

Overclocking LLM Reasoning: Monitoring and Controlling Thinking Path Lengths in LLMs

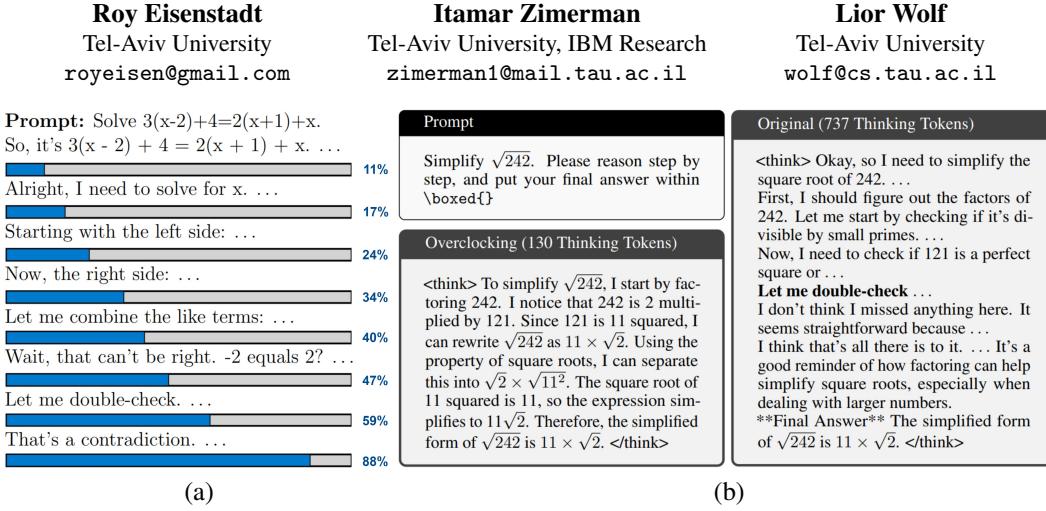


Figure 1: Applications of our method. (a) Monitoring the reasoning progress. (b) Overclocking it.

Abstract

Recently, techniques such as explicit structured reasoning have demonstrated strong test-time scaling behavior by enforcing a separation between the model’s internal “thinking” process and the final response. A key factor influencing answer quality in this setting is the length of the thinking stage. When the reasoning is too short, the model may fail to capture the complexity of the task. Conversely, when it is too long, the model may overthink, leading to unnecessary computation and degraded performance. This paper explores and exploits the underlying mechanisms by which LLMs understand and regulate the length of their reasoning during explicit thought processes. First, we show that LLMs encode their progress through the reasoning process and introduce an interactive progress bar visualization, which is then used to reveal insights on the model’s planning dynamics. Second, we manipulate the internal progress encoding during inference to reduce unnecessary steps and generate a more concise and decisive chain of thoughts. Our empirical results demonstrate that this “overclocking” method mitigates overthinking, improves answer accuracy, and reduces inference latency. Our code is publicly available.



https://github.com/royeisen/reasoning_loading_bar

1 Introduction

In recent years, Large Language Models (LLMs) such as ChatGPT [21] have demonstrated remarkable capabilities across both general-purpose and domain-specific challenges, exhibiting surprising generalization abilities [13]. An emerging frontier for enhancing their performance is test-time scaling, which aims to improve model responses by dynamically allocating additional computational effort during inference [23, 28].

Project page: <https://royeisen.github.io/OverclockingLLMReasoning-paper>

A prominent strategy for enabling effective long-form reasoning is explicit structured reasoning, which separates the model’s reasoning process from its final answer [27]. This is commonly implemented using special tokens that mark the start and end of the reasoning phase (e.g., <think> and </think>, which are employed by DeepSeek-R1 [7]), encouraging the model to deliberate before responding.

The effectiveness of such techniques depends heavily on the length of the thinking stage [10, 30]. Too few steps may fail to capture sufficient task complexity, while too many can lead to overthinking and unnecessary computation [25]. Thus, controlling the thinking path length and understanding its underlying mechanisms are essential for balancing accuracy, efficiency, and responsiveness.

While several prior works have explored the impact of reasoning length on model quality and proposed techniques to control or optimize response length through specialized training procedures [1] or prompting [29], they do not investigate the underlying mechanistic design that governs how reasoning length is determined. Moreover, none of these approaches address the unique case of structured reasoning, where the model explicitly separates a holistic thinking stage from the final response. We consider this setting particularly important, as it offers a valuable opportunity to better understand and exploit the “thinking phase” of reasoning models.

This work tackles both of these gaps. First, we investigate whether LLMs are capable of monitoring their relative position within the thinking process. Possessing even an implicit form of progress monitoring may enable aspects of key cognitive concepts such as self-regulation [32] and metacognition [6]. To identify mechanisms that encode this information, we perform a regression analysis, and show that the relative position can be captured by projections that we term “progress vectors”.

The extracted information is then used to create an interactive loading bar visualization, see Figure 1(a) that depicts the model’s progress throughout the thinking phase, making the reasoning process more transparent and easier for users to collaborate with.

The ability to extract progress information does not mean that the model employs it mechanistically, unless an intervention analysis is performed. We thus manipulate the internal representation along the progress vectors and achieve a clear modulation of the length of the thinking phase, showing overclocking effects. The former is depicted in Figure 1(b). Reassuringly, this modulation does not tend to be detrimental to the LLM’s performance. In fact, we show that overclocking can improve the model’s performance by mitigating overthinking, enhancing computational efficiency, and tailoring the model’s reasoning depth to each task’s complexity.

Our main contributions are as follows: (i) We provide the first empirical evidence that LLMs maintain an internal estimate of their relative position within the explicit thinking phase. This finding sheds light on the plausibility of planning and self-monitoring abilities of LLMs, concepts typically associated with cognitive capabilities. (ii) We identify an internal encoding of this information by learning progress vector projections, and employ these projections to expose a dynamic thinking progress bar. (iii) We perform an intervention study and manipulate the progress vectors to overclock and downclock the reasoning process. Finally, (iv) We empirically demonstrate that interventions of the progress vectors improve both the efficiency and the effectiveness of strong LLMs such as DeepSeek-R1 by mitigating overthinking and reducing the generation of unnecessary reasoning steps.

2 Related Work

Test-time scaling refers to a recent trend in which the reasoning depth of an LLM is dynamically increased during inference, allowing the model to perform complex multi-step reasoning and extend its problem-solving capabilities [23, 28]. This approach heavily relies on the model’s ability to reason effectively over long trajectories. One notable technique for improving the effectiveness of multi-step reasoning is Chain-of-Thought (CoT) prompting [27], which encourages the model to generate a sequence of intermediate reasoning steps before producing its final answer by incorporating additional guidance into the prompt. This technique has been adopted and extended in models such as DeepSeek-R1 [7], OpenAI’s O1 [9], and S1 [16] which are fine-tuned on datasets containing CoT-style reasoning steps prior to the final answer. A further refinement was proposed by DeepSeek-R1 [7], which explicitly enforces open-ended structured reasoning through the use of dedicated <think> and </think> tokens that mark the beginning and end of the model’s “thinking” phase.

The length of the reasoning chain is a key factor influencing the performance of modern reasoning models. If the chain is too long, the model may engage in unnecessary computation and suffer from

overthinking [25, 24, 2], a phenomenon in which excessive reasoning steps degrade answer quality. On the other hand, if the reasoning is too short, the model may fail to capture the complexity required to solve the task effectively [26]. To address this trade-off, state-of-the-art LLMs are increasingly designed to support adaptive thinking, where the length of the reasoning phase is dynamically adjusted based on task complexity. Two main strategies for improving the length-control the duration of a model’s reasoning have been proposed: (i) Inference-time methods modify the prompt or inject guidance while the model is generating its chain of thought. Notable examples include Kojima et al. [13], who optimize the initial prompt, and Wu et al. [29], Jin et al. [11], who dynamically intervene during reasoning by prompting to shorten or lengthen the reasoning trajectory (ii) Training-time methods fine-tune the model with objectives and datasets that explicitly encourage regulation of the reasoning length. A notable example is Length-Controlled Policy Optimization (LCPO) [1], which frames reasoning-length control as a RL objective. Beyond the context of controlling reasoning-chain length, length control has long been studied in standard LLMs to manage overall output length. A substantial body of work addresses this objective across applications such as summarization [14, 22]. In contrast to these methods that primarily aim to control the length of the generated answer for efficiency or user-specific preferences, our work focuses on the thinking length rather than the final output. This allows us to gain insights into the model’s planning, monitoring, and control capabilities, which we then leverage to improve model quality and model responsiveness.

Mechanistic interpretability aims to reverse-engineer LLMs so they can be understood, trusted, and controlled [5, 19]. This is done by connecting observable behaviors to concrete neural components by assigning semantic roles to individual neurons or weights and isolating compact “circuits”. Prominent examples include the induction-heads [20] and mechanisms for arithmetic [12, 17, 31, 3] and in-context learning via task-vectors [8]. Our work follows this line of research and aims to uncover the intrinsic mechanisms that encode a model’s relative position within its internal reasoning process, with the goal of better understanding and even controlling the planning abilities of reasoning LLMs.

3 Method

We investigate monitoring and control of the thinking phase in reasoning models. In Section 3.1, we show that the model’s hidden representations encode information about its position within the thinking phase. This allows for real-time monitoring in the form of a loading bar and reveals distinct thinking patterns. In Section 3.2, we demonstrate that this internal mechanism can be modified to control the depth of the thinking phase.

3.1 Monitoring the Thinking-Phase

We hypothesize that as part of learning to reason effectively, models must also implicitly learn to track their progress through the thinking phase, maintaining an estimate, for example, of how close they are to the final answer. Since progress-tracking is input-dependent, such information cannot be stored in the model’s static weights and must instead be encoded dynamically in the hidden representations passed between layers. We chose to extract information from the final hidden layer for two main reasons. First, this layer provides the richest token-level representations, having integrated information from all previous layers. Second, our goal is to eventually intervene in the reasoning process by manipulating the thinking progress. The final hidden layer is especially suitable for this because it is close to the token embedding space, allowing for effective intervention. Moreover, it lies downstream of the attention layers, which cache intermediate states. This means that modifying the hidden representation at this stage can influence the immediate output token without disrupting the model’s internal memory, thereby isolating the effect of the intervention to a single step.

As another design choice, we focus on models that perform explicit structured reasoning, characterized by a distinct and continuous reasoning phase delimited by `<think>` and `</think>` tokens, as employed in DeepSeek-R1 [7]. This structure enables us to quantify the model’s progression within the thinking phase by precisely labeling each token with an interpolated value between zero and one, according to its relative position.

To learn a thinking progress predictor, we initially sample entries from a dataset of mathematical problems. We then prompt a reasoning model with a single problem each time, generating an answer that contains a thinking phase. For a generation that ended successfully, we specifically observe its thinking trajectory $T = w_1 w_2 \dots w_N$, which is the sequence of tokens between “`<think>`” and

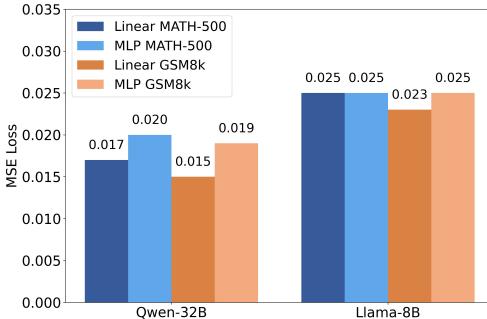


Figure 2: Linear vs Non-linear regression MSE loss for monitoring the thinking phase. For each model-dataset pair, 30 problems were sampled.

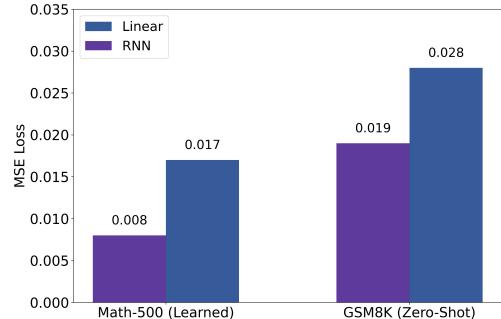


Figure 3: MSE of Linear vs RNN for thinking-phase monitoring regression in learned and zero-shot settings.

“</think>”, and collect the representations from the last layer, $\{\mathbf{h}_j^{(k)} \in \mathbb{R}^d \mid j = 1, \dots, N_k\}$ for each token j in the k trajectory, where d is the hidden size of the reasoning model and N_k is the number of tokens in that trajectory. We pair each hidden state with its normalized position in the sequence as a regression label.

Formally, we construct a dataset D by:

$$\mathcal{D} = \{(\mathbf{h}_j^{(k)}, p_j^{(k)}) \mid k = 1, \dots, K, j = 1, \dots, N_k\}, \quad (1)$$

where $\mathbf{h}_j^{(k)} \in \mathbb{R}^d$ is the hidden representation of the j -th token in the k -th thinking trajectory, and $p_j^{(k)} = \frac{j}{N_k} \in (0, 1]$ is the token’s relative position in its sequence. Here, K denotes the number of sampled trajectories and the total number of samples in \mathcal{D} is $M = \sum_{k=1}^K N_k$.

This allows us to optimize for a progress extraction function f_θ which maps the hidden representations to their relative positions in the form of a regression task $\theta^* = \arg \min_\theta \sum_{(\mathbf{h}, p) \in \mathcal{D}} (f_\theta(\mathbf{h}) - p)^2$.

We conduct experiments using two reasoning models - DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Llama-8B[7] - both of which are designed to explicitly separate the thinking process from final answers. We evaluate them on two benchmark datasets of mathematical problems: Math500[15] and GSM8k [4]. For each dataset, we sample 30 problems and prompt each model to generate 5 distinct responses per problem using a temperature of 0.6. The prompt format includes the instruction: “Please reason step by step, and put your final answer within \boxed{.} <think>”, as recommended by Guo et al. [7]. Following the dataset curation procedure described earlier, we obtain four distinct progress regression datasets, each corresponding to a unique combination of reasoning model and problem set. We split each dataset into train and test sets using an approximate 80/20 ratio based on entire thinking trajectories, ensuring that all generations for a given problem remain within the same split.

We first fit a linear regressor parametrized by $\theta \in \mathbb{R}^d$ as the function f_θ for estimating the progress property $\bar{p}_j^{(i)} = \theta^T \mathbf{h}_j^{(i)}$. We refer to the parameter vector θ as the “thinking progress vector” (TPV).

To assess whether the estimated progress $\bar{p}_j^{(i)}$ can be more effectively captured by a more complex model, we compare a simple two-layer feedforward network (FFN) with our proposed TPV model. As shown in Figure 2, both models achieve low loss on the test sets, with no observable improvement from the FFN. Guided by the principle of favoring simplicity, and supported by these empirical findings, we select TPV as our method of choice for progress extraction in subsequent experiments.

To improve the predictions, we leverage the model’s autoregressive nature, in which tokens are generated sequentially, and apply exponential smoothing over the prediction history to reduce noise. In Figure 5, we illustrate TPV predictions over problems from the Math-500 test set [15]. The light blue and dark blue points represent the predictions and smoothed predictions correspondingly, while the orange line depicts the ideal loading bar obtained through linear interpolation. Figure 5(a) presents an aggregative view over data points from several thinking trajectories, while Figures 5 (b, c) showcase TPV predictions and smoothed predictions along thinking trajectories of single problems

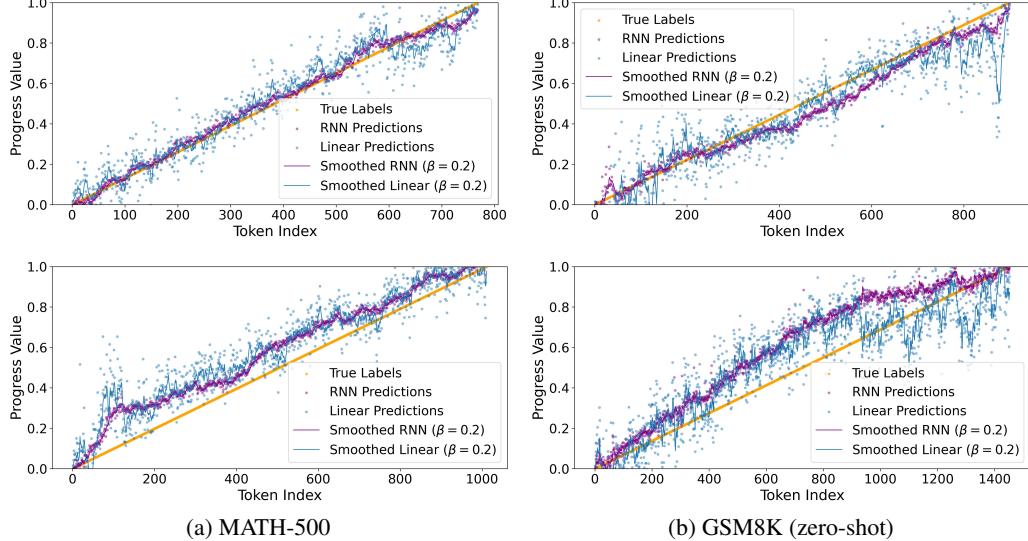


Figure 4: **Qualitative analysis of RNN-based progress prediction:** Predictions across time steps during the thinking phase. (a) Results on MATH-500 (in-domain); (b) results on GSM8K (zero-shot).

in the Math-500 test set. As shown in the figures, both approaches, with and without smoothing, successfully predict the relative position, while the latter produces more precise results that can be used to create a clearer, more interpretable loading bar.

Motivated by this observation, and to better exploit the temporal structure of the progress-bar prediction task, we replace exponential smoothing with a trainable sequence model. Concretely, we adopt a single-layer GRU, parameterized by θ_{RNN} , whose hidden and input dimensions are both set to d , and whose output at each step is a scalar. The network is trained on the sequence dataset \mathcal{D}' , using the same training samples as \mathcal{D} , only with a sequence of relative positions rather than performing single-step predictions:

$$\mathcal{D}' = \{(\mathbf{h}_1^{(k)}, \dots, \mathbf{h}_{N_k}^{(k)}), (p_1^{(k)}, \dots, p_{N_k}^{(k)}) \mid k = 1, \dots, K\}. \quad (2)$$

To extend our analysis, we train the model on the MATH-500 training set and evaluate it on both the MATH-500 and GSM8K test sets, thereby covering both the learned and zero-shot regimes. The results are presented in Figure 3, showing that the RNN consistently achieves lower loss than the linear model in both settings. Although performance drops slightly in the zero-shot regime, the loss remains relatively low, suggesting that the model generalizes well across datasets. Finally, Figure 4 presents a qualitative analysis of the RNN’s predictions over time during the thinking phase. In both learned and zero-shot regimes, the RNN predictions (shown in purple) produce outputs that are smooth, consistent over time steps, monotonically increasing, and accurately reflect the progress of the thinking phase, especially when combined with additional exponential smoothing.

3.2 Controlling Thinking Progress

A crucial question that arises is whether TPVs reflect a fundamental mechanism that the model uses to track its reasoning progress, or if they are merely residual artifacts that correlate with progress but do not play a causal role in the computation. To address this, we conduct intervention experiments.

Intervention Technique. The intervention experiments shift the hidden representation \mathbf{h} in the direction of projection vector θ by an amount α : $\mathbf{h}_\alpha = \mathbf{h} + \alpha\theta$. The altered representation has a new prediction value: $\theta^T \mathbf{h}_\alpha = \theta^T (\mathbf{h} + \alpha\theta) = \bar{p} + \alpha \|\theta\|^2$.

By performing this intervention past all attention layers, we intervene with next token prediction and refrain from editing representation values that are cached and used in consecutive decoding steps. This way, next token predictions along the thinking trajectory are influenced in two ways: (i) locally, by the intervention-edited representation values; (ii) and historically, derived from the auto-regressive process, through the tokens that were already potentially altered as a result of previous interventions.

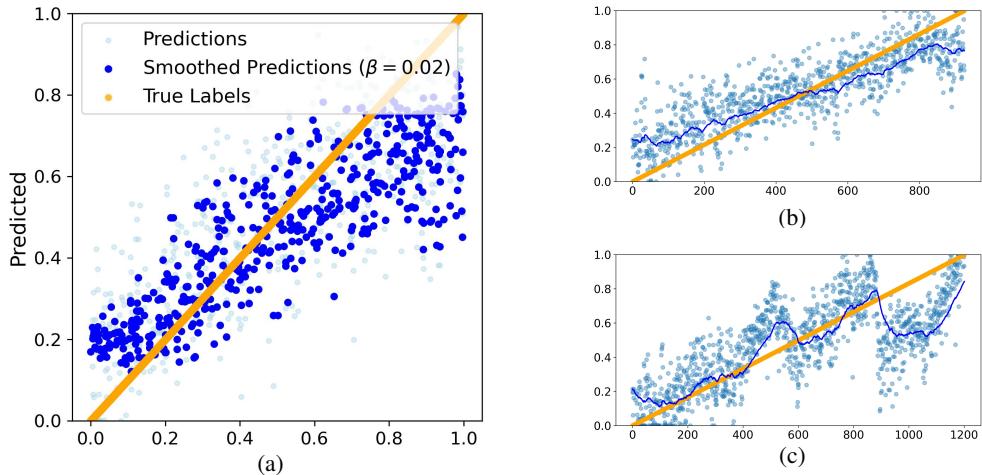


Figure 5: Relative Progress Prediction Analysis: (a) We sample 1000 random tokens from the thinking trajectories across all examples in the test set of the Math-500 dataset. The x-axis denotes the ground-truth relative position within the thinking phase, while the y-axis shows the predicted progress. (b, c) Two representative examples from the Math500 test set, illustrating how the predictor estimates progress across the full thinking trajectory of a single generation. The x-axis denotes the decoding step index, and the y-axis shows the predicted value in the $[0, 1]$ range.

In our experiments, we treat α as a hyperparameter that determines the strength of the intervention. Setting $\alpha = 0$ results in no intervention, preserving the original computation. Positive values of α induce *overclocking*. We hypothesize that overclocking will accelerate the model’s reasoning phase, making it shorter and more decisive.

Overthinking. We analyze the impact of TPV intervention on different thinking patterns. Figure 1(b) compares two thinking sequences generated by the DeepSeek-R1-Distill-Qwen-32B model—with and without intervention. The original sequence is marked by hesitation and verbosity, while the TPV-accelerated version is significantly more concise, using nearly six times fewer tokens. Notably, both trajectories arrive at the correct final answer.

Example 1 illustrates a mathematical problem where the original model, when prompted, delves into a loop of self-verification and indecisiveness, raising repeating arguments and calculations without converging to a conclusion within the limit of 2048 tokens. The accelerated version reaches the correct answer while terminating in less than half of the token limit. The corresponding predicted progress trajectories for both generations are shown in Figure 6. In addition to being shorter, the intervened trajectory shows consistently higher predicted progress beyond a certain decoding step.

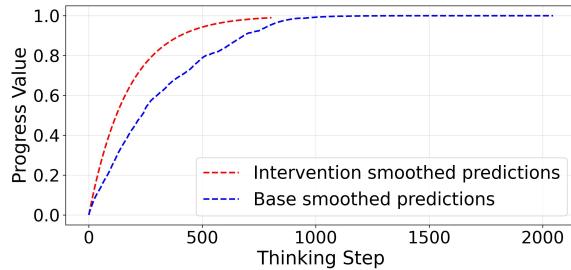


Figure 6: Predicted progress (\bar{p}) values for two thinking sequences, one from the original generation and the other from the TPV-intervened process, on the same prompt in Example 1, using DeepSeek-R1-Distill-Qwen-32B. For this prompt the original generation does not converge to an answer.

<p>Prompt</p> <p>In how many ways can 8 people sit around a round table if 3 of the people – Pierre, Rosa, and Thomas – all want to sit together? (Two seatings are considered the same if one is a rotation of the other.) Please reason step by step, and put your final answer within \boxed{}</p>	<p>Original (Continues Past 2048 Tokens)</p> <p><think> Okay, so I have this problem where I need to figure out how many ways 8 people can sit around a round table. The catch is that Pierre, Rosa, and Thomas all want to sit together. Hmm, circular permutations can be a bit tricky, but let me think through this step by step. . . So, if I consider Pierre, Rosa, and Thomas as one block But wait, . . . But hold on, I want to make sure I didn't miss anything. . . . Is there another way to think about this? Maybe by Wait, no, that might complicate . . . Wait, maybe I made a mistake. . . . Wait, perhaps the first approach is wrong. . . . Wait, no, I think I confused. . . . Wait, maybe I need to think differently. . . . Wait, so if I fix Pierre's seat, then the block is fixed relative to Pierre, Wait, maybe both approaches are correct, but they are counting different things. . . . Wait, no, . . . Wait, let me think again. If we fix Pierre's seat . . .</p>
<p>TPV $\alpha = 100$ (806 Thinking Tokens)</p> <p><think> Okay, so I have this problem: 8 people want to sit around a round table, . . . Alright, let's break this down. I remember that for circular permutations, the number of ways to arrange n people is $(n-1)!$. . . I think the method is correct because we're treating the block as a single unit, so the number of units is 6, and the circular permutations of 6 units is $5!$. Then, within the block, $3!$ arrangements. So, $5! * 3! = 720$. I think that's solid. I don't see any mistakes in the reasoning. So, the answer should be 720. **Final Answer** The number of ways is \boxed{720}. </think></p>	

Example 1: Qualitative examples of indecisive loops and overthinking in the thinking process of the base model (right), which is then mitigated in the text produced by our model (bottom left).

4 Experiments

This section presents a systematic evaluation of the intervention technique described in Sec. 3.2, which overclocks the thinking phase of reasoning LLMs. We assess the method with respect to effectiveness, measured by improvements in answer quality, and efficiency, measured by lower computational cost. The experiments were conducted with two models, DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-LLaMA-8B, which were evaluated on two mathematical benchmarks known to be effective for reasoning LLMs. The full experimental setup is detailed in Appendix A.

Baselines. To comprehensively evaluate our method, we benchmark it against the following baselines: (i) **Base Model:** Running the model on the problem concatenated with the instruction “Please reason step by step, and put your final answer within \boxed{}”, as defined in DeepSeek-R1 prompting guidelines. (ii) **Temperature:** Same as (i) except we generate five samples and select the shortest answer. Following [7], we set the temperature to 0.6. (iii) **Instruct:** A straightforward prompting strategy encourages the model to produce decisive answers with the following prompt: “Please reason step by step, place your final answer inside \boxed{}, and then immediately stop with <|end_of_sentence|>. Present all necessary calculations or arguments concisely, avoiding unnecessary elaboration or verbosity. <think>”.

4.1 Effectiveness

We measure the effectiveness of TPVs in Tables. 1, and 2 which report results for DeepSeek-R1-Qwen-32B and DeepSeek-R1-LLaMA-8B respectively. Each table summarizes performance on two mathematical datasets evaluated under three token budgets. The right half of each table presents GSM-8K scores for budgets of 256, 512, and 1024 tokens, whereas the left half presents Math-500 scores for budgets of 512, 1024, and 2048 tokens.

For each regime we report three key metrics to evaluate model performance: (i) **#Correct** - the number of problems for which the model produced a correct final answer. We define the answer as the expression enclosed within the \boxed{} symbol. This pattern is matched programmatically and verified manually. (ii) **#Answered** - the number of generations in which the model produced

an answer, operationally determined by the presence of the `\boxed{}` symbol in the output. (iii) **#Ended** - the number of generations that concluded naturally before hitting the enforced token limit.

It is important to note that the number of answered problems can exceed the number of completed generations. This is because both models often produce the `\boxed{}` token at the end of the thinking phase, before completing the full response, allowing truncated generations to still include a valid answer. The opposite may also occur: particularly the LLaMA model occasionally fails to follow the instruction to wrap the answer in `\boxed{}`, causing exclusion of otherwise valid answers.

Table 1 reveals four notable trends: (1) **The impact of α :** Increasing α in our method from 5 to 100, both with and without the instruction-based acceleration used in baseline (iii), generally increases the numbers of completed, ended, and correct answers produced by the model, providing further evidence that our intervention method influences the thinking length. In particular, on Math-500, larger values of α increased the number of correct answers by 42,39,7 on token budget of 512,1024 and 2048 respectively. Similarly, on GSM-8K, the number of ended answers increases by 15, 19, and 5 for token budgets of 256, 512, and 1024, respectively.

(2) **Comparing Acceleration Baselines Against the Base Model:** Baselines (ii) and (iii) accelerate the base model using prompting responses and temperature-based ensembling. Unsurprisingly, in most regimes both methods increase all three metrics, demonstrating that they are strong baselines against which to evaluate our overclocking approach. Specifically, on the Math-500 dataset with a 512-token budget, the number of correct answers rises from 36 (base) to 54 under the temperature baseline and 53 under the instruction-based baseline, while the number of answers increases from 38 to 59 for both methods. Finally, the number of completions increases from 28 for the base model to 43 for the temperature baseline and 52 for the instruction-based baseline. Similarly, under the same 512-token limit on GSM-8K, both acceleration strategies yield improvements. The temperature-based approach increases correct responses from 221 to 237, while the instruction-driven method produces 224 correct answers, and the total answers climb from 223 to 249 with the temperature strategy and to 237 with instruction prompts. Completions also see a marked rise, increasing from 28 for the base model to 43 under the temperature baseline and to 52 with instruction-based acceleration.

(3) **Comparing Our Method Against Baselines:** Although these baselines perform strongly and the temperature-based baseline requires roughly five times more compute, our method outperforms both by producing more correct answers and more decisive responses. In particular, under low compute budgets of 256 or 512 tokens, our approach with $\alpha = 100$ increases the number of correct answers on Math-500 by at least 80% in the 512 token-budget regime and boosts correct responses on GSM-8K by an average of 80% across the 256 and 512 token settings. For example, on Math-500 in the 512 token-budget regime, correct answers rise from 54 under the temperature baseline and 53 under the instruction baseline to 100 with our method, resulting in a gain of more than 46 correct answers, while the number of completed answers increases from 59 under both baselines to 106, a gain of 47. Remarkably, these increases in correct answers do not come at the cost of more errors, as the error rate remains unchanged. This suggests our method shortens reasoning without increasing errors, promoting more decisive thinking. Our findings for compute budgets larger than 512 generally follow the same trend, showing improvements in the number of correct answers in most regimes, without an increase in the error rate.

(4) **Complementary Contribution:** Although our empirical findings approve that our method is more effective than the baselines, there are still cases where our method lags behind the prompting-based approach (denoted as 'Instruct'). One prominent example is the regime of token-budget of 2048 on the Math 500, where the instruct baseline answers correctly 10% more than our method. A similar trend was also observed in Table 2, with experiments conducted using LLaMA instead of the Qwen DeepSeek variants. This observation raises the question of whether the improvements are orthogonal or if the two methods directly compete. To investigate, we conducted dedicated experiments in which the instruction-based prompting technique was combined with our intervention method and compared against each individual approach. As shown in the last two rows of both tables, this hybrid approach consistently yields the best performance across most regimes, achieving an improvement of 66% on average and 285% at max over the instruction-based baseline, and 223% on average and 1416% at max over the base model. These findings suggest that our method is complementary to prompting strategies and can be effectively integrated with other acceleration techniques.

Table 1: **Results for DeepSeek-R1-Qwen-32B** on GSM-8K and Math500 at different context lengths (512, 1024, 2048 for Math500; 256, 512, 1024 for GSM-8K), showing #Correct '#Cr', #Answered '#An' and #Ended '#En' for each.

Method	Math500												GSM-8K												
	512			1024			2048			256			512			1024									
	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	
Base	36	38	28	154	156	101	278	280	245	12	16	9	221	223	227	285	298	296	237	233	285	297	293	298	
Temperature	54	59	43	146	149	119	294	305	294	39	40	33	237	249	248	237	249	300	237	232	285	299	300	298	
Instruct	53	59	52	211	214	177	316	321	296	16	18	15	224	237	232	285	299	300	237	232	285	299	300	298	
TPV $\alpha = 5$	58	58	42	162	165	98	277	280	257	23	26	14	226	237	233	285	297	293	237	233	285	297	293	298	
TPV $\alpha = 100$	100	106	94	201	213	167	284	296	268	36	41	29	245	256	252	285	299	298	245	241	285	299	298	298	
TPV $\alpha = 5$ Ins	123	126	109	240	254	218	316	331	306	34	26	34	239	250	248	286	300	300	239	236	286	298	300	300	
TPV $\alpha = 100$ Ins	148	160	142	256	290	279	300	337	330	45	49	40	247	263	261	284	298	298	298	247	241	284	298	298	298

Table 2: **Results for DeepSeek-R1-LLaMA-8B** on GSM-8K and Math500 at different context lengths (512, 1024, 2048 for Math500; 256, 512, 1024 for GSM-8K), showing #Correct '#Cr', #Answered '#An' and #Ended '#En' for each.

Method	Math500												GSM-8K											
	512			1024			2048			256			512			1024								
	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En	#Cr	#An	#En
Base	6	6	11	47	54	124	112	122	273	0	0	2	21	27	99	40	47	247	237	233	285	297	293	298
Temperature	29	31	51	95	104	229	137	156	332	7	7	20	50	60	216	63	74	294	245	241	285	299	298	298
Instruct	50	55	68	107	117	190	153	172	302	7	8	13	66	81	167	85	100	273	237	233	285	297	293	298
TPV $\alpha = 5$	4	5	12	37	42	118	109	117	276	0	0	2	22	26	95	35	43	235	237	233	285	297	293	298
TPV $\alpha = 100$	26	30	41	70	77	170	102	113	273	4	5	7	28	34	118	38	46	223	237	233	285	299	298	298
TPV $\alpha = 5$ Ins	51	55	67	103	112	193	159	175	315	12	12	13	58	72	164	80	95	268	237	233	285	297	293	298
TPV $\alpha = 100$ Ins	91	118	119	124	170	227	142	192	284	27	33	70	80	96	176	81	104	211	237	233	285	297	293	298

4.2 Efficiency

We conduct a series of intervention experiments on the Math-500 and GSM8K datasets by varying the intervention parameter α to overclock the model’s thinking phase. Figure 7 shows that increasing α consistently reduces the length of the thinking phase, making the reasoning process more efficient. These findings support our hypothesis that TPV functions as an active control signal in the model’s internal computation, rather than being a passive correlate. In particular, when applying our method with $\alpha = 100$ to the DeepSeek-R1 LLaMA model using the prompting strategy (baseline iii) on the GSM8K dataset, the average number of tokens decreases from approximately 500 to fewer than 350, resulting in a 30% reduction in compute. Moreover, all positive values of α consistently accelerate the thinking phase relative to the baseline ($\alpha = 0$), while also improving its effectiveness, as shown in Tables 1–2. For additional qualitative examples, see examples in Appendix D.

4.3 Qualitative Analysis

To further evaluate the reliability of TPVs in estimating a model’s position within its reasoning process, we test their performance under two additional conditions: (i) varied prompting strategies and (ii) differing reasoning sequence lengths. Figures 8(a-d) show that TPVs remain effective across diverse instructions, distinct from the original prompt used during training (see caption for the prompts). Figure 8(e) shows consistently low test loss across bins of different thinking sequence lengths, indicating robustness to varying reasoning depths.

To better understand how progress is monitored and how it is modified during the thinking phase, we explored cases in which the behavior of the indicator is not monotonic, see Appendix B. For example it is often the case that the progress prediction increases, until a noticeable drop occurs. This drop in predicted progress often coincides with self-verification behavior, where the model attempts to solve the problem again using an alternative approach.

To further explore TPV’s capacity to capture and explain different reasoning dynamics, we examine how predicted progress responds to specific tokens that suggest either hesitation (“wait”) or advancement (“right”). We quantify this behavior, by measuring the average effect of individual tokens on

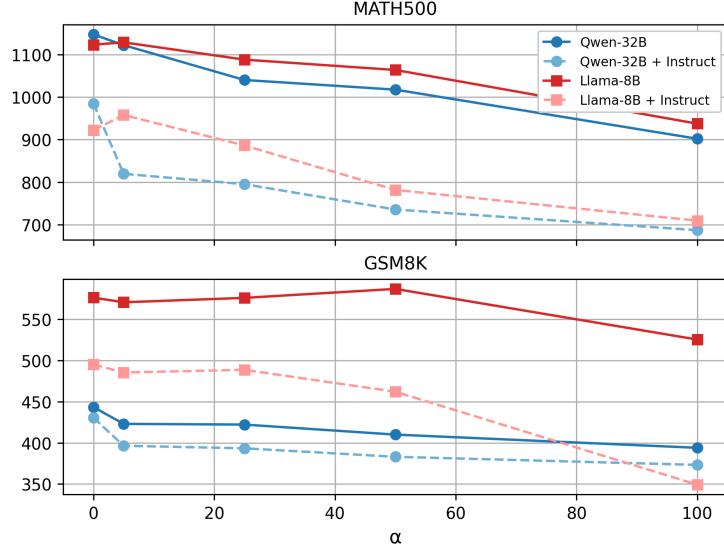


Figure 7: **Impact of α on Intervention:** The x-axis represents different values of α in $\{0, 5, 25, 50, 100\}$, where $\alpha = 0$ indicates no intervention and higher values of α amplify the amount of intervention. The y-axis shows the average number of tokens required to complete the answer. Results are obtained using DeepSeek-R1 models.

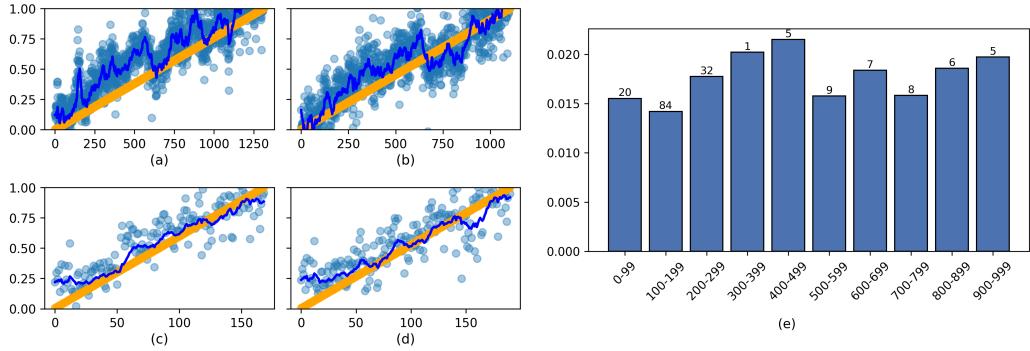


Figure 8: **Robustness of predicted progress under varied prompting styles and reasoning lengths.** **Left:** Predicted and smoothed progress trajectories for the problem “How many 3-digit numbers can be formed using the digits 1 through 9 (no zeros), with no repeated digits?” under four different prompts: (a) “Please reason step by step.” (b) “Think carefully and slowly. Provide a detailed explanation.” (c) “Think quickly and provide a concise answer.” (d) “Present all necessary calculations or arguments concisely; avoid unnecessary elaboration or verbosity.” **Right:** (e) Mean MSE test loss from both Math500 and GSM8K datasets, grouped by thinking phase length (binned). The number of trajectories in each bin is shown above each bar.

predicted progress values across the dataset, see Appendix C. The analysis shows that certain words do tend to significantly shift the predicted progress indicator in a characteristic direction.

5 Discussion

The Nelson and Narens metacognition model [18] distinguishes two interacting levels: the object level, which performs cognitive operations, and the meta level, which engages in monitoring and control over the object-level processes. We show that these two information flows, monitoring and control, are manifested in LLMs, as described in Sections 3.1, and 3.2, respectively. Similarly, Zimmerman [32] describe the process of self-regulated learning and highlights monitoring and self-

reflection as two essential components. In this context, our work shows that LLMs can estimate the number of reasoning steps required to reach a desired answer, and that this estimation evolves in real time based on the tokens generated.

On the applicative side, our method represents an important step toward making reasoning models more transparent and controllable. At the time of writing, millions of users interact with reasoning models such as OpenAI’s O1 or DeepSeek-R1. During inference, particularly in the “thinking phase,” users are often left without visibility into the model’s internal progress or how close it is to producing an answer. By introducing an interactive progress bar that visualizes the model’s internal status during reasoning, we make these models more predictable, responsive, and user-friendly. Furthermore, our approach introduces a plug-and-play intervention mechanism that enables real-time acceleration of the reasoning process through overclocking. This feature is especially useful for users who have an intuition about the complexity of their task and wish to adapt the model’s reasoning depth accordingly. This contributes to the ongoing effort to make reasoning-capable language models more interactive, adaptive, and aligned with human needs.

Limitations. While our approach demonstrates promising results for monitoring and controlling thinking path lengths in LLMs, several important limitations should be acknowledged. First, our investigations are primarily focused on mathematical reasoning tasks, which represent a well-structured domain with clear evaluation metrics. The applicability of our progress vector technique to more open-ended reasoning domains (e.g., ethical reasoning, creative problem-solving, or multi-step planning) remains unexplored. Specifically, while overclocking improves performance on our tested benchmarks, we have not thoroughly investigated potential negative side effects of progress vector manipulation on reasoning coherence or factual accuracy outside our test distributions. Second, our method requires access to model hidden states during inference, making it challenging to implement in black-box or API-only settings where only input-output functionality is available. Finally, our experiments focus exclusively on the length dimension of reasoning, without addressing other important aspects such as reasoning breadth, diversity of approaches considered, or uncertainty estimation.

6 Conclusions

This paper introduces a novel approach to monitoring and controlling thinking path lengths, revealing fundamental mechanisms by which LLMs regulate their reasoning processes. By identifying and manipulating the progress mechanism, we have developed two practical applications with significant implications for AI systems: (i) a dynamic thinking progress visualization that enhances transparency, and (ii) an “overclocking” intervention technique that significantly improves both efficiency and performance. The ability to modulate reasoning depth without sacrificing—and often enhancing—performance challenges conventional assumptions about the relationship between computation and accuracy in LLMs. This suggests that modern LLMs are capable of more efficient reasoning but may benefit from external guidance to avoid overthinking patterns.

7 Acknowledgments

This work was supported by a grant from the Tel Aviv University Center for AI and Data Science (TAD). This research was also supported by the Ministry of Innovation, Science & Technology ,Israel (1001576154) and the Michael J. Fox Foundation (MJFF-022407).

References

- [1] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.
- [2] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- [3] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, pages 6243–6267. PMLR, 2023.

- [4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [5] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- [6] John H Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, 34(10):906, 1979.
- [7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [8] Roei Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Conference on Empirical Methods in Natural Language Processing*, 2023.
- [9] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [10] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.108. URL <https://aclanthology.org/2024.findings-acl.108/>.
- [11] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*, 2024.
- [12] Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL <https://openreview.net/forum?id=CqViN4dQJk>.
- [13] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [14] Jingun Kwon, Hidetaka Kamigaito, and Manabu Okumura. Abstractive document summarization with summary-length prediction. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 618–624, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.45. URL <https://aclanthology.org/2023.findings-eacl.45/>.
- [15] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [16] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- [17] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=9XFSbDPmdW>.

- [18] Thomas O Nelson. Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation*, volume 26, pages 125–173. Elsevier, 1990.
- [19] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. <https://distill.pub/2020/circuits/zoom-in>.
- [20] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
- [21] OpenAI. Chatgpt: Optimizing language models for dialogue. <https://openai.com/chatgpt>, 2023. Accessed: 2025-05-11.
- [22] Fabian Retkowsky and Alexander Waibel. Zero-shot strategies for length-controllable summarization. *arXiv preprint arXiv:2501.00233*, 2024.
- [23] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- [24] Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.
- [25] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. URL <https://arxiv.org/abs/2503.16419>, 2025.
- [26] Yue Wang, Qiuwei Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuseng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.
- [27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [28] Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.
- [29] Tong Wu, Chong Xiang, Jiachen T Wang, and Prateek Mittal. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*, 2025.
- [30] Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.
- [31] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=S5wmbQc1We>.
- [32] Barry J Zimmerman. Becoming a self-regulated learner: An overview. *Theory into practice*, 41(2):64–70, 2002.

A Experimental Setup

This section describes the experimental setup used for the empirical studies in Sec. 4. All experiments were conducted in PyTorch with official Hugging Face models and DeepSeek prompts. Each run used a single A100 GPU and finished within at most two days.

Datasets. We evaluated performance on two publicly available mathematics datasets: GSM-8K and Math-500. For Math-500, we randomly sampled 80 problems to train our loading-bar regressor and used the remaining 420 problems as the test set. For GSM-8K, we randomly sampled 330 problems, reserving the first 30 for training and the remaining 300 for testing. These datasets were chosen because they effectively assess the reasoning capabilities of LLMs, as reasoning has been shown to be remarkably effective on mathematical problems.

B Non-monotonic Progress Prediction

We perform a qualitative analysis of TPV-predicted progress trajectories and find that non-monotonic patterns often correspond to self-verification behaviors in the model’s reasoning. In such cases, the model reapproaches the problem using a different method to confirm its solution. For instance, as illustrated in Figure 9, the model first solves a combinatorics question using the counting principle, then revalidates its answer by applying the permutations formula. This second solving attempt begins at token 665—indicated by a vertical line—which coincides with a clear drop in the predicted progress, reflecting the model’s re-evaluation phase.

(Q) How many 3-digit numbers can be formed using the digits 1 through 9 (no zeros), with no repeated digits?

(T 1) Okay, so I have this problem: How many 3-digit numbers ...
(T 42) First, I know that ...
(T 201) Now, moving on to the second digit ...
(T 332) Now, according to the counting principle ...
(T 572) It is indeed 504. Okay, that seems right ...
(T 630) Let me think about another way ...
(T 665) The formula for permutations is ...
(T 883) I think that solidifies the answer ...

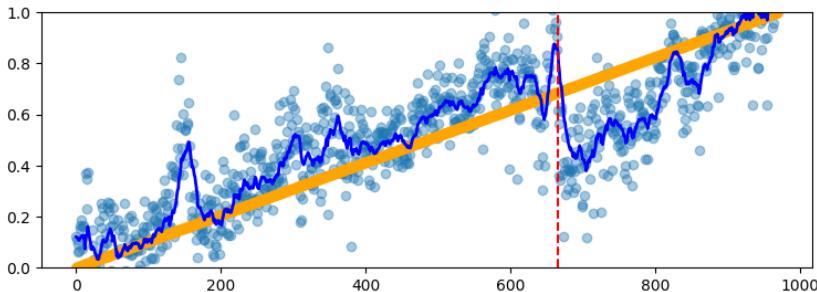


Figure 9: A math question and the associated thinking sequence generated by DeepSeek-R1-Distill-Qwen-32B, with each row indicating the index of its first token. The sequence is paired with its TPV-predicted progress trajectory, highlighting the effect of self-verification behavior within the model’s reasoning process. A vertical line at token index 665 marks a notable drop in predicted progress, aligning with the point where the model begins re-solving the problem.

C Effect of Specific Words on Predicted Progress

When examining thinking sequences qualitatively, certain tokens frequently appear in contexts that suggest either hesitation or progression toward concluding the reasoning process. Words such as

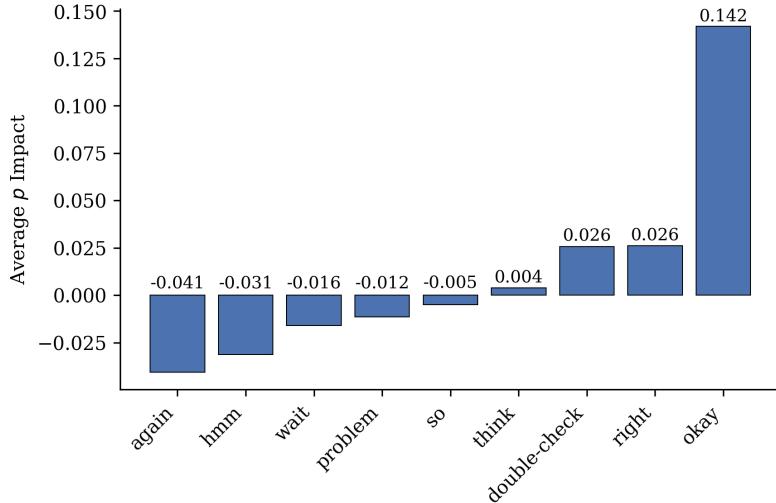


Figure 10: Average change in \bar{p} values caused by significant tokens across the dataset, highlighting their impact on predicted progress.

“hmm,” “wait,” and “right” often signal meaningful transitions in the model’s internal state. Some of these tokens reflect a recognition of inconsistency or error, prompting the model to adjust its reasoning strategy. Others indicate the completion of a reasoning step and a shift toward the final solution. TPVs enable a principled way to quantify this behavior.

We analyze the average influence of individual tokens on predicted progress values across the dataset. As illustrated in Figure 10, tokens with the most negative impact—such as “again,” “hmm,” “wait,” and “problem”—are closely tied to hesitation or reevaluation. In contrast, tokens like “right,” “so,” and “okay” are associated with upward shifts in predicted progress, signaling growing confidence and momentum in the model’s reasoning process.

D Additional Qualitative Analysis

To highlight the impact of our intervention in mitigating overthinking, we present a qualitative analysis in Examples 2–4. Examples 3–4 demonstrate that while both the baseline model and its over-clocked variant arrive at the same answer, the over-clocked model does so more decisively and with significantly fewer thinking tokens. In contrast, Example 2 shows a case where the base model does not succeed in providing an answer within the token limit, in contrast to the over-clocked variant, which provides an answer after 259 thinking tokens.

Prompt	Original (Continues past 2048 tokens)
<p>Suzanne walks four miles every third day. What is the fewest number of miles she can walk in February? Please reason step by step, and put your final answer within \boxed{}</p> <p>TPV $\alpha = 100$ (259 Thinking Tokens)</p> <p><think> Suzanne walks 4 miles every 3 days. To find the fewest number of miles she can walk in February, we need to determine how many 3-day periods are in 28 days (a non-leap year February).</p> <ol style="list-style-type: none"> 1. **Calculate the number of 3-day periods in 28 days:** $\frac{28}{3} \approx 9.33$ <p>Since Suzanne can't walk a fraction of a 3-day period, we take the integer part, which is 9.</p> <ol style="list-style-type: none"> 2. **Determine the total miles walked:** $9 \times 4 = 36 \text{ miles}$ <ol style="list-style-type: none"> 3. **Check if 28 days is a multiple of 3:** $28 \div 3 = 9 \text{ with a remainder of } 1$ <p>Since 28 is not a multiple of 3, 9 periods of 3 days account for 27 days, leaving 1 day. Suzanne doesn't walk on the 28th day, so 36 miles is the minimum.</p> <p>**Final Answer** The number of ways is \boxed{36}. </think></p>	<p><think> Okay, so I need to figure out the fewest number of miles ... First, I should know how many days are in February. ... maybe I can assume it's a non-leap year with 28 days? Or does it matter? Let me think. ...</p> <p>Let me list the days she walks: Day 1, 4, 7, 10, 13, 16, 19, 22, 25, 28.</p> <p>Wait, that's 10 walks. ... Maybe she can start on a later day to minimize the number of walks in February. ...</p> <p>But how late can she start? ... if she starts on day 26, for example, she would walk on day 26, then day 29, but February only has 28 days, ... she would only walk once in February. ...</p> <p>Wait, actually, the problem says she walks four miles every third day. ...</p> <p>But wait, if she starts on day 26, does that mean she didn't walk on day 23? ... she should have walked on day 23 as well. ...</p> <p>If she walks every third day, that means her walking days are spaced three days apart. So, if she walks on day x, she must have walked on day $x - 3$, $x - 6$, etc., and will walk on day $x + 3$, $x + 6$, etc. ...</p> <p>Wait, so starting on day 2 or 3, she has 9 walks, whereas starting on day 1, she has 10 walks.</p> <p>But earlier, I thought she could start on day 26 and only walk once, but that seems incorrect because her walking schedule is every third day, so she can't just start on day 26 without having walked on day 23, 20, etc. ...</p>

Example 2: An example of overthinking through first presenting an argument not aligned with the constraints of the problem and the effort of reconciling it (right); and a straightforward answer presenting a simple solution (left).

Prompt	Original (1872 Thinking Tokens)
<p>Compute the 9th Fibonacci number, assuming $F(1) = 1$. Please reason step by step, and put your final answer within \boxed{}</p> <p>TPV $\alpha = 100$ (293 Thinking Tokens)</p> <p><think> To compute the 9th Fibonacci number, I start by understanding the Fibonacci sequence. The sequence is defined such that each number is the sum of the two preceding ones, with the first two numbers being 1. I will list out the Fibonacci numbers up to the 9th term. Starting with $F(1) = 1$ and $F(2) = 1$, I can calculate each subsequent term by adding the two previous terms: $F(3) = F(2) + F(1) = 1 + 1 = 2$ \dots $F(9) = F(8) + F(7) = 21 + 13 = 34$ By following this sequence, I find that the 9th Fibonacci number is 34. </think></p>	<p><think> Okay, so I need to find the 9th Fibonacci number, and they mentioned that $F(1)$ is 1. Hmm, Fibonacci numbers... I remember they start with 0 and 1, but ... The Fibonacci sequence is where each number is the sum of the two preceding ones. So, if $F(1)$ is 1, then what is $F(2)$? I think it's also 1 because usually, the sequence starts with 0, 1, 1, 2, 3, 5, etc. ... Let me write down the sequence step by step to make sure I get it right.</p> <p>$F(1) = 1$ $F(2) = 1$ \dots $F(9) = F(8) + F(7) = 21 + 13 = 34$ Wait, so $F(9)$ is 34? Let me double-check my calculations to make sure I didn't make a mistake. Starting from $F(1) = 1$: 1. $F(1) = 1$ 2. $F(2) = 1$ \dots 9. $F(9) = 21 + 13 = 34$ Yes, that seems correct. Each time, I'm adding the two previous numbers to get the next one. So, the 9th Fibonacci number is 34. ... let me recall another way ... I remember there's Binet's formula, which uses the golden ratio. The formula is:</p> $F(n) = (\Phi^n - \Psi^n)/\sqrt{5} \dots$ <p>Let me try using this formula to compute $F(9)$ \dots First, compute ... Now, compute ... Let me compute this step by step: ... \dots Similarly, compute ... \dots Now, plug these into Binet's formula:</p> $F(9) = (\Phi^9 - \Psi^9)/\sqrt{5} \approx (76.013 - (-0.01316))/2.23607 \approx \dots \approx 34.0$ <p>Wow, that's exactly 34. So, using Binet's formula also gives me 34 for $F(9)$. That confirms my earlier result. Alternatively, I could have used the recursive formula ... Just to make sure, let me list out the Fibonacci numbers from $F(1)$ to $F(9)$:</p> <p>1. $F(1) = 1$ 2. $F(2) = 1$ \dots 9. $F(9) = 34$ Yep, that's consistent. So, I'm confident that the 9th Fibonacci number is 34. </think></p>

Example 3: An example of overthinking through excessive self-verification and redundant, duplicated computation in the thinking process of the base model (right), which is mitigated in the text produced by our model (bottom left).

<p>Prompt</p> <p>Solve $X + Y = 5$, $3X - Y = 8$. Please reason step by step, and put your final answer within <code>\boxed{}</code></p> <p>TPV $\alpha = 100$ (202 Thinking Tokens)</p> <p><think> To solve the system of equations: 1. Add the two equations to eliminate Y:</p> $(X + Y) + (3X - Y) = 5 + 8$ $4X = 13$ $X = \frac{13}{4}$ <p>2. Substitute X back into the first equation to find Y:</p> $\frac{13}{4} + Y = 5$ $Y = 5 - \frac{13}{4} = \frac{20}{4} - \frac{13}{4} = \frac{7}{4}$ <p>3. Final Answer:</p> $X = \frac{13}{4}, \quad Y = \frac{7}{4}$ <p></think></p>	<p>Original (892 Thinking Tokens)</p> <p><think> Okay, so I have these two equations here: $X + Y = 5$ and $3X - Y = 8$. I need to solve for X and Y.</p> <p>Hmm, let me think about how to approach this. I remember from class that there are a couple of methods to solve systems of equations, like substitution and elimination. Maybe I can use one of those here.</p> <p>Let me write down the equations again ...</p> <p>Hmm, ... Let me try that.</p> <p>Adding equation 1 and equation 2:</p> $(X + Y) + (3X - Y) = 5 + 8$ <p>Simplify the left side:</p> $X + Y + 3X - Y$ <p>Combine like terms:</p> <p>$X + 3X$ is $4X$, and $Y - Y$ is 0. So, $4X = 13$</p> <p>Wait, $5 + 8$ is 13, so $4X = 13$. Then, to solve for X, I can divide both sides by 4.</p> <p>$X = 13/4$</p> <p>Hmm, ...</p> <p>Okay, now that I have X, I can plug it back into one of the original equations to find Y.</p> <p>Let me use equation 1 because it looks simpler: $X + Y = 5$.</p> <p>Substituting $X = 13/4$ into equation 1:</p> $13/4 + Y = 5$ <p>To solve for Y, I can subtract $13/4$ from both sides.</p> $Y = 5 - 13/4$ <p>Hmm, 5 is the same as $20/4$, right? So, ...</p> <p>So, $Y = 7/4$, which is 1.75 or 1 and $3/4$.</p> <p>Let me double-check my work ...</p> <p>Starting with the two equations: ...</p> <p>Which simplifies to ...</p> <p>Then, plugging X back into equation 1: ...</p> <p>Subtract $13/4$: ...</p> <p>That also seems correct.</p> <p>Just to be thorough, let me plug ...</p> <p>... Yes, that works out. So, both equations are satisfied with $X = 13/4$ and $Y = 7/4$.</p> <p>I think that's it. I used the elimination method by adding the two equations to eliminate Y, solved for X, then substituted back to find Y. Checked my solution in both equations, and it works.</p> <p>So, I feel confident that X is $13/4$ and Y is $7/4$.</p> <p></think></p>
--	---

Example 4: An example of overthinking through excessive self-verification and redundant, duplicated computation in the thinking process of the base model (right), which is mitigated in the text produced by our model (bottom left).