LLMs Encode How Difficult Problems Are

Anonymous ACL submission

Abstract

Large language models exhibit a puzzling inconsistency: they solve complex problems yet frequently fail on seemingly simpler ones. We investigate whether LLMs internally encode problem difficulty in a way that aligns with human judgment, and whether this representation tracks generalization during reinforcement learning post-training. We train linear probes across layers and token positions on 60 models, evaluating on mathematical and coding subsets of Easy2HardBench. We find that human-labeled difficulty is strongly linearly decodable (AMC: $\rho \approx 0.88$) and exhibits clear model-size scaling, whereas LLM-derived difficulty is substantially weaker and scales poorly. Steering along the difficulty direction reveals that pushing models toward "easier" representations reduces hallucination and improves accuracy. During GRPO training on Qwen2.5-Math-1.5B, the human-difficulty probe strengthens and positively correlates with test accuracy across training steps, while the LLM-difficulty probe degrades and negatively correlates with performance. These results suggest that human annotations provide a stable difficulty signal that RL amplifies, while automated difficulty estimates derived from model performance become misaligned precisely as models improve. We release probe code and evaluation scripts to facilitate replication.

Question. Let $(a_1, \ldots, a_n)$ be positive reals with $\sum a_i = 96$, $\sum a_i^2 = 144$, $\sum a_i^3 = 216$. *Find the sum of all possible values of n.*

Steered $\alpha = -3$

Response: "Let's solve step-by-step using `Python` / `SymPy`. Given $\sum a_i = 96$, $\sum a_i^2 = 144$, $\sum a_i^3 = 216$, we solve for admissible n via Cauchy–Schwarz / power-mean constraints… … 64. (Use `import sympy as sp`, `factor()`, `solve()` to verify.)"

Parsed answer: 64 **Response length:** 1298

Baseline $\alpha = 0,\ +3$

Response: legitim legitim legitim legitim legitim legitim legitim legitim legitim legitim …

Parsed answer: none **Response length:** 3000

Figure 1: Steering towards "easy" reduces hallucination-like repetition in Qwen2.5-Math-1.5B. With $\alpha = -3$, the model switches from repetitive output (baseline / $+3$) to concise, code-assisted reasoning and a parsed correct answer (64).

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance on complex tasks. However, they exhibit a fundamental inconsistency: they solve challenging problems while frequently failing at seemingly simpler ones (Yang et al., 2024; Williams and Huckle, 2024; Wang and Sun, 2025). This inconsistency manifests in various forms, from failures in basic logical reasoning to difficulties with symmetric relationships. As LLMs are deployed in agentic and high-risk applications, understanding and mitigating this inconsistency becomes increasingly critical.

A natural question thus arises: do LLMs internally represent problem difficulty in a way that aligns with human judgment? While LLMs fail when explicitly prompted to estimate problem difficulty (Ding et al., 2024) and struggle to calibrate their confidence on challenging questions (Kadavath et al., 2022), this does not mean that difficulty is absent from their internal representations. Recent work has demonstrated that high-level concepts such as truthfulness and sentiment are linearly encoded in LLM activations (Mikolov et al., 2013; Park et al., 2024), suggesting that difficulty

might similarly exist as a recoverable direction in the model's representation space—even if models do not articulate it when prompted.

In this work, we investigate whether LLMs encode a notion of problem difficulty that aligns with human judgment, examine the effects of steering along this direction, and track how this representation evolves during Reinforcement Learning from Verifiable Rewards (RLVR) for mathematical reasoning. We train linear probes across 60 models and evaluate their difficulty estimation performance on mathematical and coding subsets from Easy2HardBench (Ding et al., 2024).

Our key findings are:

- Human-derived difficulty ratings are strongly and linearly decodable from model activations, while LLM-derived difficulty ratings show substantially weaker encoding.
- Steering along the difficulty direction reveals that pushing models toward "easier" representations reduces output length and improves accuracy by preventing hallucination.
- During GRPO training, human-difficulty representations strengthen alongside model capabilities, whereas LLM-difficulty representations degrade—suggesting that human annotations capture a stable signal that RL amplifies, while automated difficulty estimates become increasingly misaligned as models improve.

2 Background

2.1 Linear Probes

Substantial previous work has shown that certain high-level features such as truthfulness and political perspectives, may be linearly represented in the activations of an LLM (Mikolov et al., 2013; Park et al., 2024; Kim et al., 2025). Linear probes are simple models trained on activations extracted from an LLM at a given timestep t to predict a "ground truth" label y. Some existing work advocates for probing the outputs of individual attention heads and fitting a separate probe to each head (Kim et al., 2025). A contrasting approach is to probe the outputs of the concatenated attention outputs from the attention block at each layer of the model. (Kim et al., 2025)

2.2 Scaling Laws

Previous research on scaling laws has demonstrated that architectural and training regime differences lead to divergent behaviours for similar model sizes across different model families (Kaplan et al., 2020). Consequently, we conduct our probing study across five major model families—Qwen2.5, Qwen3, LLaMA 3.1, DeepSeek (Llama Team, 2024; DeepSeek-AI et al., 2025)—to observe whether difficulty is linearly encoded among them.

2.3 Difficulty estimation

Traditional difficulty estimation relies on domain-specific characteristics like language similarity in linguistic reasoning (Beeching et al., 2023), equation complexity in mathematical problems (Yang et al., 2024), or human annotations (Rein et al., 2023).

More recently, Ding et al. (2024) introduced Easy2Hard-Bench, a dataset that aggregates multiple benchmarks spanning various difficulty levels. The dataset provides standardized, human-grounded difficulty labels across six subsets: E2H-AMC, E2H-Codeforces, E2H-Lichess, E2H-GSM8K, E2H-Winogrande, and E2H-ARC. These difficulty ratings are derived from two primary sources: human performance statistics from competitive leaderboards (for E2H-AMC, E2H-Codeforces, and E2H-Lichess) and LLM performance data from the Open LLM Leaderboard (Beeching et al., 2023) (for E2H-GSM8K, E2H-ARC, and E2H-Winogrande). For the AMC subset specifically, difficulty is estimated using Item Response Theory (IRT) models fitted on multiple attempts by humans.

2.4 RLVR

Popular LLM's such as OpenAI-o1 (OpenAI), DeepSeek-R1 (DeepSeek-AI et al., 2025), and Kimi-1.5 (Team et al., 2025) have used Reinforcement Learning (RL) as a post-training approach to enhance the reasoning capabilities of models and consequently improve their performance on complex mathematical tasks. *Reinforcement Learning with Verifiable Rewards* (RLVR) (DeepSeek-AI et al., 2025; Gao et al., 2024; Wang et al., 2025) is a key method central to these advances in which RL is incorporated in an LLM with a rule-based outcome reward, such as a binary reward for the accuracy of a model's final answer to a math problem or following a template of expected answers. RLVR is commonly optimised with policy-gradient methods such as PPO and GRPO.

Recent work therefore targets *selective sampling*

by difficulty variability. Wang et al. (2025) demonstrate that performance comparable to or better than GRPO with full-dataset training can be achieved with one or two carefully selected problems, chosen using a *historical variance* score that tracks fluctuations in the model's success rate across steps. Complementarily, Foster et al. (2025) propose LILO, which measures *within-step* success variability to surface challenging items online. Both strategies rely on observing variance—either across steps (RLVR) or within a step (LILO)—and typically require an initial warm-up or logging pass to identify high-variance problems.

3 Experiments

3.1 Probing for Difficulty

Data We construct our probing datasets using two mathematical subsets from Easy2HardBench (Ding et al., 2024): E2H-AMC and E2H-GSM8K. Each problem has a difficulty score derived from Item Response Theory (IRT) modeling, where higher scores indicate greater difficulty. The key difference between the subsets is that E2H-AMC uses human success rates to calibrate the difficulty, while E2H-GSM8K uses LLM success rates. For each subset, we randomly sample 500 questions from the original training set to create our probing dataset $\mathcal{D}_{\text{train}}$.

Models We extract activations from eighteen model families across 60 variants: DeepSeek, Qwen-2.5, Qwen-Math, Llama, Qwen-3, FineMath, OpenMath, Qwen-Coder. To assess whether domain-specific fine-tuning affects difficulty representations, we include math-specialized variants—FineMath and OpenMath (Llama-3.1 based) and Qwen-Math (Qwen-2.5 based)—for the AMC and GSM8K subsets. We also include reasoning-focused models (DeepSeek and Qwen-3) to examine whether enhanced reasoning capabilities yield better difficulty representations. For the coding dataset, we add two code-specialized Qwen-Coder variants (OpenHands and SWE-Agent) following the same rationale. A complete list of models is provided in Appendix 3.

Linear Probe Setup For each problem, we apply the model's default chat template and extract activations from all tokens following the final prompt token (post-instruction tokens). At each post-instruction token position, we extract activations from every model layer l.

For each problem, we format the input with the model's chat template and collect activations from every token after the final prompt token ("post-instruction" tokens). Therefore, at each such position, p, we extract the hidden states from every layer l.

Evaluation. We evaluate all linear probes using 5-fold cross-validation on the training split $\mathcal{D}_{\text{train}}$. For each fold, we train a probe on 80% of the data and evaluate on the remaining 20%. The metric is Spearman rank correlation between predicted and gold difficulty. For each (layer, position) probe we report the mean across folds, denoted $\widehat{\rho}^{\text{CV}}_{\ell,p}$.

To assess the effect of model size on performance, we fit a model's best $\widehat{\rho}^{\text{CV}}_{\ell,p}$ to the following power law:

$$1 - \widehat{\rho}^{\text{CV}}_{\ell,p} = C \cdot N^{-\alpha}$$

where N is the number of model parameters, α is the scaling exponent (with $\alpha > 0$ indicating performance improves with model size), and C is a normalization constant. This formulation models the gap to perfect performance, allowing performance to asymptotically approach 1.0 as model size increases. We perform the fit in log-log space by regressing $\log(1 - \text{perf})$ against $\log(N)$, ensuring numerical stability through epsilon clipping. We report R^2_{log} and α, which measure the proportion of variance in the log-transformed performance gap explained by log model size.

Assuming that we observe strong performance of our linear probes, we can assume that a linear "easy-hard" axis exists within LLM representations. Following (Li et al., 2024), we can then use the fitted regression coefficients from our trained probes to steer generation during inference and observe how this affects task performance. In our case, we will define a steering coefficient grid ranging from $\alpha \in [-3, +3]$, and our metric of interest will be Pass@1.

3.1.1 Results

Human Difficulty is Linearly Encoded in LLM Activations As shown in Table 1, the highest probe performance is achieved with activations extracted from the largest models in our suite, with E2H-AMC reaching $\widehat{\rho}^{\text{CV}}_{\ell,p} = 0.88$, followed by E2H-Codeforces ($\widehat{\rho}^{\text{CV}}_{\ell,p} = 0.81$) and E2H-GSM8K ($\widehat{\rho}^{\text{CV}}_{\ell,p} = 0.58$). Notably, all top-performing probes are trained using activations from Llama-based

Data	Model	$\widehat{\rho}^{\mathrm{CV}}_{\ell,p}$	Layer	Pos
AMC	**DeepSeek-Llama-70B**	**0.8842**	**38**	**-1**
	Qwen2.5-72B-Instruct	0.8799	61	-2
	Qwen2.5-Math-72B	0.8780	74	-1
CF	**Llama-3.1-70B-Instruct**	**0.8115**	**42**	**-1**
	DeepSeek-Llama-70B	0.7980	41	-1
	Llama-3.1-8B-Instruct	0.7571	16	-1
G8K	**Llama-3.1-70B-Instruct**	**0.5799**	**35**	**-1**
	DeepSeek-Llama-70B	0.5639	38	-3
	OpenMath2-Llama3.1-8B	0.5516	15	-5

Table 1: Top-3 probe performance scores per dataset. Probe Performance is the average Spearman Rank correlation over 5 folds.Data abbreviations: AMC = E2H-AMC, CF = E2H-Codeforces, G8K = E2H-GSM8K. The results for all models can be observed in the Appendix 4

models, with the best AMC probe using the R1-distilled variant.

The substantial performance gap (≈ 0.3) between E2H-AMC and E2H-GSM8K reveals that human difficulty ratings are better encoded in model activations than LLM-derived difficulty estimates. One potential explanation is that difficulty scores derived from LLM problem-solving attempts are inherently noisy due to the high variance in LLM performance on mathematical problems. However, the contrast between E2H-AMC and E2H-GSM8K probes suggests a more nuanced picture: models encode representations of human-perceived difficulty while struggling to predict how difficult other LLMs will find the same problems. This aligns with prior work showing that LLMs perform poorly when explicitly prompted to predict problem difficulty. Our contribution extends this finding by demonstrating that this limitation persists at the representational level for both mathematical and coding tasks.

Difficulty representations scale with model size Figure 2 demonstrates that probe performance ($\widehat{\rho}^{\mathrm{CV}}_{\ell,p}$) follows a power law with model parameters across all three datasets. The scaling strength varies by dataset: E2H-Codeforces shows the strongest scaling ($\alpha = 0.066$), suggesting that larger models develop better representations of coding difficulty. E2H-AMC exhibits moderate scaling ($\alpha = 0.045$), with probe scores improving from 0.82 to 0.89 as model size increases. E2H-GSM8K shows the weakest scaling ($\alpha = 0.020$), indicating that model size alone provides limited benefit for LLM-derived difficulty scores.

The scaling patterns reveal an important distinction between human-labeled and LLM-labeled difficulty. Datasets with human difficulty labels (E2H-AMC and E2H-Codeforces) exhibit clearer scaling trends than those with LLM-derived labels (E2H-GSM8K), suggesting that models more consistently encode human-perceived difficulty in their representations. The weak scaling for GSM8K likely stems from task-specific bottlenecks in individual models, as seen in Llama-3.1-32B-Base in Figure 2, that prevent larger architectures from forming better difficulty representations. This interpretation is supported by the fit quality: E2H-AMC achieves R^2_{log}=0.40, indicating a strong power law relationship for human-rated math problems, while E2H-GSM8K's R^2_{log}=0.12 reflects a much weaker and noisier trend. This tells us that human difficulty in math problems is robustly encoded amongst a suite of models as compared to LLM-sourced difficulty labels for math questions..

The Optimal Probing Position Varies by Task Domain The best performing probes train on activations from middle layers at the final token position, consistent with prior probing studies. However, Figure 3 reveals that while the final token generally yields optimal probes across datasets, certain positions (-4 and -2) show strong performance for specific models, indicating that the final token is not universally superior.

This pattern is most evident when comparing Codeforces with the mathematical datasets (AMC and GSM8K). For Codeforces, 50% of the best probes are trained at the last token, with other probe positions within the last five tokens across all models. In contrast, the mathematical datasets range from -16 to -1 despite having 4 fewer models to train probes with. This strongly suggests that coding difficulty is more strongly encoded near the prompt's end than in other task domains.

Linear probes can steer models towards longer generations and induce tool reasoning Given the strong performance of our linear probes on the E2H-AMC subset, we assume a linear "easy-hard" axis exists within LLM representations in math settings. Following (Li et al., 2024), we use fitted regression coefficients from our trained probes to steer generation during inference. We explore steering with difficulty vectors on Qwen2.5-Math-

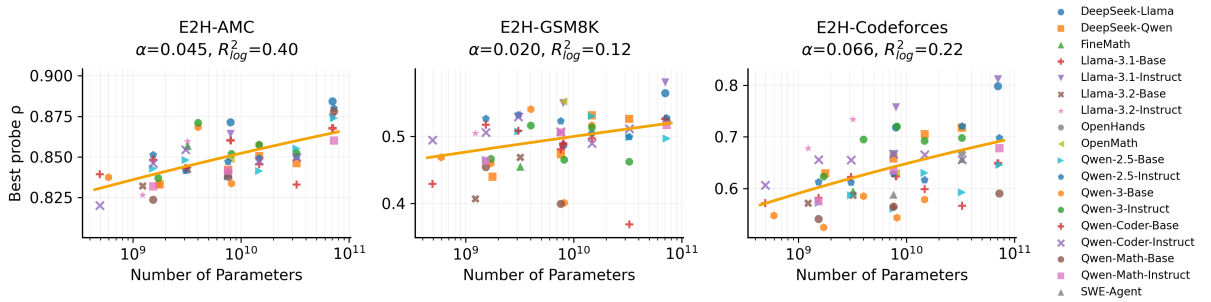

Figure 2: Probe performance scales with model size following a power law. The scatterplot shows the best performing probes for various models on each of the three chosen datasets. Each point represents the best-performing linear probe for a given model. We fits a power law in log space on the gap to perfect probe performance (1 - performance) as a function of model size. From this, we obsevrve a scaling trend that persists across all datasets, but is strongest in E2H subsets with human labels (AMC and Codeforces), than GSM8K with LLM labels.

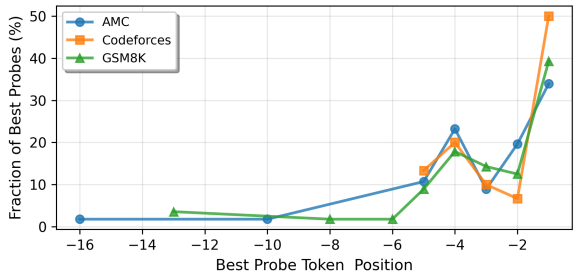

Figure 3: Optimal probe positions vary by domain. Percentage of best-performing probes at each token position (negative indices from end of prompt). Codeforces shows strong preference for the final token (50% of probes), while mathematical datasets (AMC, GSM8K) distribute more evenly across recent tokens, suggesting that difficulty signals are encoded differently across domains.

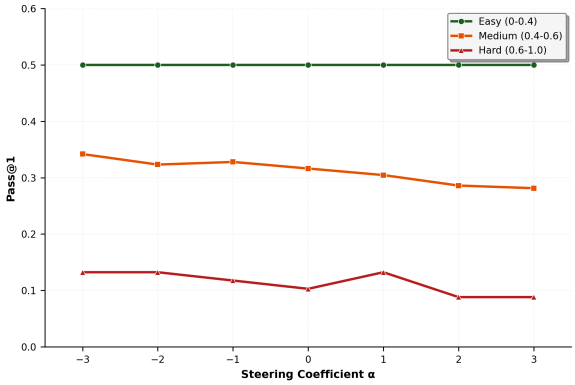

Figure 4: Steering towards "easy" improves mathematical reasoning performance. Pass@1 accuracy on MATH500 for Qwen2.5-Math-1.5B across steering coefficients $\alpha \in [-3, 3]$. Negative coefficients (easier direction) improve performance across all difficulty bins, while positive coefficients (harder direction) degrade performance, particularly on medium and hard problems.

1.5B's Pass@1 performance on the MATH500 benchmark (Hendrycks et al., 2021), using the steering coefficient grid defined in Section 3.1.

In Figure 4, we visualize Qwen2.5-Math-1.5B's performance on MATH500 for each steering coefficient α. Steering towards the "easy" (negative) direction improves performance across all predicted difficulty bins, particularly at $\alpha = -3$. Surprisingly, steering towards the "hard" direction reduces performance on medium and hard questions, with performance gains on hard questions only at $\alpha = 1$.

As shown in Figure 1, at $\alpha = 0$ or $\alpha = +3$, the model hallucinates, fails to generate an answer, and reaches maximum generation length. In contrast, at $\alpha = -3$, the model generates code before and after producing the correct answer. This pattern holds across most questions: steering with $\alpha = -3$ reduces generated text length and increases code generation during inference. We attach more of these results in the Appendix 10 and 9.

Does specialised post-training improve probe performance? Specialised fine-tuning beyond instruction tuning shows mixed results. Figure 5 compares Qwen-2.5-Instruct models (which have undergone instruction tuning) against specialised variants that receive additional task-specific fine-tuning. While some 7B models benefit modestly from the additional specialisation (OpenHands-7B: +0.04, DeepSeek-Qwen-7B: +0.03, Qwen-Coder-Instruct-7B: +0.04), most 32B models show slight degradation (OpenHands-32B: -0.05, SWE-Agent-32B and Qwen-Coder-Instruct-32B: -0.06 each). This suggests that instruction tuning already enables models to form robust difficulty representations, particularly in larger models. Additional task-specific fine-tuning may help smaller models refine these representations, but risks disrupting the general difficulty understanding acquired during instruction tuning in larger models.

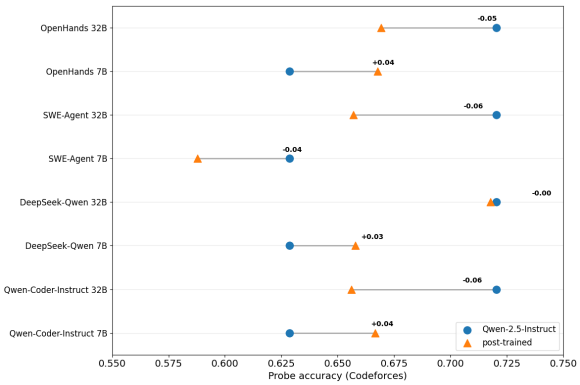

Figure 5: Specialised fine-tuning shows diminishing returns for larger models. Probe accuracy on Codeforces for Qwen-2.5-Instruct and its specialised variants. Values show accuracy change after task-specific fine-tuning: 7B models improve modestly, while most 32B models degrade slightly.

3.2 Tracking Difficulty Representations during GRPO

If our probes have identified a linear "easy-hard" direction, the next natural question is what happens to this direction during post-training. To operationalise this, we investigate how GRPO for math reasoning affects the linear representations of difficulty and whether this evolution depends on the source of difficulty labels (human vs. LLM annotations).

Experimental Design. We train probes on model checkpoints throughout GRPO training using both E2H-AMC (human-labelled) and E2H-GSM8K (LLM-labelled) difficulty scores. This allows us to track how difficulty representations evolve and whether human vs. LLM difficulty labels show different adaptation patterns. Due to computational constraints, we focus on Qwen2.5-Math-1.5B and Llama-3.2-3B-Instruct.

GRPO Configuration Unless otherwise noted, we follow the Dr. GRPO recipe (Liu et al., 2025).We use the MATH training split with 12,000 examples and filtered it to problems with an annotated difficulty ≥ 3, yielding 8,890 questions. Our changes are that training utilises the Verl library on a single A100-80GB GPU with a batch size of 256 and 64 gradient accumulation steps. We increase the learning rate from 1×10^{-6} to 1×10^{-5} based on preliminary experiments, with all other hyperparameters remaining the same as in Dr GRPO. We save a checkpoint of the model at each step.

Probe Training. At each GRPO checkpoint, we extract activations and train linear probes using the same configuration described in Section 3.1. Each checkpoint is treated as an independent model, allowing us to track changes in probes at each training step.

Table 2: GRPO on MATH500: baseline (step 0) and peak Pass@1. Rows under *Dr.GRPO (reported)* are taken from the paper and don't report when the peak step is reached, but give us the total number of training steps when following the training recipe.

Model	Baseline	Peak Pass@1	Peak Step
Dr.GRPO (Ours)			
Qwen2.5–Math–1.5B	64.7	**76.2**	43/67
Dr.GRPO (Liu et al., 2025)			
Oat–Zero–1.5B	61.8	74.2	NA/11,200

Evaluation. We track two complementary metrics throughout GRPO training. Model performance is measured with Pass@1 accuracy on MATH500 (Hendrycks et al., 2021). The answer correctness is verified using MATH-VERIFY (Kydlíček, 2025), and for the evaluation settings, we use greedy decoding with a max token budget of 3000. Probe quality is as in subsection 3.1. This allows us to observe how difficulty representation evolves alongside mathematical reasoning capability.

3.2.1 Results

GRPO results Table 2 shows that our Dr.GRPO implementation achieves a higher peak Pass@1 on MATH500 than the reported Dr. GRPO baseline (Oat-Zero-1.5B: 74.2), using fewer total training steps and with a single GPU. This validates our experimental setup and allows us to study how our probes change during training at each step. Full training curves are provided in Appendix 8.

Human and LLM difficulty representations evolve oppositely during training To understand how difficulty representations change during GRPO, we track probe performance across training checkpoints. Figure 6 reveals contrasting patterns: human-aligned difficulty probes (E2H-AMC, top row) remain stable or improve across training steps, particularly in middle layers. In contrast, LLM-derived difficulty probes (E2H-GSM8K, bottom row) show pervasive degradation in early and middle layers, with performance declining by up to 50%. This layer-wide degradation suggests that LLM-derived difficulty is a noisy signal that GRPO systematically overwrites.

To test whether these representational changes relate to task performance, we examine the correlation between probe strength and Pass@1 accuracy after controlling for overall training progress. Fig-

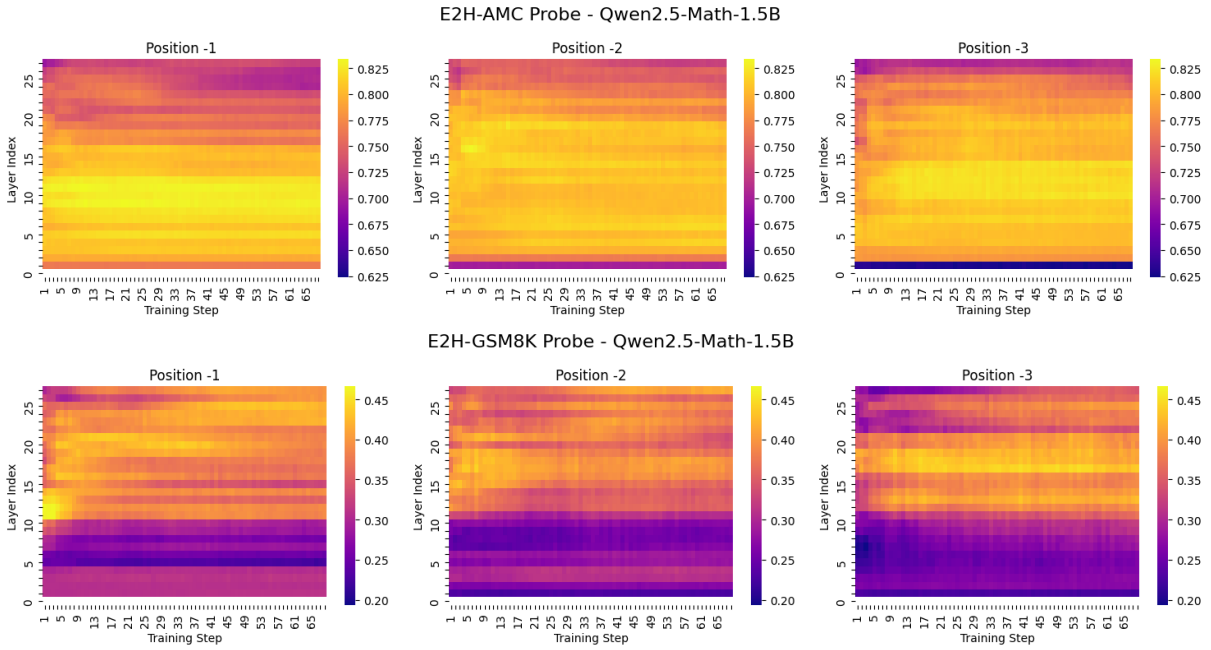

Figure 6: LLM-derived difficulty degrades across most layers while human-aligned difficulty remains mostly stable. Qwen2.5-Math-1.5B linear Probe performance across layers (y-axis) and training steps (x-axis) for the last three token positions. GSM8K probes (rows 2,4) show pervasive degradation in early and middle layers (up to 50% decline). AMC probes (Top row) maintain stable or improving performance across all layers and positions.

ure 7 plots residuals from regressing both variables on training step, revealing the probe-performance relationship independent of general improvement. For human-labeled difficulty (AMC), stronger probe performance strongly predicts higher test accuracy (β=+6.66, p<0.001). For LLM-derived difficulty (GSM8K), the relationship inverts: stronger probe performance predicts lower test accuracy (β=-0.63, p=0.022). This confirms that human difficulty representations strengthen alongside capabilities, while LLM-derived representations degrade as models improve—consistent with the layer-wise degradation in Figure 6.

4 Limitations

This work has several key limitations. First, we focus exclusively on coding and math tasks from E2H, omitting three other subsets. We selected these domains to compare difficulty representations from LLM versus human judgments, with Codeforces as an out-of-domain baseline. Computational constraints prevented extensive experiments across all E2H subsets and GRPO training on larger models. Similarly, memory constraints from training probes and saving checkpoints prevented tracking multiple GRPO seeds.

Our steering experiments focus on Qwen2.5-Math-1.5B, which motivated our RL experiments. However, we recognise that the observed steering effects may be model-specific and broader cross-model studies are central to characterising the full effect of steering along the difficulty direction.

Future work should investigate whether these probes generalise to other agentic tasks requiring coding and reasoning beyond Codeforces and explore how difficulty representations evolve during inference and multi-turn conversations.

5 Conclusion

We demonstrate that difficulty exists as a linear direction in LLM activations that aligns more strongly with human judgment than LLM-derived estimates. Across 60 models, human-labelled difficulty shows robust encoding and clear scaling laws, while LLM-derived difficulty exhibits weak encoding and poor scaling. This reveals a critical asymmetry: models maintain strong implicit representations of human-perceived difficulty but struggle to articulate it explicitly, suggesting internal representations capture signals inaccessible through generation.

Steering experiments on Qwen2.5-Math-1.5B show functional consequences: steering toward "easier" (α = -3, -2, -1) improves MATH500 accuracy while reducing length and hallucination; steering toward "harder" degrades accuracy and in-

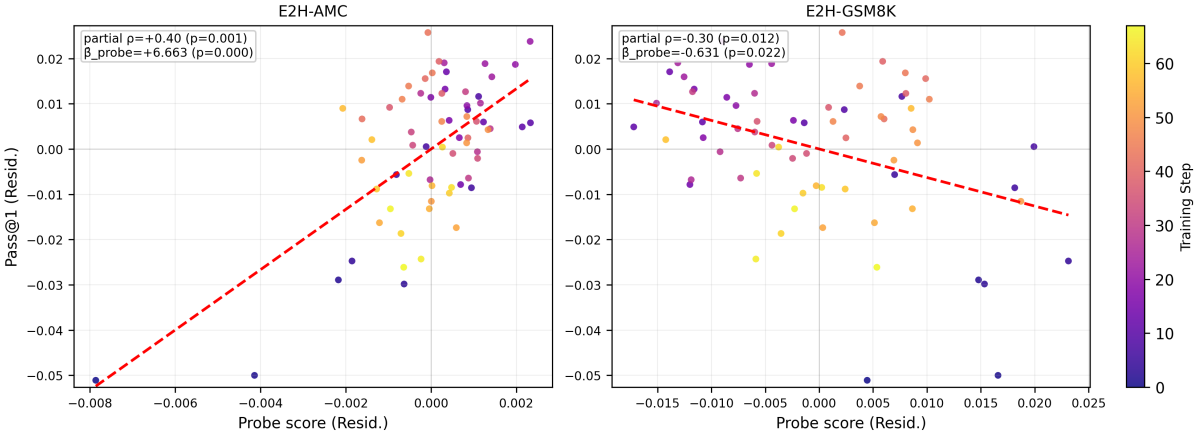

Figure 7: Human-difficulty encoding correlates positively with performance; LLM-difficulty encoding correlates negatively. Each point represents a training checkpoint (color indicates step) for Qwen2.5-Math-1.5B. Axes show residuals after removing the linear effect of training step, isolating the relationship between probe performance (x-axis) and test Pass@1 (y-axis) independent of overall training progress. Human-labeled difficulty (AMC, left) shows strong positive correlation (β=+6.66, p<0.001), while LLM-derived difficulty (GSM8K, right) shows negative correlation (β=-0.63, p=0.022).

creases both. Easier steering also increases code generation, mirroring patterns from GRPO training, suggesting shared mechanisms.

During GRPO training, we also observed that human-aligned difficulty probes strengthened with task performance, while LLM-aligned difficulty probes declined. Combined with steering results, this suggests GRPO refines difficulty representations in ways that align with effective reasoning strategies such as tool use. Collectively, these gains indicate that refined difficulty representation may be functionally important for math reasoning and potentially other tasks.

Future work should investigate why easier steering enhances reasoning, whether these patterns generalise to other domains, and whether adaptive steering during inference time could boost performance.

References

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open LLM Leaderboard (2023-2024).

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint*. ArXiv:2501.12948 [cs].

Mucong Ding, Chenghao Deng, Jocelyn Choo, Zichu Wu, Aakriti Agrawal, Avi Schwarzschild, Tianyi Zhou, Tom Goldstein, John Langford, Anima Anandkumar, and Furong Huang. 2024. Easy2Hard-Bench: Standardized Difficulty Labels for Profiling LLM Performance and Generalization. *arXiv preprint*. ArXiv:2409.18433 [cs].

Thomas Foster, Anya Sims, Johannes Forkel, Mattie Fellows, and Jakob Foerster. 2025. Learning to Reason at the Frontier of Learnability. *arXiv preprint*. ArXiv:2502.12272 [cs].

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On Designing Effective RL Reward at Training Time for LLM Reasoning. *arXiv preprint*. ArXiv:2410.15115 [cs].

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language Models (Mostly) Know What They Know. *arXiv preprint*. ArXiv:2207.05221 [cs].

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint*. ArXiv:2001.08361 [cs].

Junsol Kim, James Evans, and Aaron Schein. 2025. Linear Representations of Political Perspective Emerge in Large Language Models. *arXiv preprint*. ArXiv:2503.02080 [cs].

Hynek Kydlíček. 2025. Math-Verify: Math Verification Library. Original-date: 2025-01-17T11:01:36Z.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. *arXiv preprint*. ArXiv:2306.03341 [cs].

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding R1-Zero-Like Training: A Critical Perspective. *arXiv preprint*. ArXiv:2503.20783 [cs].

AI @ Meta Llama Team. 2024. The Llama 3 Herd of Models. _eprint: 2407.21783.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

OpenAI. Learning to reason with LLMs.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The Linear Representation Hypothesis and the Geometry of Large Language Models. *arXiv preprint*. ArXiv:2311.03658 [cs].

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 preprint*. ArXiv:2311.12022 [cs].

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 77 others. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *arXiv preprint*. ArXiv:2501.12599 [cs].

Boshi Wang and Huan Sun. 2025. Is the Reversal Curse a Binding Problem? Uncovering Limitations of Transformers from a Basic Generalization Failure. *arXiv preprint*. ArXiv:2504.01928 [cs].

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. 2025. Reinforcement Learning for Reasoning in Large Language Models with One Training Example. *arXiv preprint*. ArXiv:2504.20571 [cs].

Sean Williams and James Huckle. 2024. Easy Problems That LLMs Get Wrong. *arXiv preprint*. ArXiv:2405.19616 [cs].

Zhe Yang, Yichang Zhang, Tianyu Liu, Jian Yang, Junyang Lin, Chang Zhou, and Zhifang Sui. 2024. Can Large Language Models Always Solve Easy Problems if They Can Solve Harder Ones? *arXiv preprint*. ArXiv:2406.12809 [cs].

A Appendix

A.1 Probe Results for all Models

E2H-AMC

Please take your time to thoroughly analyze and solve the following math competition problem step by step. Your approach should be detailed, ensuring that each step of your reasoning is clearly explained to minimize errors and maximize understanding.
[PROBLEM_START] {problem} [PROBLEM_END]
While solving, consider all possible scenarios and subtleties involved in the problem. Each step should build upon the previous one logically, leading to a cohesive solution.
Once you arrive at the solution, please present the final answer enclosed in ‘ ’ ‘.
Ensure the answer is displayed using appropriate LaTeX formatting to maintain mathematical precision and clarity.

E2H-Codeforces

Please generate executable Python 3.10 code that directly solves the problem described below. The code should be ready to run without any modifications or additional comments. It must strictly follow Python 3.10 syntax and be formatted correctly for direct execution. Do not include explanations or comments within the code.
[PROBLEM_MAIN_START] {problem_main} [PROBLEM_MAIN_END]
[PROBLEM_NOTE_START] {problem_note} [PROBLEM_NOTE_END]
[INPUT_SPEC_START] {input_spec} [INPUT_SPEC_END]
[OUTPUT_SPEC_START] {output_spec} [OUTPUT_SPEC_END]
[SAMPLE_INPUTS_START] {sample_inputs} [SAMPLE_INPUTS_END]
[SAMPLE_OUTPUTS_START] {sample_outputs} [SAMPLE_OUTPUTS_END]

1. Include any required imports at the top of the script.
2. Use `sys.stdin.readline()` instead of `input()`.
3. Begin with `"""python` and end with `"""`. The content between must be valid Python 3.10 and directly savable as a `.py` file.

Table 3: Model families with sizes and variant counts

Model Family	Sizes (B)	Variants
Base Models		
Qwen-2.5-Base	1.5, 3, 7, 14, 32, 72	6
Qwen-3-Base	0.6, 1.7, 4, 8, 14, 32	6
Qwen-Coder-Base	0.5, 1.5, 3, 7, 14, 32	6
Qwen-Math-Base	1.5, 7, 72	3
Llama-3.2-Base	1, 3	2
Llama-3.1-Base	8, 70	2
Instruct Models		
Qwen-2.5-Instruct	1.5, 3, 7, 14, 32, 72	6
Qwen-Coder-Instruct	0.5, 1.5, 3, 7, 14, 32	6
Qwen-3-Instruct	1.7, 4, 8, 14, 32	5
Qwen-Math-Instruct	1.5, 7, 72	3
Llama-3.1-Instruct	8, 70	2
Llama-3.2-Instruct	1, 3	2
Specialized Models		
DeepSeek-Qwen	1.5, 7, 14, 32	4
DeepSeek-Llama	8, 70	2
OpenHands	7, 32	2
SWE-Agent	7, 32	2
FineMath	3	1
OpenMath	8	1

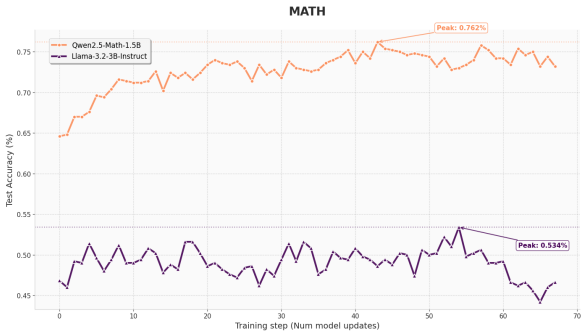

Figure 8: GRPO Training Curves for Qwen and Llama

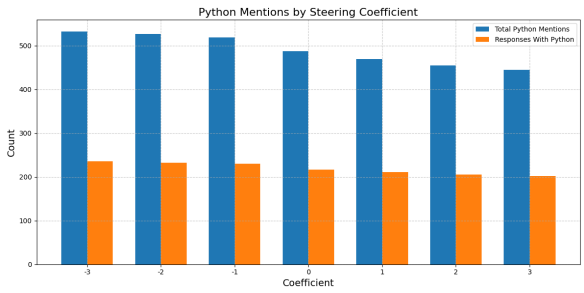

Figure 9: Steering towards the easy direction increases the amount of Python code generated during inference

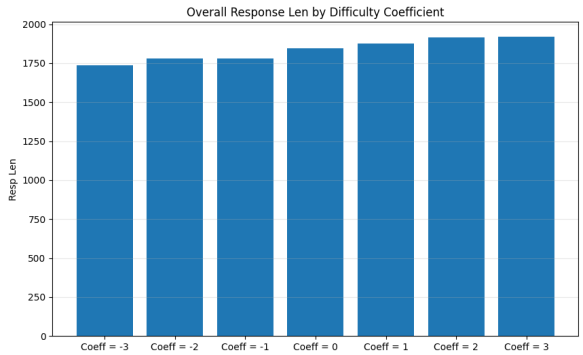

Figure 10: Steering towards the hard direction increases the response length.

Table 4: Linear probe results for all models (higher is better). Best per dataset in **bold**.

Model	E2H-AMC	E2H-GSM8K	E2H-Codeforces
Llama-3.1-70B-Instruct	0.874	**0.580**	**0.811**
DeepSeek-R1-Distill-Llama-70B	**0.884**	0.564	0.798
Llama-3.1-8B-Instruct	0.864	0.549	0.757
Llama-3.2-3B-Instruct	0.860	0.533	0.734
Qwen2.5-72B-Instruct	0.880	0.527	0.698
DeepSeek-R1-Distill-Qwen-32B	0.846	0.526	0.718
DeepSeek-R1-Distill-Qwen-14B	0.851	0.531	0.705
Qwen3-4B	0.871	0.516	0.695
DeepSeek-R1-Distill-Llama-8B	0.871	0.487	0.718
Qwen2.5-32B-Instruct	0.850	0.499	0.720
Qwen3-14B	0.857	0.513	0.692
Qwen2.5-Math-72B-Instruct	0.860	0.517	0.678
Llama-3.1-70B	0.868	0.526	0.649
Qwen2.5-Coder-3B-Instruct	0.854	0.529	0.655
Qwen3-8B	0.852	0.465	0.720
OpenMath2-Llama3.1-8B	0.849	0.552	0.630
Qwen2.5-Coder-32B-Instruct	0.850	0.511	0.656
Qwen2.5-72B	0.874	0.497	0.646
Qwen2.5-Coder-7B-Instruct	0.839	0.507	0.667
Qwen3-32B	0.852	0.462	0.698
Llama-3.2-1B-Instruct	0.826	0.504	0.677
Qwen2.5-Coder-1.5B-Instruct	0.846	0.506	0.656
openhands-lm-32b-v0.1	–	–	0.669
openhands-lm-7b-v0.1	–	–	0.668
Qwen2.5-Coder-14B-Instruct	0.847	0.489	0.665
Qwen2.5-14B	0.841	0.530	0.630
Qwen2.5-7B-Instruct	0.847	0.522	0.629
Qwen3-4B-Base	0.868	0.540	0.585
Qwen2.5-Math-72B	0.878	0.522	0.590
Qwen2.5-1.5B-Instruct	0.851	0.526	0.612
Qwen2.5-3B-Instruct	0.841	0.532	0.612
Qwen2.5-Math-7B-Instruct	0.842	0.506	0.635
Qwen2.5-Coder-3B	0.843	0.508	0.622
DeepSeek-R1-Distill-Qwen-7B	0.842	0.473	0.658
Llama-3.1-8B	0.860	0.489	0.624
SWE-agent-LM-32B	–	–	0.657
Qwen2.5-14B-Instruct	0.849	0.504	0.616
Qwen3-14B-Base	0.857	0.516	0.578
Qwen2.5-Coder-1.5B	0.848	0.517	0.582
Qwen2.5-32B	0.855	0.499	0.592
Qwen2.5-3B	0.848	0.507	0.587
Qwen2.5-Coder-14B	0.845	0.497	0.599
Qwen3-1.7B	0.837	0.466	0.623
Qwen2.5-Coder-0.5B-Instruct	0.820	0.494	0.606
FineMath-Llama-3B	0.857	0.455	0.594
Qwen2.5-7B	0.838	0.506	0.561
DeepSeek-R1-Distill-Qwen-1.5B	0.833	0.440	0.629
Llama-3.2-3B	0.842	0.469	0.587
Qwen2.5-Coder-7B	0.840	0.480	0.565
Qwen2.5-1.5B	0.843	0.465	0.572
Qwen2.5-Math-1.5B-Instruct	0.832	0.464	0.575
Qwen3-0.6B-Base	0.837	0.469	0.547
Qwen2.5-Coder-0.5B	0.839	0.429	0.571
Qwen2.5-Math-1.5B	0.824	0.454	0.541
Qwen3-1.7B-Base	0.833	0.460	0.524
Llama-3.2-1B	0.832	0.407	0.571
Qwen2.5-Math-7B	0.838	0.399	0.564
Qwen3-8B-Base	0.834	0.401	0.543
Qwen2.5-Coder-32B	0.833	0.369	0.566
SWE-agent-LM-7B	–	–	0.588

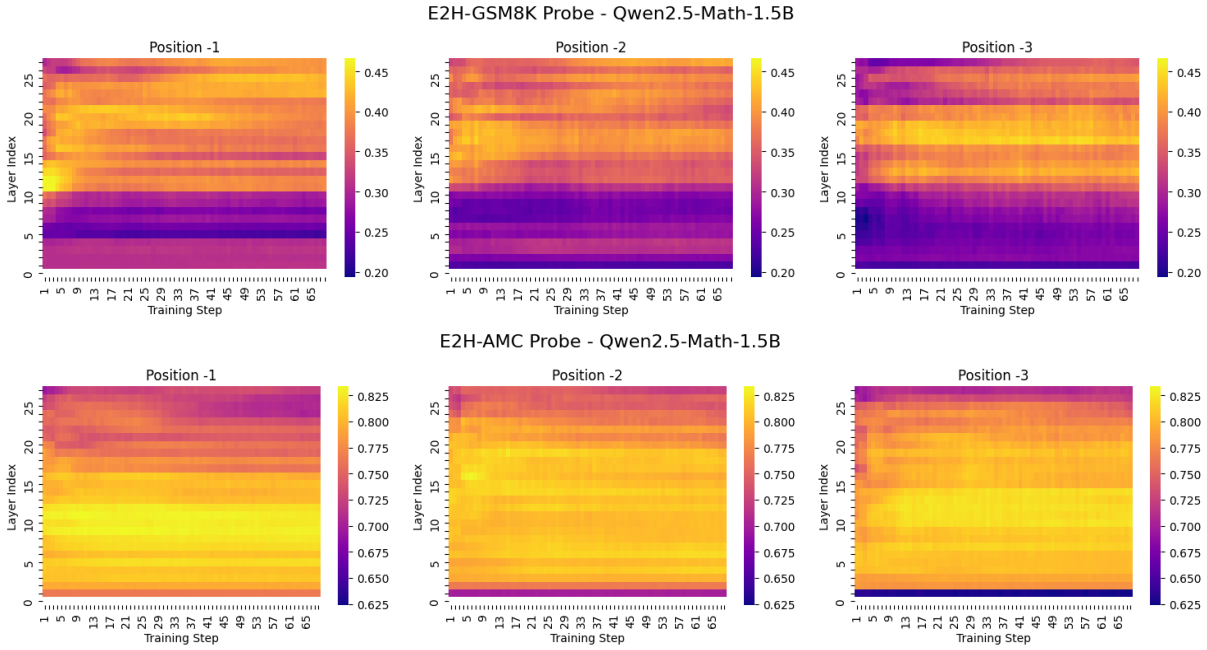

Figure 11: Llama Comparison