. Saad-Falcon et. al successfully addressed this generation-verification gap via WEAVER (Saad-Falcon et al., 2025), combining multiple *weak verifiers* through weighted aggregation. However, in practice WEAVER uses up to 30+ verifiers, many of which are highly correlated and expensive.

We identified the following potential areas to augment the WEAVER approach:

1. Though WEAVER filters out low-quality verifiers, it does not select a cost-effective, minimal subset. This leads to redundant verification with correlation between verifiers, as well as unnecessary inference cost.
2. Problems within a dataset vary widely in difficulty, but WEAVER uses the same verifier set for all of them.

As a result, we focus on the following question: Is it possible to improve WEAVER performance while simultaneously reducing the amount of inference compute required? We explore this question by developing a verifier selection algorithm and extending this augmented WEAVER approach to difficulty routing methods. We first found augmented WEAVER consistently outperforms the original, bringing about improvements of 2% to 5.6% in raw accuracy when using small generators such as Llama 3.1 8B Instruct. Given that augmented WEAVER is performant for smaller generators, we extend experiments to use unsupervised LLM-as-a-judge difficulty routing with Llama 3.1 8B and find that augmented WEAVER with difficulty clustering consistently outperforms augmented WEAVER.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Related Work

As aforementioned, WEAVER aggregates multiple weak verifiers across different types (RM, PRM, LM-Judge) with learned weights from weak supervision to select the most likely correct LLM output (Saad-Falcon et al., 2025). We leverage this baseline architecture, but aim to both further reduce inference compute time and improve performance via verifier selection methods and further reduce inference-time compute. We note that the authors of WEAVER explore filtering their initial set of verifiers for the purpose of removing verifiers with skewed marginals and to encourage convergence of the objective for weak supervision (subsetted WEAVER). Our work further prunes the set of verifiers in a greedy fashion for the purpose of optimizing WEAVER performance, and we compare our approach to both the original WEAVER method and subsetted WEAVER in Table 11.

Another line of work is in designing stronger and more efficient individual verifiers. (Cobbe et al., 2021) train a single lightweight verifier to rank generated math solutions at token- or sentence-level granularity, motivating efficiency-oriented verifier architectures. Step-level reward-modeling approaches (Lightman et al., 2023a) further demonstrate that injecting fine-grained supervision improves verifier quality, highlighting how finer-grained signals can improve performance. Later work further improves on this through automatic step-level labeling (Wang et al., 2024), reducing labeling costs while still improving selection accuracy.

Many recent works have explored dynamically routing inputs to different models or ensembles, mostly for generation. (Tekin et al., 2025) dynamically select model ensembles with two RL agents, where the first learns to dynamically select a small ensemble and the other resolves their reasoning conflicts to settle on a final answer. (Guha et al., 2024) introduce SMOOTHIE, which explores how LLM routing can be accomplished with no labeled data, using a latent variable graphical model to estimate "quality" scores for each model and route accordingly. (Qi et al., 2025) adaptively route problems to strategies for enhanced mathematical reasoning, relying on multiple strategies for problems with lower predicted confidence. These works show that routing can improve performance and potentially save compute, depending on the task.

Given that Zhou et al. (2025) show that verification ability varies with difficulty, we aim to expand upon these works by pruning the verifiers in the WEAVER architecture, and also route based on problem difficulty.

3 Methods

3.1 Motivation

We propose a Penalized Greedy Selection method, inspired by forward stepwise feature selection. Through this method, we learn which verifiers to include step-by-step up to K verifiers, with the goal of both reducing inference compute requirements and improving performance.

As an extension, we propose model routing. In (Saad-Falcon et al., 2025), the authors observed increased performance via difficulty clustering with a smaller generator, Llama-8B (instead of 70B). As a result, we experimented with difficulty-aware routing using our augmented WEAVER. We hypothesized that difficulty routing could interact favorably with augmented WEAVER by allowing for a more suitable number and set of verifiers chosen based on problem difficulty.

3.2 Penalized Greedy Selection

Let u_i denote the utility of verifier i (AUC in this case, although other experiments may include mean accuracy or another metric), $S \in \mathbb{R}^{n \times n}$ the Pearson correlation between verifiers (calculated from the verifier scores for a set of generations), and $C_i \in [0, 1]$ a normalized parameter-cost for verifier i (e.g., model size in terms of parameter count, normalized to a $[0, 1]$ range).

Let $\mathcal{S}_t$ be the set of selected verifiers after step t, and let $\alpha, \beta, \gamma \geq 0$ be weights for utility, similarity penalty, and cost penalty, respectively.

Initialization. To choose the first element, we define

$$\text{score}_0(i) \;=\; \alpha\, u_i \;-\; \gamma\, C_i, \qquad i^* \;=\; \arg\max_i \text{score}_0(i). \tag{1}$$

For our experiments, we use $\text{sim\_agg}(j, \mathcal{S}_t) = \max_{k \in \mathcal{S}_t} S_{jk}$, although, empirically, we see that max and mean aggregation yield similar results.

Greedy Penalized Algorithm. For a remaining candidate $j \notin \mathcal{S}_t$,

$$\text{score}_t(j) \;=\; \alpha\, u_j \;-\; \beta\, \text{sim\_agg}(j, \mathcal{S}_t) \;-\; \gamma\, C_j, \tag{2}$$

then choose

$$j^* \;=\; \arg\max_{j \notin \mathcal{S}_t} \text{score}_t(j), \qquad \mathcal{S}_{t+1} \;=\; \mathcal{S}_t \cup \{j^*\}. \tag{3}$$

Conceptually, this is similar to a sequential/boosting algorithm, in which we greedily add the next verifier that most increases the utility of the ensemble, while also minimizing similarity between the already chosen subset of verifiers.

As part of our experiments, we tune α, β, γ and analyze how they impact the final Weaver performance.

3.3 Difficulty Clustering

We focus on Llama 3.1 8B Instruct as our generator, as Saad-Falcon et al. found that difficulty clustering exhibited larger gains in performance for the 8B model as opposed to the 70B model. To implement routing, we must assign difficulty labels to each problem. As a proof of concept, we first determine how useful difficulty routing could potentially be by assuming access to model accuracy over k generations. Following (Saad-Falcon et al., 2025), we define difficulty as the mean correctness of the model responses per question using the oracle answer and route by difficulty into buckets, whose edges are determined by an even distribution of the dev set. Then, we exhibit an unsupervised approach which uses LLM-as-a-judge to separate problems into clusters of different difficulty. Following the class labeling, we train a separate augmented WEAVER with hyperparameter search on each class of questions.

4 Experiments

4.1 Datasets

We use the same benchmarks as the original WEAVER paper, primarily aimed at mathematics, reasoning, instruction-following, and coding: **MATH500** Lightman et al. (2023b), **GPQA** Rein et al. (2023), and **MMLU College** Hendrycks et al. (2021b,a). This allows us to compare our results to the performance of the original WEAVER model without redoing the necessary expensive generations of the original WEAVER approach, which used up to several H100s for more compute-intensive weak verifiers.

MATH500 is a dataset of 500 math problems from several levels of math competitions (AMC8, AMC10, AIME) (Lightman et al., 2023b). **GPQA**, short for Graduate-Level Google-Proof Q&A Benchmark, consists of 646 problems spanning several topics in science (Rein et al., 2023). Finally, we use the same subset of 719 college-level **Massive Multitask Language Understanding (MMLU)** problems as Saad-Falcon et al. spanning from biology to computer science (Hendrycks et al., 2021b,a).

4.2 Configurations

We utilize the same 10-10-80 dev/val/test dataset split across all experiments. We evaluate augmented WEAVER against subsetted WEAVER and the original WEAVER approach across MMLU, GPQA, and MATH500 using Llama 3.1-8B Instruct as the generator.

4.2.1 Augmented WEAVER

For our approach, we first take all 20 verifiers for Llama-8B and perform a grid search over α, β, γ using the (labeled) dev set. For each set, we apply our Greedy Algorithm to prune verifiers from the space of all verifiers. We then use our validation set to determine the optimal hyperparameters, then

finally utilize the original weak supervision approach over the chosen subset of verifiers for the final evaluation. We use baselines given by Saad-Falcon et al. (2025): WEAVER with all 20 verifiers and also with the aforementioned filtering technique.

4.2.2 Difficulty Routing

For oracle clustering, we define *empirical difficulty* as the fraction of incorrect generations among all generations. We then use our dev set to determine the bucket boundaries based on empirical difficulty, choosing edges that make the dev set as evenly distributed across clusters as possible. Once these edges are fixed, we assign all remaining data to clusters according to those boundaries, and if edges across buckets are not distinct we aggregate as needed.

For our unsupervised difficulty clustering, we use Qwen2.5 7B Instruct Turbo as our LLM judge, as it is a small and inexpensive model. We prompt the judge to rate each problem in a given dataset from 1 (easiest) to 5 (most difficult). We created custom prompts for each dataset (see Appendix Section B.3 for prompts). For some datasets, this clustering approach resulted in sparse clusters, so we combined neighboring score buckets until each cluster was represented in all training and test splits.

4.3 Metrics

We report accuracy as the fraction of problems where selected response is correct across all datasets and configurations. We also report the number of verifiers selected, to demonstrate saved inference compute costs (see Appendix Section E).

5 Results

5.1 Full Augmented WEAVER Results

Overall, we found that across all 3 datasets, Augmented WEAVER outperformed both full WEAVER and subsetted WEAVER. Augmented WEAVER shows a 5.6% improvement over both baselines for MMLU, with smaller gains for the other two datasets. We additionally note that, relative to subsetted WEAVER, our hyperparameter search yielded mixed results in terms of verifier count. Qualitatively, we notice that it seems to matter more *which* verifiers are chosen; for instance, 4 of the 5 verifiers Augmented WEAVER selects are PRMs, which may be advantageous for mathematical reasoning.

Dataset	Approach	Num. Verifiers	Accuracy
MATH500	Full WEAVER	20	0.820
	Subsetted WEAVER	18	0.820
	Augmented WEAVER (k=5, β=0.5)	5	**0.840**
GPQA	Full WEAVER	20	0.312
	Subsetted WEAVER	7	0.344
	Augmented WEAVER (k=10, β=0.25)	10	**0.359**
MMLU	Full WEAVER	20	0.845
	Subsetted WEAVER	14	0.845
	Augmented WEAVER (k=10, β=0.25)	10	**0.901**

Table 1: Comparison of WEAVER variants across datasets.

5.2 Oracle Difficulty Clustering

Across all three datasets, we observe two consistent trends. First, accuracy decreases monotonically as difficulty increases, which makes sense, since oracle buckets meaningfully capture increasing challenge (Appendix Section C). Second, Augmented WEAVER performs comparably to or better than Vanilla WEAVER within most difficulty buckets while using substantially fewer verifiers. On MATH500, Augmented WEAVER achieves essentially identical accuracy to the full WEAVER model in every bucket (Table 5.2). Similarly, for GPQA, oracle routing shows Augmented WEAVER improving over Vanilla WEAVER in four out of five buckets and achieving an overall routed accuracy

of 0.422, compared to 0.395 for routed Vanilla WEAVER and 0.359 for the Augmented WEAVER baseline (Table 5.2).

With oracle routing, MMLU presents an interesting contrasting pattern. While oracle routing again produces clean monotonic difficulty curves, Augmented WEAVER underperforms both its non-routed counterpart and routed Vanilla WEAVER. This suggests that the MMLU difficulty structure is less aligned with verifier score patterns, making bucket-specific verifier subsets less effective. Whereas MATH500 and GPQA contain homogeneous problem types where difficulty strongly correlates with verifier agreement, MMLU's diverse subject matter may reduce the benefit of difficulty-based routing.

Dataset	Approach	Num. Verifiers	Accuracy
MATH 500	LLM-Judge Diff. Routing (Vanilla WEAVER)	20	0.760
	LLM-Judge Diff. Routing (Augmented WEAVER)	5,5,5,10	**0.860**
	Oracle Diff. Routing (Vanilla WEAVER)	20	0.759
	Oracle Diff. Routing (Augmented WEAVER)	10, 5, 10, 5, 5	0.759
	Augmented WEAVER	5	0.840
GPQA	LLM-Judge Diff. Routing (Vanilla WEAVER)	20	0.304
	LLM-Judge Diff. Routing (AugmentedWEAVER)	10,10,15	0.396
	Oracle Diff. Routing (Vanilla WEAVER)	20	0.395
	Oracle Diff. Routing (Augmented WEAVER)	10, 15, 15, 15, 15	**0.422**
	Augmented WEAVER	10	0.359
MMLU	LLM-Judge Diff. Routing (Vanilla WEAVER)	20	0.755
	LLM-Judge Diff. Routing (Augmented)	5,5,10	0.829
	Oracle Diff. Routing (Vanilla WEAVER)	20	0.853
	Oracle Diff. Routing (Augmented WEAVER)	5, 10, 10, 5, 5	0.804
	Augmented WEAVER	10	**0.901**

Table 2: Comparison of unsupervised difficulty clustering to Augmented WEAVER across datasets. The number of verifiers for the augmented WEAVER difficulty approach is reported in order of difficulty level, from least to most difficult. Bold indicates best performance within each dataset.

5.3 Unsupervised Difficulty Routing

Overall, Augmented WEAVER combined with difficulty routing outperforms the original Augmented WEAVER method for MATH 500 and GPQA by 2.3% and 10.3%, although it underperforms by 8% on MMLU College (Table 5.2). See Appendix Section D for per-cluster accuracy results, which consistently show Augmented WEAVER outperforming Vanilla WEAVER on each cluster with difficulty routing. As the difficulty increases, the performance decreases, as expected. However, notably, the *improvement* in performance with difficulty routing of Augmented WEAVER over Vanilla WEAVER increases as well. This is especially evident in a dataset such as MATH500, where the LLM performs relatively well on difficulty labeling (Figure 1).

One thing we qualitatively notice is that, for the cluster of highest difficulty, the hyperparameter search for Augmented WEAVER yields a larger number of verifiers being selected (Table 5.2). This yields improvements in accuracy for MATH 500 and GPQA while still maintaining a lower number of verifiers. For MATH 500 and GPQA, we also note that, for each cluster-specific WEAVER, we maintain an equal or greater number verifiers compared to Augmented WEAVER without routing. However, with MMLU, Augmented WEAVER selects 10 verifiers, while buckets 1 and 2 with routing only select 5 verifiers, which could contribute to the decreased performance.

It is also worth noting that the LLM-judged difficulties are only a rough approximation of actual problem difficulty; for MATH 500, the only dataset with "ground truth" problem difficulties, the correlation between the LLM annotations and the ground truth annotations was 0.7285, with an exact match of only 16.67%.

Figure 1: WEAVER + LLM-Judge difficulty routing accuracy by difficulty for MATH500

We also include box plots and correlation metrics for the LLM-as-a-Judge vs empirical accuracy in Appendix Section B. The MATH 500 dataset shows a much clearer trend in Oracle problem success rate versus difficulty bucket (Table 4). For MATH 500, the LLM-determined difficulty buckets roughly align with the Oracle success percentile, whereas the LLM labeling less clearly correlates with decreasing empirical success on GPQA and MMLU.

6 Discussion and Conclusion

We demonstrate that, with Augmented WEAVER, carefully selecting a subset of verifiers can yield substantial improvements. Verifier choice matters, both with regards to inter-verifier correlation and problem domain; for instance, having a smaller set of mostly PRMs for mathematical reasoning proved to be advantageous for the MATH500 dataset.

Generally, oracle routing appears to show that when difficulty is accurately identified, routing can improve accuracy. In particular Augmented WEAVER with oracle difficulty clustering seems to achieve the largest gains precisely in datasets where difficulty is well structured and well aligned with verifier behavior.

For LLM-as-a-Judge routing with WEAVER, we suprisingly surpass the performance of the oracle routing with WEAVER. However, when examining the per-difficulty bucket performance, we see that augmented WEAVER and oracle routing struggled especially with the most difficult bucket (Appendix 8), which was often collapsed into the next highest bucket with LLM-as-a-Judge (Appendix B.2). We hypothesize that the extreme nature of the most difficult problems, perhaps without suitable homogeneity to leverage, along with verifier count, contributed to this difference.

Overall, we see that LLM-as-a-Judge routing combined with augmented WEAVER training is able to implicitly leverage difficulty during training by selecting more verifiers for more difficult problems. Notably, the *improvement* in performance increases with increasing difficulty. Thus, there is significant potential in this approach to improve performance while maintaining efficiency across difficulty strata, effectively mitigating the generation verification gap.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

Neel Guha, Mayee F. Chen, Trevor Chow, Ishan S. Khare, and Christopher Ré. 2024. Smoothie: Label Free Language Model Routing. arXiv:2412.04692 [cs.AI] `https://arxiv.org/abs/2412.04692`

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning AI With Shared Human Values. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050* (2023).

Shihao Qi, Jie Ma, Ziang Yin, Lingling Zhang, Jian Zhang, Jun Liu, Feng Tian, and Tongliang Liu. 2025. Plan before Solving: Problem-Aware Strategy Routing for Mathematical Reasoning with LLMs. arXiv:2509.24377 [cs.AI] https://arxiv.org/abs/2509.24377

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. arXiv:2311.12022 [cs.AI] https://arxiv.org/abs/2311.12022

Jon Saad-Falcon, E Kelly Buchanan, Mayee F Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, et al. 2025. Shrinking the Generation-Verification Gap with Weak Verifiers. *arXiv preprint arXiv:2506.18203* (2025).

Selim Furkan Tekin, Fatih Ilhan, Gaowen Liu, Ramana Rao Kompella, and Ling Liu. 2025. Dynamic Optimizations of LLM Ensembles with Two-Stage Reinforcement Learning Agents. arXiv:2502.04492 [cs.CL] https://arxiv.org/abs/2502.04492

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9426–9439.

Yefan Zhou, Austin Xu, Yilun Zhou, Janvijay Singh, Jiang Gui, and Shafiq Joty. 2025. Variation in Verification: Understanding Verification Dynamics in Large Language Models. arXiv:2509.17995 [cs.CL] https://arxiv.org/abs/2509.17995

A Augmented WEAVER Results for LLama-8B and Llama-70B

Dataset	Approach	Num. Verifiers	Accuracy
math500-llama70b	**Full WEAVER**	36	0.940
	Subsetted WEAVER	10	0.940
	Augmented WEAVER (k=10, β=0.25)	10	**0.940**
math500-llama8b	Full WEAVER	20	0.820
	Subsetted WEAVER	18	0.820
	Augmented WEAVER (k=5, β=0.5)	5	**0.840**
gpqa-llama70b	**Full WEAVER**	36	**0.688**
	Subsetted WEAVER	15	**0.688**
	Augmented WEAVER (k=10, β=0.5)	10	0.625
gpqa-llama8b	Full WEAVER	20	0.312
	Subsetted WEAVER	7	0.344
	Augmented WEAVER (k=10, β=0.25)	10	**0.359**
mmlu-llama70b	Full WEAVER	36	0.944
	Subsetted WEAVER	22	0.944
	Augmented WEAVER (k=5, β=0.25)	5	**0.958**
mmlu-llama8b	Full WEAVER	20	0.845
	Subsetted WEAVER	14	0.845
	Augmented WEAVER (k=10, β=0.25)	10	**0.901**

Table 3: Comparison of WEAVER variants across datasets.

B Difficulty Routing

B.1 LLM Labels vs Oracle Difficulty

We include plots that display the performance of the LLM-as-a-Judge compared to the actual empirical difficulty.

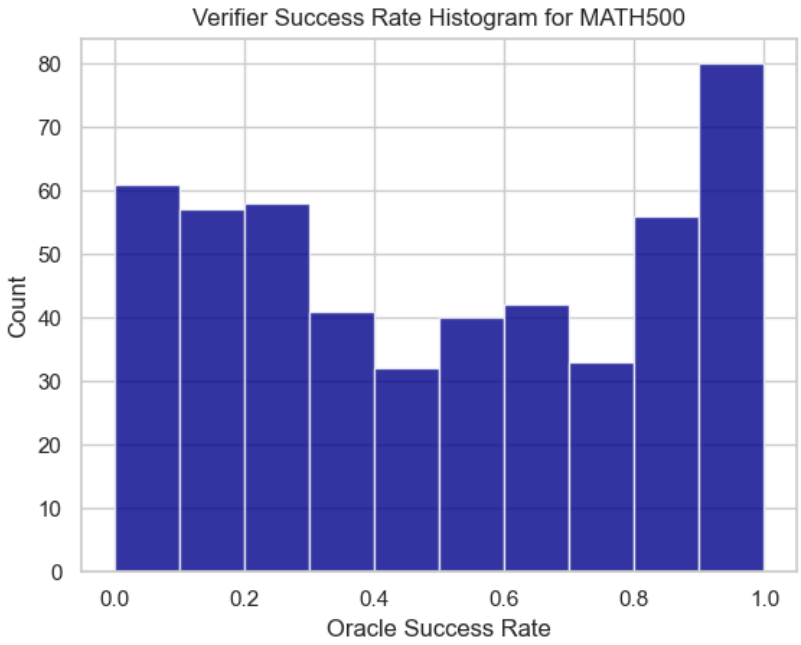

Figure 2: Oracle Difficulty Distribution for GPQA

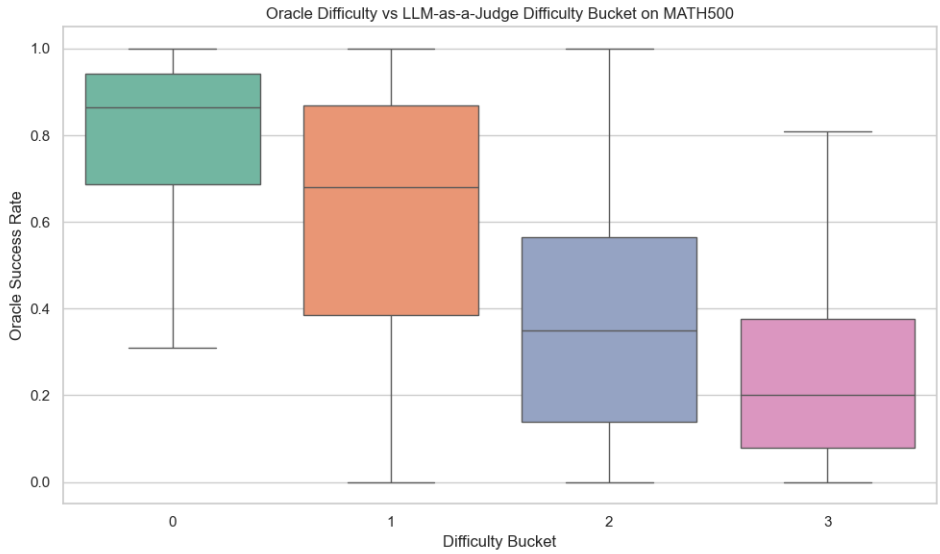

Figure 3: LLM Judge vs Oracle Difficulty for GPQA

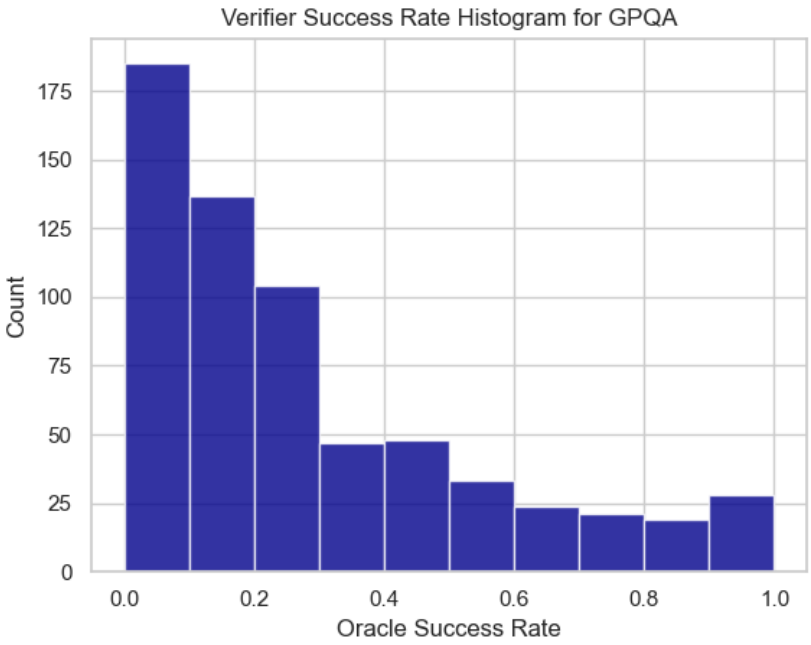

Figure 4: Oracle Difficulty Distribution for GPQA

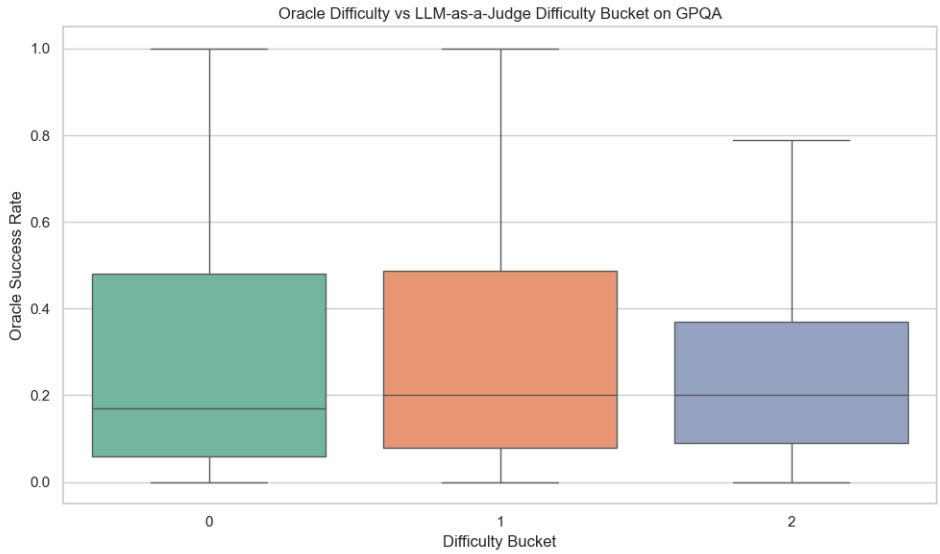

Figure 5: LLM Judge vs Oracle Difficulty for GPQA

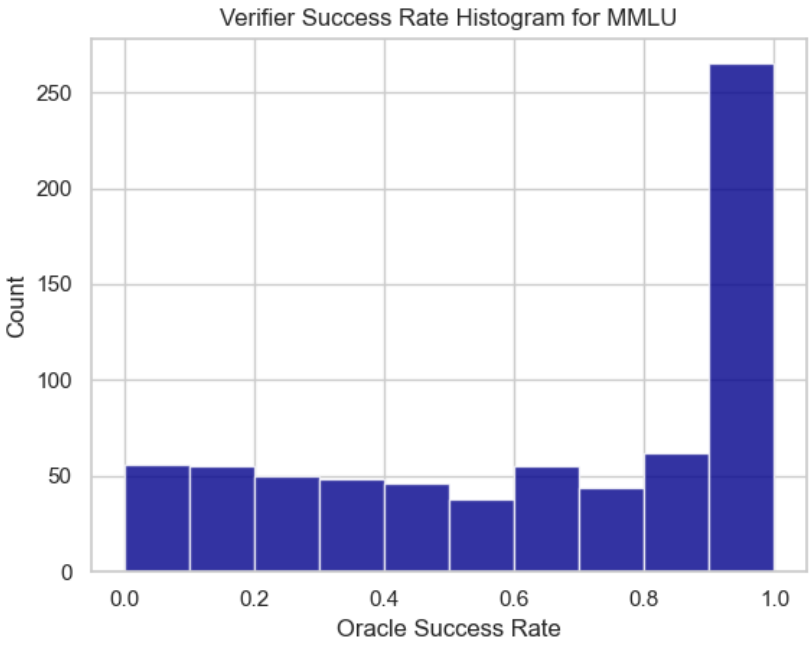

Figure 6: Oracle Difficulty Distribution for MMLU

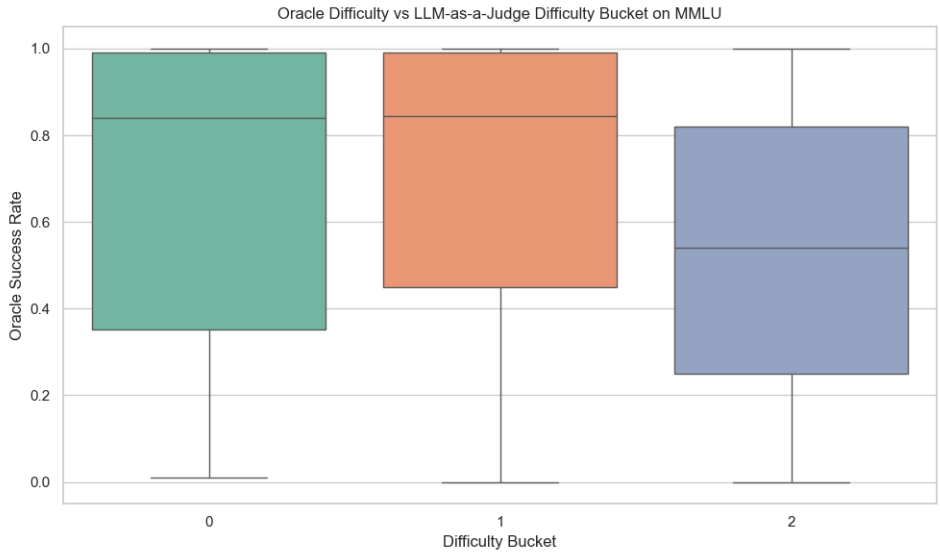

Figure 7: LLM Judge vs Oracle Difficulty for MMLU

Dataset	Spearman Corr.	Kendall Corr.
MATH500	-0.5604	-0.4375
GPQA	-0.0107	-0.0094
MMLU	-0.2357	-0.1902

Table 4: Correlations between LLM-as-a-Judge bucket labels and empirical (oracle) accuracy.

B.2 Combining Clusters

For MATH500, we combined clusters 4 and 5 into one difficulty bucket. For GPQA and MMLU College, we combined clusters 1 and 2 into one bucket as well as clusters 4 and 5.

B.3 LLM Judge Prompts

B.3.1 MMLU

```
You are an expert evaluator of problem difficulty across many academic domains (science, math, humanities, social science, law, medicine, etc.), similar to the MMLU benchmark.

Rate the problem's difficulty on a scale from 1 to 5. Use the full scale.

Difficulty Anchors:

Level 1: Very easy. Can be answered by direct recall of a fact, definition, or single-step computation. No reasoning required.

Level 2: Easy–moderate. Requires understanding a basic concept or recognizing a straightforward pattern; 1–2 simple reasoning steps.

Level 3: Moderate. Requires multi-step reasoning, combining multiple ideas, or interpreting information; typical exam-level difficulty.

Level 4: Hard. Requires deeper conceptual understanding, multi-part reasoning, abstraction, or applying nontrivial domain knowledge.

Level 5: Very hard. Requires expert-level insight, advanced understanding of the domain, or extended reasoning comparable to upper-division or graduate-level exams.

Problem: {problem_text}

Respond with only a single integer (1–5). Use the FULL range of the scale from 1-5 to rate this problem.
```

B.3.2 GPQA

```
You are an expert evaluator of advanced academic problem difficulty across all scientific and technical domains (e.g., mathematics, physics, computer science, chemistry, engineering), similar to the GPQA benchmark.

Rate the problem's difficulty on a scale from 1 to 5. Use the full scale.

Difficulty Anchors:

Level 1: Routine undergraduate-level question requiring direct application of a definition or standard fact.

Level 2: Upper-division undergraduate or early graduate problem requiring multiple steps but using well-known methods.

Level 3: Graduate-level homework difficulty; requires sustained reasoning, abstraction, or combining several advanced ideas.

Level 4: Very advanced; comparable to a difficult graduate qualifying exam problem or a challenging research-preparation assignment.

Level 5: Expert-level. Requires research-level insight, nonstandard techniques, deep conceptual understanding, or sophisticated multi-part reasoning.

Problem: {problem_text}

Respond with only a single integer (1–5). Use the FULL range of the scale from 1-5 to rate this problem.
```

B.3.3 MATH500

You are an expert at evaluating the difficulty of mathematical competition problems.

Your task is to rate the difficulty of a given problem on a scale from 1 to 5, where:

- Level 1 - Easiest problems, typically solvable by strong high school students with basic competition training

- Level 2 - Moderate difficulty, requiring more mathematical maturity

- Level 3 - Challenging problems, comparable to mid-level competition questions

- Level 4 - Very difficult problems, approaching olympiad-level difficulty

- Level 5 - Hardest problems, comparable to International Mathematical Olympiad (IMO) level questions

Here are some examples:

Example 1: Problem: A recipe calls for $1/4$ teaspoon of salt for each quart of water. How many quarts of water will be used for two teaspoons of salt?

Difficulty: 1

Example 2:Problem: Find $(-1)^{-10} + (-1)^{-9} + (-1)^{-8} + \cdots + (-1)^9 + (-1)^{10}$.(The dots $\cdots$ mean that there are 21 numbers being added, one for each integer from -10 to 10.)

Difficulty: 2

Example 3:Problem: Evaluate $\log_3 \frac{1}{\sqrt{3}}$.

Difficulty: 3

Example 4:Problem: What value of x will give the minimum value for $9x^2 + 18x + 7$?

Difficulty: 4

Example 5:Problem: A parabola $ax^2 + bx + c$ contains the points $(-1, 0)$, $(0, 5)$, and $(5, 0)$. Find the value $100a + 10b + c$.

Difficulty: 5

Use the FULL range of the scale from 1-5 to rate this problem.

Respond with only a single integer.

Problem: {problem_text}

Difficulty (1-5):

C Per-Cluster Oracle Difficulty Clustering Results

Difficulty Level	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	**1.000**	10	**1.000**
1	20	**0.980**	5	0.964
2	20	0.920	10	**0.959**
3	20	**0.850**	5	0.802
4	20	**0.120**	5	0.068

Table 5: Comparison of Oracle Difficulty Clustering per-cluster for MATH500

Difficulty	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	0.824	10	**0.861**
1	20	0.568	15	**0.580**
2	20	0.347	15	**0.413**
2	20	0.233	15	**0.240**
2	20	**0.056**	15	0.016

Table 6: Comparison of Oracle Difficulty Clustering per-cluster for GPQA

Difficulty	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	**1.000**	5	**1.000**
1	20	**1.000**	10	**1.000**
2	20	**0.980**	10	**0.980**
3	20	0.712	5	0.**740**
4	20	**0.345**	5	0.302

Table 7: Comparison of Oracle Difficulty Clustering per-cluster for MMLU

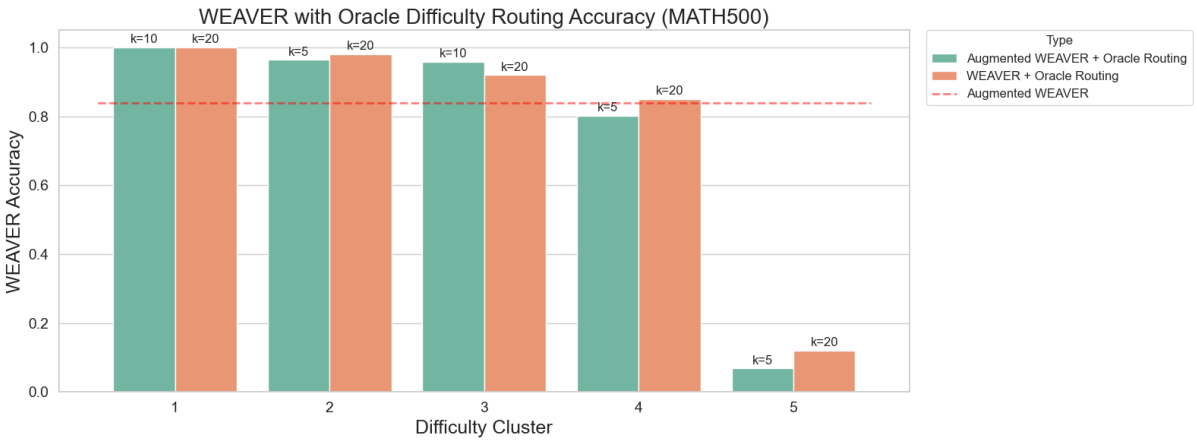

Figure 8: Weaver with oracle difficulty routing accuracy by difficulty for MATH500

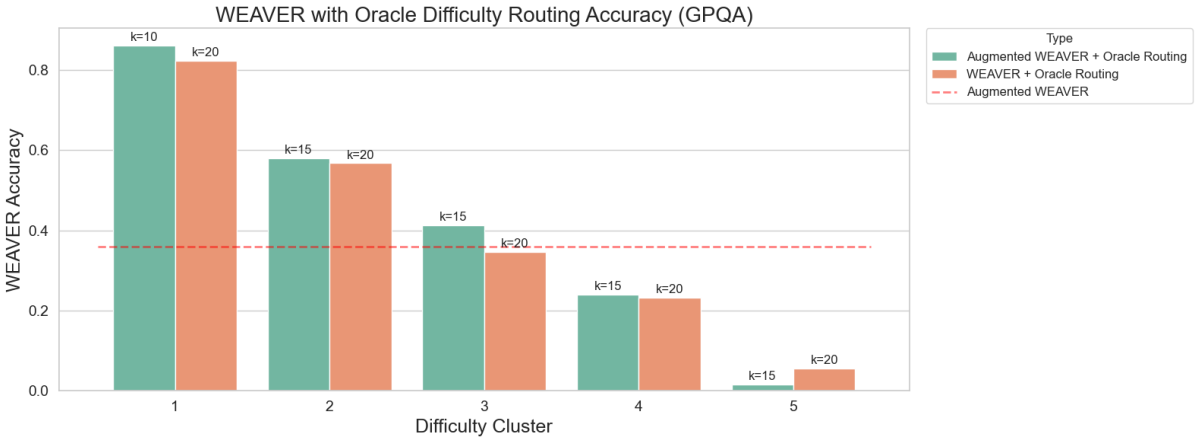

Figure 9: Weaver with oracle difficulty routing accuracy by difficulty for GPQA

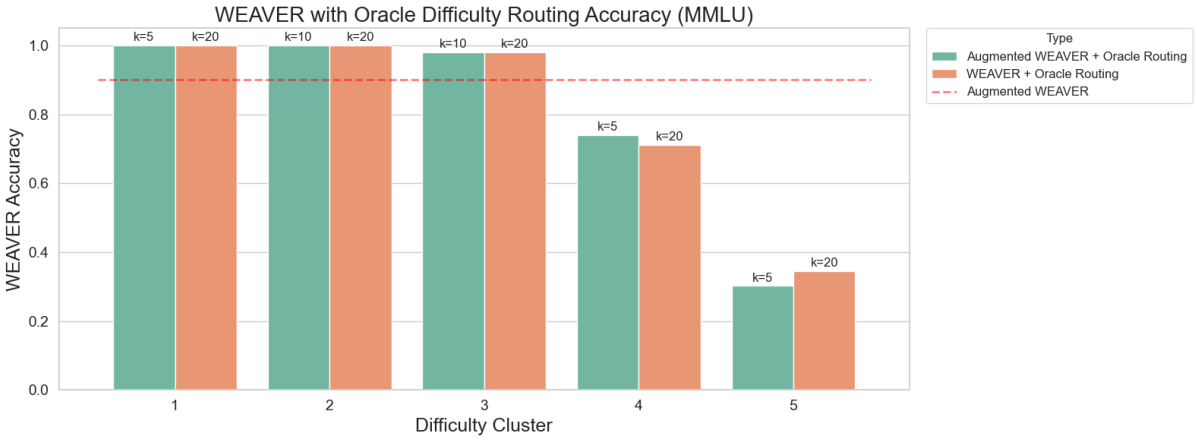

Figure 10: Weaver with oracle difficulty routing accuracy by difficulty for MMLU

D Per-Cluster Unsupervised Difficulty Clustering Results

Difficulty Level	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	0.906	5	**0.969**
1	20	**0.930**	5	0.916
2	20	0.728	5	**0.880**
3	20	0.455	10	**0.673**

Table 8: Comparison of Unsupervised Difficulty Clustering per-cluster for MATH500

Difficulty	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	0.328	10	**0.406**
1	20	0.232	10	**0.370**
2	20	0.346	15	**0.412**

Table 9: Comparison of Unsupervised Difficulty Clustering per-cluster for GPQA

Difficulty	Vanilla WEAVER		Augmented WEAVER	
	Num. Verifiers	Accuracy	Num. Verifiers	Accuracy
0	20	0.742	5	**0.806**
1	20	0.808	5	**0.874**
2	20	0.673	10	**0.806**

Table 10: Comparison of Unsupervised Difficulty Clustering per-cluster for MMLU

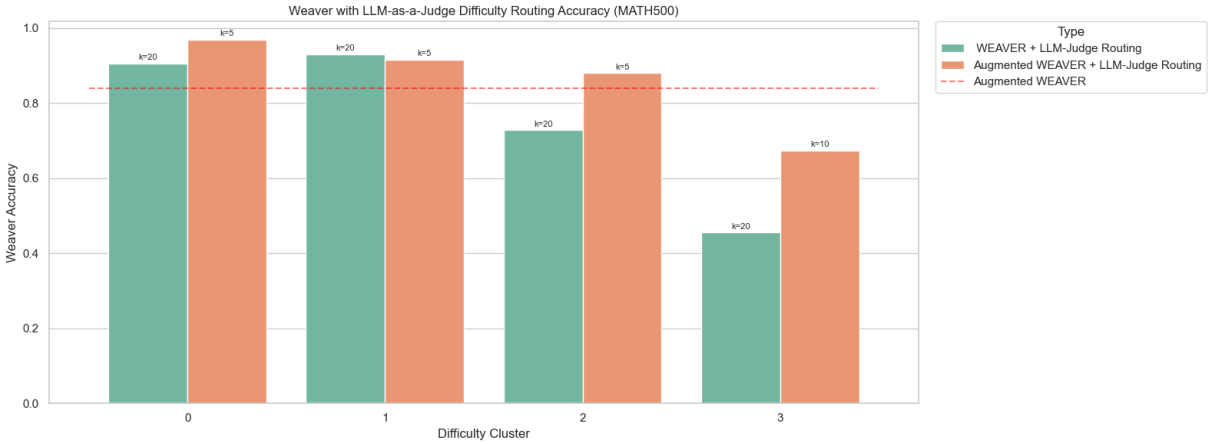

Figure 11: Weaver with difficulty routing accuracy by difficulty for MATH500

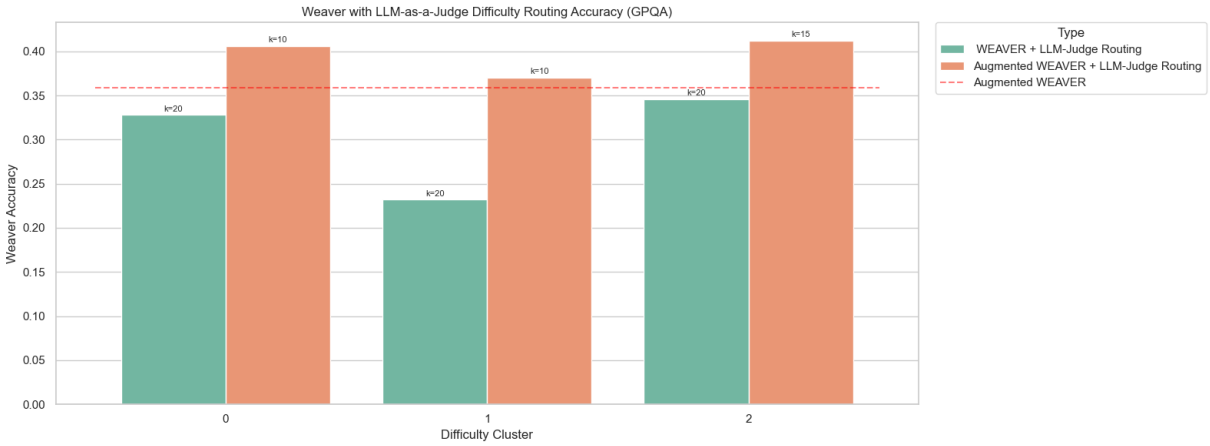

Figure 12: Weaver with difficulty routing accuracy by difficulty for GPQA

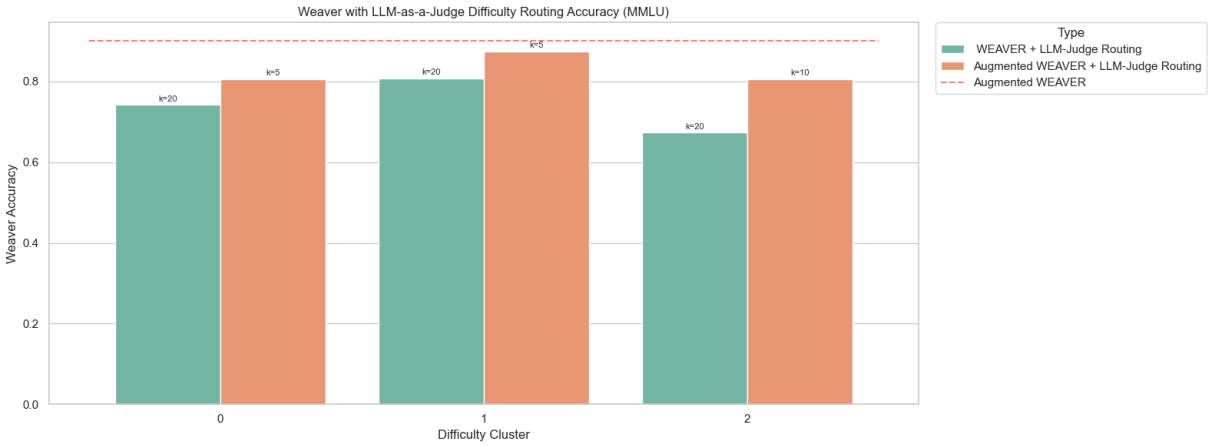

Figure 13: Weaver with difficulty routing accuracy by difficulty for MMLU

E Inference Time Costs

Vanilla WEAVER uses up to 20 verifiers for the 8B case, which we studied. Let T, V, D be the number of problems in the test, validation, and development sets, respectively. We can assume in the worst case that our grid search uses all verifiers (but we need only compute these verifier scores once). Suppose that we choose k verifiers, and there are n generations to verify. Then we can compute the inference cost as

$$20 \times D \times n + 20 \times V \times n + k \times T \times n$$

We use an $80/10/10$ split, so let $D = V = 0.1Q$, and $T = 0.8Q$, where Q is the total number of questions. Then we an inference compute of

$$(4 + 0.8k)Qn$$

as opposed to the Vanilla WEAVER inference compute of $20Qn$. In particular notice that when $k < 20$, we get strictly improved inference compute costs. Note that for 36 verifiers, this number is $(7.2 + 0.8k)Qn$. In the table below, all improvements are calculated with respect to the full Vanilla WEAVER, and not subsetted WEAVER which saves even more compute than augmented WEAVER on 1/3 of the datasets.

Note that subsetted WEAVER inference costs are computed as follows: For 8B $(2 + 0.9k)Qn$ and for 70B $(3.6 + 0.9k)Qn$.

Dataset	Approach	Num. Verifiers	Accuracy	Compute Cost (Qn)
math500-llama70b	**Full WEAVER**	36	0.940	36
	Subsetted WEAVER	10	0.940	**12.6**
	Augmented WEAVER	10	**0.940**	15.2 (58% saved)
math500-llama8b	Full WEAVER	20	0.820	20
	Subsetted WEAVER	18	0.820	18.2
	Augmented WEAVER	5	**0.840**	**8** (60% saved)
gpqa-llama70b	**Full WEAVER**	36	**0.688**	36
	Subsetted WEAVER	15	**0.688**	17.1
	Augmented WEAVER	10	0.625	**15.2** (58% saved)
gpqa-llama8b	Full WEAVER	20	0.312	20
	Subsetted WEAVER	7	0.344	**8.3**
	Augmented WEAVER	10	**0.359**	12 (40% saved)
mmlu-llama70b	Full WEAVER	36	0.944	36
	Subsetted WEAVER	22	0.944	23.4
	Augmented WEAVER	5	**0.958**	**11.2** (69% saved)
mmlu-llama8b	Full WEAVER	20	0.845	20
	Subsetted WEAVER	14	0.845	14.6
	Augmented WEAVER	10	**0.901**	**12** (40% saved)

Table 11: Comparison of Augmented WEAVER inference compute costs across datasets.