MAT1510 Final Report

Khash Azad Giovanni Galea Curmi

December 7, 2025

1 Introduction

Deep neural networks learn complex representations of data by sequentially transforming input features. The paper "Deep Learning and the Information Bottleneck Principle" demonstrates that each layer progressively compresses the input information while preserving features relevant to the training task [1]. Building on this perspective, we can view network layer regions as serving distinct functional roles: early layers perform complex transformations that yield compressed, simpler representations of the data, while later layers refine these representations to extract abstractions relevant to the task.

Understanding the effect of model depth is particularly relevant for large language models (LLMs), where recent efforts have focused on incentivizing reasoning capabilities. The most prominent approaches involve Reinforcement Learning with Verifiable Rewards, which involves optimizing a pretrained model using automatically generated reward signals. Although these techniques have shown impressive improvements, it has not been proven that RLVR enables models to acquire new reasoning abilities. In fact, a recent paper has further investigated this question and claims that the reasoning paths generated by the current RLVR fine-tuned models already exist in their corresponding base models, and that the reinforcement learning process merely makes the model more sample-efficient [2].

2 Related Work

Our work is mainly based on two papers: "Reasoning with Sampling: Your Base Model is Smarter Than You Think" [3] and "Do Language Models Use Their Depth Efficiently?" [4]. The former proposes a training-free technique to elicit reasoning capabilities from LLMs, while preserving the model's output diversity. Their findings suggest that reasoning paths in RL-trained models are already present within pretrained models. The latter analyzes whether deep models enable different computations than shallow models. The key findings suggest that the layers in the second half contribute significantly less to the residual stream, skipping those layers has minimal effect on downstream computations, and deeper models map linearly to shallower ones.

2.1 Reasoning with Sampling: Your Base Model is Smarter Than You Think

The motivation for this paper stems from the observation that RL post-training does not add new reasoning capabilities to the base model, but rather makes it more sample-efficient. The authors claim that the post-trained distribution of the model is simply a sharper version of the base model distribution, rather than adding reasoning traces that the base model is unlikely to generate. The authors propose a sampling algorithm that leverages additional compute at inference time, achieving single-shot performance that nearly matches RL fine-tuned models. Their method does not require training and maintains the base model's output diversity.

The intuition behind their method is based on distribution sharpening, which refers to modifying a distribution so that higher-likelihood regions are upweighted while lower-likelihood regions are dampened. The distribution-sharpening strategy they use is sampling from the power distribution. The paper emphasizes the difference between sampling from the power distribution and low-temperature sampling: "The power distribution upweights tokens with few but high likelihood future paths, while low-temperature sampling upweights tokens with several but low likelihood completions" [3]. Low-temperature sampling considers tokens only based on local likelihood, without accounting for how each selection affects the likelihood of the full reasoning path. On the other hand, sampling from the power distribution accounts for future completions while computing the weights for next-token prediction.

2.1.1 Algorithm: Metropolis-Hastings

Although we have access to the values of p^α over any sequence, they are unnormalized, and computing the partition function to normalize them is computationally intractable. They use the Metropolis-Hastings algorithm, which exactly solves this problem by allowing approximate sampling from an unnormalized probability distribution. The algorithm forms a Markov chain by proposing candidate sequences and accepting them with a probability that depends only on the unnormalized relative weights and a tractable sampler. The only conditions for the algorithm to converge to the target sequence are that the sampler can reach any high probability region (irreducible) and does not cycle deterministically (aperiodic). Their mechanism works by randomly selecting a position in the current sequence and resampling the remaining positions using the same LLM at a lower temperature (the sampler). The workaround they propose to avoid the high computational cost of applying Metropolis-Hastings to long sequences is a block-wise algorithm that samples from distributions of increasing length, using each converged sample to initialize the next.

Algorithm 1 Power Sampling for LLM

```
1: for each block in the sequence do
2:     1) Extend the prefix to initialize the block
       by autoregressively generating from the
       proposal model
3:     2) MH Loop: Randomly choose a position in the sequence, resample the tokens
       following the selected position, and accept
       or reject based on likelihood ratios
4:     3) Set the current block as the prefix for
       the next stage
5: end for
6: return the complete sequence
```

2.1.2 Results

The paper's results show that power sampling matches and in some cases outperforms GRPO post-training across multiple models and benchmarks. Their method samples from high-likelihood regions of the base model distribution while maintaining noticeable spread, whereas GRPO samples are concentrated around the highest likelihood peak. Most importantly, power sampling achieves GRPO-level single-shot performance without compromising the model's output diversity, resulting in much better multi-shot performance compared to GRPO.

2.2 Do Language Models Use Their Depth Efficiently?

This paper aims to investigate whether deep models use their depth efficiently. More specifically, the experiments test whether deeper models compose more features to perform computations that are impossible in shallow models. Their experiments consist mainly of four parts: 1) Analyzing the norm of the residual stream and comparing it after each layer's contribution 2) Measuring the effect of skipping layers on future layers computations 3) Analyzing multihop challenges to look for evidence of deeper computations in more complex examples 4) Training linear maps between layers of shallower and deeper models to measure the correspondence between them.

2.2.1 Layers' Interaction With The Residual Stream

To analyze the contribution of each layer to the residual stream, the authors measure the l_2 norm of the residual stream and compare it to the attention and MLP sublayer outputs. Additionally, they measure the cosine similarity between sublayer outputs and the residual stream, where values close to zero indicate the addition of new features to the residual stream, negative values indicate erasing features, and positive values indicate strengthening existing ones.

Their results reveal that the residual norm grows across layers. However, later layers contribute proportionally less. The relative contribution is consistent in the first half of the network but drops noticeably around the middle point, and recovers slightly in the final layers. The cosine similarity analysis shows that early layers integrate new context from neighboring tokens and erase features in the first half of the network. A noticeable phase transition happens around the middle point, where layers shift from erasing features to strengthening existing ones. This transition aligns with the drop in the relative contribution of layers in the second half, suggesting a change in the network's behavior.

2.2.2 Layers' Influence On Downstream Computations

To analyze interactions between layers, the authors use interventions to measure the effect that skipping a layer has on downstream computations. They first run a prompt and log the residual at each layer. Then, they run the same prompt again, but this time by skipping a specific layer and measuring the relative change in each subsequent layer's contribution. Their results show that layers in the second half of the network have minimal influence on later computations, unlike early layers, which focus on integrating information and building on each other's outputs. However, they state that computations in the later layers are important for the model's output.

To further validate that the later layers are refining the output distribution rather than computing reusable features, the authors measure the KL divergence between intermediate and final predictions. The results show that distribution refinement begins at the same phase transition point where cosine similarities shift. This further supports their initial hypothesis that the second half of the network performs smaller updates to refine the prediction distribution using information already present in the residual.

2.2.3 Lack Of Evidence For Compositional Computations

The authors state that if larger models perform compositional computations, they must break them into subproblems, with later layers combining solutions from earlier ones. They use two approaches to test whether this is in fact happening in deeper models. The first focuses on performing residual erasure interventions, replacing the residual at a given position with an average, and measuring the effect on predictions. The purpose is to determine until which layer information from a given token remains useful. The second method involves computing a depth score that measures the influence of layers on later layers and the output on future positions, with higher scores indicating the model uses later layers more. The authors evaluate these methods on a series of challenging multi-hop and math questions. However, their results show no evidence that more complex problems use deeper computations.

2.2.4 Layer Correspondence Between Shallow and Deep Models

The authors trained linear probes to map each layer of a shallow model to each layer of a deeper model and measured the relative prediction error for each layer pair. Their goal was to verify whether deeper models perform computations not present in shallower ones, or if they simply spread the same computation across more layers. The results show strong correspondence between matched layers across the two models. This strongly suggests that deeper models are likely stretched-out versions of shallower ones, spreading the same computation across additional layers rather than performing new computations.

3 Hypothesis

We investigate our hypothesis:

Can inference-time power sampling compensate for reduced model depth?

We postulate that the reasoning traces in LLMs are formed in the model's earlier layers, while the later layers primarily serve to refine confidence. Therefore, by removing these refinement layers (truncation) and applying the power sampling proposed in "Reasoning with Sampling", we aim to test whether the model's core reasoning capabilities remain intact and whether power sampling can substitute for some of the computation normally performed by late layers.

4 Experiment

4.1 Methodology

To empirically test our hypothesis, we designed a three-stage experiment. First, we leveraged the LogitLens analysis used in "Do Language Models Use Their Depth Efficiently?" [4] to quantifiably map the information flow within the Qwen2.5-Math models. By projecting hidden states from intermediate layers into the vocabulary space and measuring the divergence between those predictions and the final ones, we identified the specific layers where the model's internal representation begins to converge toward the final ones. Second, based on these saturation points, we implemented a "Stitch-and-Preserve" truncation method. Unlike standard layer pruning which simply discards the end of the network, our approach retains layers $0 \dots k$ and attaches the original final layer (Layer L) and RMSNorm. This ensures that the intermediate features are projected using the fully trained output distribution, preserving the semantic alignment of the hidden states. Finally, we integrated the Metropolis-Hastings Power Sampling algorithm introduced by "Reasoning with Sampling" [3] on these truncated models. This allowed us to determine if additional compute spent searching the probability landscape at inference time can reveal the true reasoning capabilities of the model developed in its earlier layers.

4.1.1 LogitLens KL Divergence Test

To determine the optimal truncation depth, we employed a LogitLens analysis, applying the final unembedding head to the hidden states of each intermediate layer to project them into the vocabulary space. We then compared how the output distribution of each intermediate layer differs from the final layer's output using Kullback-Leibler (KL) divergence. This comparison allowed us to measure information saturation at each depth and motivated the decision of which layer should serve as the 'cut' point for the model.

4.1.2 Stitch-and-Preserve Truncation

Unlike standard pruning techniques that remove individual neurons, we perform block-level removal of transformer layers. We implemented a "stitch-and-preserve" strategy: for a target depth k, we retain layers $0 \dots k$ and explicitly attach the original final layer (Layer L) and the final RMSNorm output layer. We initially attempted the same experiments using naive truncation, simply discarding all layers beyond k, but this produced incoherent outputs, likely because the intermediate representations were never trained to be fed directly to the unembedding layer.

4.1.3 Integrated MCMC Sampling

We applied power sampling using the truncated models at a lower temperature, treating them as the proposal distribution for the Metropolis–Hastings sampling process. We generated a spectrum of truncated models ranging from Layer 21 to Layer 26 and evaluated them on a subset of the MATH500 benchmark. All experiments were conducted with $k = 10$ resampling steps and a sharpening coefficient of $\alpha = 2$. We selected $\alpha = 2$ to moderately sharpen the proposal distribution, upweighting high-likelihood reasoning paths without inducing the severe mode collapse often associated with higher values of α. This ensures the sampler focuses on coherent logical chains while retaining sufficient diversity [3]. The choice of $k = 10$ balances computational cost with sufficient mixing time, allowing the Markov chain to converge to the target distribution effectively.

4.2 Results

4.2.1 LogitLens Analysis

The LogitLens analysis (Figure 1) demonstrates why specific layers were targeted. For both *Qwen2.5-Math-1.5B* and *Qwen2.5-Math-7B*, the KL divergence remains high for most of the network depth, indicating that early representations differ significantly from those of the final layer. Around Layers 21–25, we observe a sharp drop in divergence, suggesting that the models' internal representations are converging toward the representations of the final layer. The two curves also differ significantly: the 7B model shows a smoother, more gradual decline, while the 1.5B model drops more abruptly at a slightly earlier layer. This difference suggests that the larger model's greater capacity per layer allows it to make gradual refinements toward the final representation, whereas the smaller model must compress these refinements into fewer layers, resulting in the sharper transition.

LogitLens KL Divergence Across Depth
Qwen2.5-Math-1.5B (top) vs Qwen2.5-Math-7B (bottom)

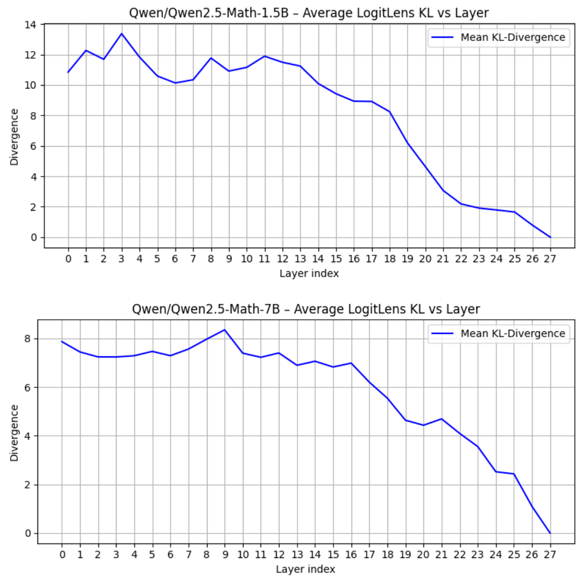


Figure 1: Average LogitLens KL divergence across layers for Qwen2.5-Math-1.5B and Qwen2.5-Math-7B on MATH500.

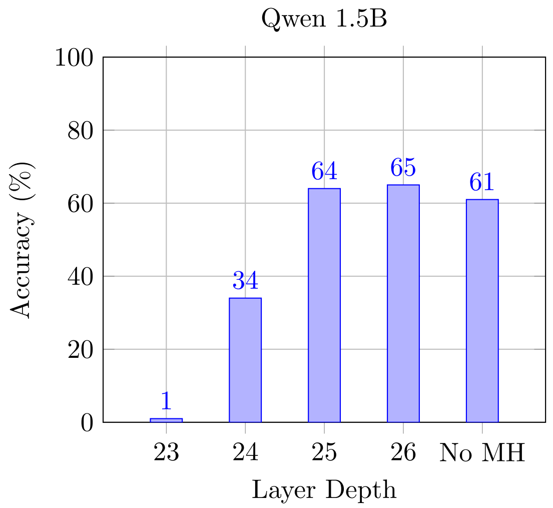
4.2.2 Critical Depth Threshold

Our results reveal a clear phase transition in capability as a function of depth. Truncating below Layer 23 causes catastrophic collapse (0–1% accuracy on MATH500) for both models, regardless of whether power sampling is used. This indicates a hard boundary where early-to-mid layers perform computation essential for reasoning that inference-time search cannot recover.

4.2.3 Search vs. Depth Trade-off

When operating above the critical depth threshold, performance recovers rapidly. At 25–26 layers, inference-time power sampling successfully compensates for the missing layers' computations. The truncated 7B model (Layer 26) with power sampling achieves **79.0%** accuracy, significantly outperforming the full, non-truncated baseline, which achieved 71.0% accuracy. A similar pattern holds for Qwen2.5-Math-1.5B, where truncating at layer 26 with power sampling reaches **65.0%** accuracy compared to 61.0% for the full baseline.

Although both models share the same architecture and number of layers, truncating at an earlier stage yielded better performance for the smaller model than for the larger one. We hypothesize two potential reasons for this observation:



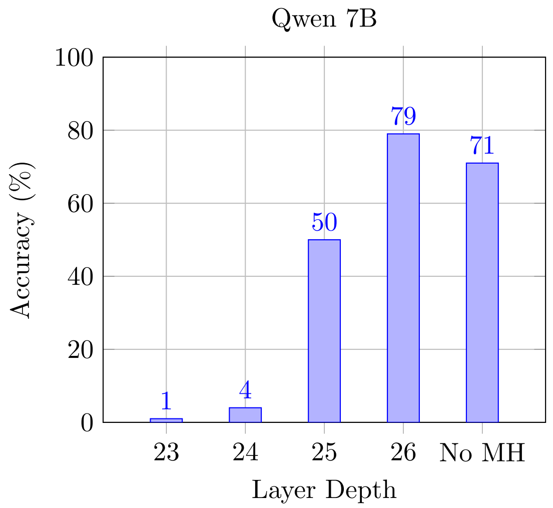


Figure 2: Accuracy % (MATH500) - Qwen 1.5B and Qwen 7B. "No MH" bar represents the full 28-layer baseline model without inference-time power sampling.

Capacity-Induced Computational Spreading: Since both models share the same depth (28 layers), the performance difference is driven by the model width (hidden dimension size). The 7B model's wider hidden states allow it to maintain complex, superposed feature representations for longer portions of the forward pass. This "computational bandwidth" enables the model to spread the reasoning process across more layers, effectively delaying the collapse to a specific token choice. In contrast, the 1.5B model, constrained by a narrower residual stream, is forced to resolve ambiguities and converge to a solution in earlier layers, resulting in higher accuracy at shallower truncation points, though at the cost of reduced benefit from the later layers' refinement capacity.

Delayed Phase Transition: This hypothesis is strongly supported by our LogitLens analysis (Figure 1), which shows that the 7B model maintains a higher KL divergence at Layers 21–24, whereas the 1.5B model's divergence is lower in the same region. The higher KL divergence in the larger model indicates that its intermediate representations remain distinct from the final output distribution for longer, suggesting that the "phase transition", the phase where the network switches from extracting reasoning features to merely refining output confidence, occurs later in the wider model.

4.2.4 Cost-Benefit Analysis

While power sampling is shown to be effective at increasing the accuracy of the truncated model, the efficiency trade-off remains substantial. The power sampling algorithm requires iterative resampling for every block to approximate samples from the target distribution p^α. More specifically, the generated sequence is divided into $\lceil T/B \rceil$ blocks, and at each block k, the algorithm performs N_{MCMC} resampling, each regenerating on average $\frac{kB}{2}$ tokens from a randomly selected position. This results in a total token generation cost of $\frac{N_{\text{MCMC}} \cdot T^2}{4B}$ compared to standard autoregressive decoding [3]. With the authors' suggested parameters that were used for our experiments ($N_{\text{MCMC}} = 10$, $B = 192$, average output length $T \approx 679$), this results in approximately 8.84x more token generations than a single decoding step. Although truncating a 7B parameter model from 28 to 26 layers results in saving roughly 7% in computations, this is far from sufficient to cover for the order-of-magnitude increase in inference cost from power sampling. We conclude that inference time power sampling is a powerful technique for revealing latent reasoning capabilities, but layer truncation alone does not offer enough compute saving to offset the cost of iterative sampling.

References

[1] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

[2] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025.

[3] Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think, 2025.

[4] Róbert Csordás, Christopher D. Manning, and Christopher Potts. Do language models use their depth efficiently?, 2025.