



UNIVERSITY OF
CAMBRIDGE

Department of Engineering

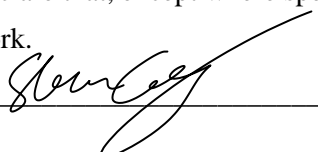
Mitigating Hallucinations in LLMs

Author Name: Stephen Cowley

Supervisor: Dr Marcus Tomalin

Date: 02/06/2025

I hereby declare that, except where specifically indicated, the work submitted herein is my own original work.

Signed  date 2/6/2025

Mitigating Hallucinations in LLMs:

Technical Abstract

Author: Stephen Cowley, Magdalene College

Supervisor: Dr Marcus Tomalin

Date: 2nd June 2025

Large Language Models (LLMs) demonstrate exceptional capabilities across a range of tasks, but despite their rapid improvement, they continue to suffer from a failure mode known as *hallucination*—the generation of information that is coherent but factually incorrect. As LLMs become more integrated into real-world domains such as software and law, mitigating hallucinations becomes increasingly essential to prevent misinformation with potentially serious consequences. This report investigates practical strategies aimed at producing more reliable LLM outputs by reducing such hallucinations.

The report is divided into two main parts, reflecting two related research directions. The first part, *Additive Context-Aware Decoding: Addressing Limitations of Contrastive Decoding*, investigates decoding strategies in scenarios where relevant contextual information is available. While practical systems often retrieve extracts from external sources such as Wikipedia, LLMs can sometimes ignore this context in favour of inaccurate internal beliefs—so-called knowledge-conflict scenarios. We explore a novel combination of two prior techniques: Context-Aware Decoding (CAD), which biases output toward provided context; and Decoding by Contrasting Layers (DoLa), which amplifies internally stored factual knowledge. We find that combining these methods (a technique we call *CAD-DoLa*) yields little improvement in knowledge-conflict scenarios, but partially mitigates drawbacks of CAD in general knowledge settings where relevant context is supplied.

Through visualising CAD’s influence on output distributions, we motivate a new variant, *Additive CAD*, which replaces the multiplicative weightings of CAD with additive shifts in token probabilities. This modification improves accuracy in knowledge-conflict cases but degrades performance in general knowledge scenarios, suggesting a trade-off between overriding model priors and preserving broad factual coverage.

The second part of the report, *Wider Exploration of Idea Space with Doubt Injection*, focuses on reasoning models where external context is unavailable. These models generate a Chain-of-Thought (CoT) before arriving at a final answer, mimicking a logical reasoning process. While CoT techniques improve performance on complex tasks, interventions during generation in the CoT itself remain underexplored. We investigate whether encouraging wider exploration of ideas during CoT improves robustness on adversarial questions. Based on observations that higher-temperature sampling promotes idea diversity, we propose *Doubt Injection*, a method that inserts uncertainty-inducing phrases into the CoT to steer the model toward alternative reasoning paths.

We find that Doubt Injection produces small gains in accuracy on adversarial and arithmetic tasks. However, by introducing a simple statistical framework, we show that these gains have only a 58% and 55% probability, respectively, of reflecting true improvements, due to stochastic variation and limited sample sizes.

Overall, this work identifies both promise and limitations in contrastive decoding and Chain-of-Thought intervention as viable strategies for mitigating hallucinations in LLMs.

Contents

1	Introduction	3
1.1	Motivation	3
1.2	Technical Background	3
1.2.1	Large Language Models (LLMs)	3
1.2.2	LLM Hallucination	4
1.2.3	Retrieval-Augmented Generation	5
1.2.4	Chain-of-Thought (CoT)	5
2	Additive Context-Aware Decoding: Addressing Limitations of Contrastive Decoding	6
2.1	Method	6
2.1.1	Context-Aware Decoding (CAD)	6
2.1.2	Additive CAD (Proposed Method)	6
2.1.3	Decoding by Contrasting Layers (DoLa)	6
2.1.4	CAD-DoLa	7
2.1.5	Additive CAD-DoLa	7
2.1.6	Adaptive Plausibility Constraint (APC)	7
2.2	Experimental Setup	8
2.2.1	Datasets	8
2.2.2	Metrics	9
2.2.3	Models	9
2.2.4	Other Experimental Details	9
2.3	Results and Discussion	10
2.3.1	Reproduction of CAD Results on the MemoTrap Benchmark	10
2.3.2	Reproduction of CAD Results on the Natural Questions Benchmark	11
2.3.3	CAD-DoLa	11
2.3.4	Probability Simplex Visualisation	12
2.3.5	Additive CAD	15
2.3.6	Additive CAD-DoLa	15
2.3.7	Addressing Inconsistencies Across Papers	16
2.3.8	The Effect of Prompt on Natural Questions Performance	17
2.3.9	Sensitivity to Decoding and Evaluation Settings	18
3	Wider Exploration of Idea Space with Doubt Injection	19
3.1	Method	19
3.2	Experimental Setup	20
3.2.1	Datasets	20
3.2.2	Metrics	21
3.2.3	Models	21
3.2.4	Other Experimental Details	21
3.2.5	Calculation of Confidence Intervals	22
3.3	Results and Discussion	24
3.3.1	Motivating Results (the Altered River Crossing Problem)	24
3.3.2	Doubt Injection: Effect of Temperature	26
3.3.3	Effect of Injection String	28
3.3.4	Effect of Injection Probability	29
3.3.5	Effect of Injection Probability on the AIME 2024 Benchmark	29
3.3.6	Increasing the Generation Limit on the AIME 2024 Benchmark	30

4	Conclusions	32
4.1	Future Work	32
	References	35
	Appendix	36
A	Prompts and Responses	36
A.1	CAD Prompt Templates	36
A.2	Doubt Injection Prompts	36
A.3	Doubt Injection Example	37
B	Additional Experimental Results	38
B.1	CAD-DoLa and Additive CAD-DoLa	38
B.2	Doubt Injection	40
C	Risk Assessment Retrospective	41

1 Introduction

1.1 Motivation

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language tasks, including question answering and logical reasoning (Petroni et al., 2019; OpenAI, 2025b; Guo et al., 2025). Despite their success, LLMs frequently suffer from a critical failure mode known as *hallucination*—the generation of information that is coherent but factually incorrect (Ji et al., 2023). These hallucinations can range from benign factual errors to potentially dangerous misinformation in high-stakes domains such as law or medicine (Bohannon, 2023; Huang et al., 2025). As LLMs become more integrated into real-world applications, mitigating hallucination becomes essential to ensuring their reliability and safety.

A number of strategies have been proposed to reduce hallucinations. Among them are *Context-Aware Decoding* (CAD) (Shi et al., 2024), which biases model outputs toward external contextual information over a model’s internal prior beliefs by comparing outputs with and without relevant information. Another is *Decoding by Contrasting Layers* (DoLa) (Chuang et al., 2023), which has been shown to improve factual accuracy by strategically amplifying differences between intermediate representations across internal model states. While both methods have shown promise independently, it remains unclear whether they are complementary. The first half of this project (Section 2) investigates whether a combination of these techniques can improve performance over their individual implementations.

In parallel with new decoding strategies, a second promising research direction focuses on improving the reasoning capabilities of LLMs to enhance their factual accuracy—an important subtask within the challenge of hallucination mitigation. This is achieved through built-in *Chain-of-Thought* (CoT) generation (OpenAI, 2025a; Guo et al., 2025). CoT techniques train models to articulate a step-by-step reasoning process, often mimicking human

logical progression. This not only enhances performance on complex tasks but also makes it easier for humans to audit model outputs. However, there is limited quantitative understanding of how ideas are developed across a CoT sequence, and little research has been done on actively intervening in a CoT during generation.

The second half of this report (Section 3) investigates this underexplored area, where we propose a method for modifying the CoT mid-generation. The method aims to encourage the reasoning process to explore a broader set of ideas, thereby increasing the probability of arriving at correct conclusions.

Together, these two strands of research aim to contribute toward the development of more reliable strategies for mitigating hallucinations in LLMs—which is essential in improving their safety and suitability for practical applications.

1.2 Technical Background

1.2.1 Large Language Models (LLMs)

Large Language Models (LLMs) can be abstracted—for the purposes of this report—as highly sophisticated *next word predictors*. Given an input message (or *prompt*), an LLM generates a likely next word in the sequence. Passing the original prompt plus the generated word as input back into the same LLM allows us to generate a second word. This can be repeated iteratively to generate an entire response. This is known as autoregressive generation.

Although conceptually similar, LLMs use *tokens*, not words, to represent language. Tokens can be whole words or parts of words, and may be combined with punctuation. The set of tokens in use by an LLM is known as its *vocabulary*. The set of corresponding token IDs is denoted by $\mathcal{V} = \{0, 1, \dots, V-1\}$ where V is the vocabulary size.

The input prompt, in text, is first broken up and encoded into a sequence of tokens. This sequence can be written as:

$$\mathbf{w}_{<t} = [w_1, w_2, \dots, w_{t-1}] \quad (1)$$

Where each $w_i \in \mathcal{V}$. The LLM then generates a probability distribution over possible next tokens given this input sequence:

$$P(w_t \mid \mathbf{w}_{<t}) \quad (2)$$

This probability distribution is then sampled from to generate the next token in the sequence.

Temperature The model’s raw outputs before normalisation are referred to as *logits*, which are real-valued scores associated with each token in the vocabulary:

$$\text{logit}(w_t \mid \mathbf{w}_{<t}) \in \mathbb{R} \quad \forall w_t \in \mathcal{V} \quad (3)$$

These logits are converted into probabilities via the softmax function, often scaled by a factor known as the *temperature*, T :

$$\begin{aligned} P(w_t \mid \mathbf{w}_{<t}) &= \text{softmax}\left(\frac{1}{T}\text{logit}(w_t \mid \mathbf{w}_{<t})\right) \\ &= \frac{\exp\left(\frac{1}{T}\text{logit}(w_t \mid \mathbf{w}_{<t})\right)}{\sum_{w \in \mathcal{V}} \exp\left(\frac{1}{T}\text{logit}(w \mid \mathbf{w}_{<t})\right)} \end{aligned} \quad (4)$$

The temperature parameter T controls the sharpness of the resulting probability distribution. A higher temperature results in a flatter distribution, increasing randomness in token sampling. Lower temperature settings cause the distribution to become more peaked, favouring higher-logit tokens. As $T \rightarrow 0$, the distribution approaches a one-hot vector, resulting in deterministic decoding by selecting the token with the highest logit.

In this paper, the temperature setting plays an important role in shaping model behavior, and its value is specified for each experiment.

1.2.2 LLM Hallucination

The phenomenon of hallucination in LLMs has received significant attention within the natural language processing (NLP) community, prompting efforts to define and categorize its various forms (Rawte et al., 2023; Huang et al., 2025). For instance, Ji et al. (2023) define hallucination broadly as “generated content that is nonsensical or unfaithful to the provided source content”, and classify it into two pri-

mary types: *intrinsic hallucinations*, where the output contradicts the input content, and *extrinsic hallucinations*, where the output cannot be validated using the input. In the context of LLMs, however, this definition expands. Because LLMs leverage vast amounts of parametric knowledge acquired during training, Ji et al. (2023) also define hallucination as content that deviates from facts present in the training data.

However, in practical applications, hallucination is often used interchangeably with factual inaccuracy. For instance, OpenAI (2025b) refer to hallucinations as cases where “a model makes factual errors”, and Anthropic (2025) describe them as outputs that are “factually incorrect or inconsistent with the given context”.

To evaluate hallucination empirically, OpenAI reports both accuracy and a dedicated *hallucination rate* on benchmarks such as SimpleQA (Wei et al., 2024) – a dataset consisting of short-form factual questions. It is important to note that hallucination rate is not simply the complement of accuracy: a model may respond with “I don’t know”, which is neither a hallucination nor a correct answer. Since distinguishing hallucinations from other incorrect responses requires manual annotation—rendering large-scale evaluation impractical—accuracy is commonly used as a proxy for hallucination in the literature, a convention we adopt in this paper.

In line with other studies (Shi et al., 2024; Wang et al., 2024; Yuan et al., 2024), we use model accuracy on a general knowledge benchmark as a practical proxy for hallucination rate—where higher accuracy is taken to indicate lower hallucination rate.

We also evaluate model performance on tasks where source context may conflict with parametric knowledge (Section 2), as well as on benchmarks involving social intelligence, adversarial questions, and mathematical reasoning (Section 3). These tasks, although more complex, ultimately evaluate the model’s ability to produce factually correct final outputs.

Not all failure modes captured by evaluations on these benchmarks are necessarily due to hallucination since the model may refuse to

answer. However, errors made in arithmetic or logical reasoning can be considered to be factual inaccuracies (e.g., “2+2=5” is clearly factually incorrect). As such, they fall within the broader scope of hallucination.

1.2.3 Retrieval-Augmented Generation

Among a range of proposed methods to reduce LLM hallucination, Retrieval Augmented Generation (RAG)—introduced by [Lewis et al. \(2020\)](#)—stands out as one of the most prominent.

RAG enhances factuality by retrieving trusted information from external sources (such as a knowledge base or the web) via an information retrieval system. The retrieved documents are then provided to the LLM alongside the original prompt, grounding the response in verifiable content. Techniques like Dense Passage Retrieval (DPR) ([Karpukhin et al., 2020](#)) enable the efficient selection of highly relevant passages from large corpora such as Wikipedia.

As a benchmark, the Natural Questions dataset ([Kwiatkowski et al., 2019](#)) provides general knowledge question-answer pairs, each accompanied by a relevant Wikipedia excerpt. In our experiments (Section 2), we use this dataset to investigate an LLM’s ability to utilise supplied context effectively. Our use of Natural Questions simulates a RAG-like setup, under the assumption that retrieval has already been performed.

However, RAG does not always ensure that the model output faithfully reflects the retrieved content. As shown by [Wu et al. \(2024\)](#), LLMs may sometimes *override* retrieved evidence with their own internal, parametric knowledge—effectively ignoring the external context. Whether this behaviour is desirable or not depends on the application. We investigate approaches to resolve such conflicts between retrieved and internal knowledge in Section 2.

Finally, RAG is not universally applicable. In Section 3, we examine LLM performance in scenarios where no relevant context is available, and the model must rely solely on internal reasoning.

1.2.4 Chain-of-Thought (CoT)

A Chain-of-Thought (CoT) refers to a sequence of intermediate reasoning steps produced by an LLM that lead to a final answer. In arithmetic or multi-step reasoning tasks, this sequence may include sub-calculations, simplifications, assumptions, or logical deductions.

[Wei et al. \(2022\)](#) introduced the technique of Chain-of-Thought Prompting, in which a few examples demonstrating explicit reasoning chains are included in the input prompt. This approach has been shown to substantially improve LLM performance on a range of complex reasoning tasks by encouraging models to generate structured intermediate steps before producing a final answer.

More recent developments have focused on training models that autonomously generate Chain-of-Thought reasoning, without requiring explicit prompting. These models have achieved state-of-the-art performance on benchmarks involving arithmetic reasoning, scientific questions, and programming tasks. For example, OpenAI’s o-series models ([Jaech et al., 2024](#)) introduced the use of reinforcement learning during training to directly optimize for reasoning quality. These models implement *inference-time scaling*, which allow models to generate longer, more elaborate reasoning chains when necessary.

Building on this foundation, DeepSeek-R1 ([Guo et al., 2025](#)) demonstrated that Chain-of-Thought reasoning can emerge naturally through reinforcement learning alone—without the need for supervised fine-tuning. By directly rewarding correct answers to complex questions, the model learns to exhibit behaviors such as self-verification, error correction, and reflection. This can result in coherent and often lengthy reasoning chains, which in turn leads to strong performance across a wide range of reasoning benchmarks.

Since the field remains relatively new, many potential modifications to the Chain-of-Thought approach remain unexplored. In Section 3, we propose a method for modifying the reasoning process during generation, aimed at reducing hallucination in the final answer.

2 Additive Context-Aware Decoding: Addressing Limitations of Contrastive Decoding

2.1 Method

2.1.1 Context-Aware Decoding (CAD)

Context-Aware Decoding (CAD), introduced by Shi et al. (2024), applies a *contrastive decoding* strategy (Li et al., 2022a) to amplify the changes in output probabilities when a model is used with and without access to some helpful contextual information. In a practical setting, the context may be a short extract useful in answering the question, obtained by a separate retrieval system.

CAD has been shown to be particularly effective in overriding a model’s prior knowledge when it conflicts with the provided contextual information (Shi et al., 2024). This might include situations where retrieved context contains factual information that wasn’t available at training time (e.g., the results of a recent scientific discovery).

Formally, representing strings of text as tokenised sequences as in Equation 1, let \mathbf{x} denote a question or query, \mathbf{c} some useful context, and $\mathbf{y}_{<t}$ the sequence of previously generated tokens. Let $P(y_t | \cdot)$ denote the output probability assigned by the LLM to a token y_t given an input sequence, as defined in Equation 2. The output distribution under CAD is defined as:

$$P_{\text{CAD}}(y_t) \propto P(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) \left(\frac{P(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})}{P(y_t | \mathbf{x}, \mathbf{y}_{<t})} \right)^\alpha \quad (5)$$

That is, the token distribution with context is scaled by the pointwise ratio between the distributions with and without context, raised to the CAD parameter $\alpha \in \mathbb{R}$. The resulting expression is then scaled by a multiplicative constant to yield a valid probability distribution.

Setting $\alpha = -1$ simplifies the expression to $P(y_t | \mathbf{x}, \mathbf{y}_{<t})$, corresponding to regular no-context decoding. Setting $\alpha = 0$ simplifies the expression to $P(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$, corresponding

to regular with-context decoding.

Increasing α beyond 0 *amplifies* the token probabilities that have increased once context is given. Large α biases the resulting output distribution toward those tokens that have increased the most—causing the resulting output to be biased in favour of the supplied contextual information.

2.1.2 Additive CAD (Proposed Method)

As will be motivated by a case study in Section 2.3.4, multiplying by a ratio of probability distributions can sometimes cause extremely small probabilities—when paired with a large ratio—to be disproportionately amplified, occasionally resulting in nonsensical outputs. To address this, we introduce *Additive CAD*.

Using the same notation as above, the unnormalised output distribution under Additive CAD is given by:

$$P'_{\text{AddCAD}}(y_t) = P(y_t | \mathbf{x}, \mathbf{y}_{<t}) + \gamma (P(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) - P(y_t | \mathbf{x}, \mathbf{y}_{<t})) \quad (6)$$

That is, the no-context token distribution is adjusted by the pointwise difference between the with-context and no-context distributions, scaled by the Additive CAD parameter $\gamma \in \mathbb{R}$. To ensure a valid probability distribution, the result is renormalised:

$$P_{\text{AddCAD}}(y_t) = \frac{P'_{\text{AddCAD}}(y_t)}{\sum_{y \in \mathcal{V}} P'_{\text{AddCAD}}(y)} \quad (7)$$

Setting $\gamma = 1$ simplifies $P_{\text{AddCAD}}(y_t)$ to $P(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$, corresponding to regular with-context decoding.

Similar to increasing α beyond 0 with regular CAD, increasing γ beyond 1 *amplifies* the token probabilities that have increased once context is given. Large γ similarly biases the resulting output distribution toward those tokens that have increased the most, but this time in an additive—not multiplicative—manner.

2.1.3 Decoding by Contrasting Layers (DoLa)

Decoding by Contrasting Layers (DoLa) is a method introduced by Chuang et al. (2023), shown to improve truthfulness across a variety

of multiple-choice and open-ended generation tasks. The approach contrasts the final output distribution of a language model with the output distribution produced by an earlier *transformer layer*, using a pointwise-ratio technique similar to that of Equation 5 for CAD.

Transformer-based language models consist of multiple *layers*, each applying transformations to the vector representations of tokens produced by the layer before it. The number of layers is fixed at design time—e.g., LLaMA-2-7B used in this paper includes 32 layers (Touvron et al., 2023).

An *early exit* at layer l involves decoding the output at some layer l —other than the final layer—into a probability distribution over tokens, denoted here as $P^{(l)}(y_t)$. DoLa computes a new distribution by contrasting this early-layer distribution with the final output distribution $P(y_t)$ as follows:

$$P_{\text{DoLa}}(y_t) \propto P(y_t) \left(\frac{P(y_t)}{P^{(l)}(y_t)} \right) \quad (8)$$

Similarly to CAD, this *amplifies* the tokens that have increased in probability between layer l and the final layer. Since factual knowledge in an LLM has been shown to be localised to particular transformer layers, DoLa amplifies the *factuality* of the output distribution.

To select the layer whose early exit output is most different from the final layer output, the layer l is dynamically selected at each decoding step as the one whose output distribution maximizes the *Jensen–Shannon divergence* (JSD) from the final output. The JSD is a symmetric variant of the Kullback–Leibler (KL) divergence $D(P\|Q)$ between two distributions P and Q , defined as:

$$\text{JSD}(P\|Q) = \frac{1}{2}D(P\|M) + \frac{1}{2}D(Q\|M) \quad (9)$$

where $M = \frac{1}{2}(P + Q)$ is the mixture distribution.

Additionally, it is possible to restrict premature layer selection to either the lower or upper half of the model’s transformer layers. For the main DoLa results reported in this paper, selection was limited to the lower half. How-

ever, for many of the results provided, DoLa was also evaluated using upper layer selections, with these additional results provided in Appendix B.1. Since upper layer configurations generally yielded little to no improvement over those using lower layers, they are excluded from the main results for clarity.

Hugging Face’s transformers library offers a convenient way to implement DoLa in practice¹.

2.1.4 CAD-DoLa

To investigate whether the DoLa method can enhance the performance of CAD, we propose a hybrid approach termed *CAD-DoLa*. This method retains the core contrastive decoding formulation of CAD (Equation 5), but augments either one or both of the distributions— $P(y_t \mid c, x, y_{<t})$ and $P(y_t \mid x, y_{<t})$ —using the DoLa procedure described in Section 2.1.3.

2.1.5 Additive CAD-DoLa

In parallel to the CAD-DoLa variant, we propose *Additive CAD-DoLa*, which applies the DoLa augmentation within the Additive CAD framework. Specifically, this method follows the additive contrastive decoding strategy of Additive CAD (Equation 6), but augments either or both of the component distributions with the DoLa approach, as described in Section 2.1.3.

2.1.6 Adaptive Plausibility Constraint (APC)

The Adaptive Plausibility Constraint (APC), proposed by Li et al. (2022b), is an optional method designed to prevent a contrastive decoding process such as CAD from amplifying otherwise implausible tokens. It does so by suppressing tokens that are not sufficiently likely under the base model.

Formally, let $P(y_t)$ denote the token distribution over the vocabulary \mathcal{V} produced after applying some contrastive decoding method. The APC modifies this distribution by setting

¹See https://huggingface.co/docs/transformers/en/generation_strategies#dola for implementation details.

the probability of token y_t to zero if its original likelihood under the base model, conditioned on the context, falls below a threshold $\beta \in [0, 1]$:

$$P_{\text{APC}}(y_t) \propto \begin{cases} P(y_t) & \text{if } P(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) > \beta \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Throughout this paper, whenever CAD is applied, we set the APC threshold to $\beta = 0.1$, following the setup established in prior work on contrastive decoding (Li et al., 2022b; Chuang et al., 2023).

Since Additive CAD does not take the ratio of token probabilities, the risk of amplification of unlikely tokens is lower—hence the APC is not applied to experiments involving Additive CAD.

2.2 Experimental Setup

The evaluation scripts used in this section are available on GitHub².

2.2.1 Datasets

The CAD and Additive CAD methods are tested on datasets that specifically provide both an input prompt \mathbf{x} and some form of context \mathbf{c} . This allows for the LLM output distributions with and without context to be contrasted against one another as defined by the CAD and Additive CAD equations (Equation 5, Equation 6).

Knowledge conflicts We evaluate performance on the 860-question MemoTrap dataset (Liu and Liu, 2023). MemoTrap is designed to test the susceptibility of LLMs to fall into memorisation traps. For instance, a typical question would be ‘Write a quote that ends in the word “thoughts”: Actions speak louder than’. It is common for an LLM’s output to be “words” (which is the standard ending of the classic proverb “Actions speak louder than words”), despite the prompt specifically instructing that a different ending be made. The tendency for models to defy instructions as

²See <https://github.com/stephen1cowley/memotrap-testing>.

Table 1: Example prompts \mathbf{x} , contexts \mathbf{c} and correct (gold) answers \mathbf{g} for the MemoTrap and Natural Questions datasets. The prompts and context form the inputs to CAD and Additive CAD according to the implementation details provided in section 2.1.

MemoTrap	
c	Write a quote that ends in the word "early":
x	Better late than
g	early
Natural Questions	
c	Ashrita Furman (born Keith Furman, September 16, 1954) is a Guinness World Records record-breaker. As of 2017, Furman has set more than 600 official Guinness Records and currently holds more than 191 records, thus holding the most Guinness world records. He has been breaking records since 1979.
x	who holds the world record for the most world records?
g	Ashrita Furman

in this example is viewed as undesirable behaviour. An example prompt, context and correct (gold) answer is provided in Table 1.

General Knowledge The Natural Questions dataset (Kwiatkowski et al., 2019) is a question-answer (QA) dataset. We use a filtered 2,773-example variant that includes only short-form (1-2 word) answers.

The context in this case is an excerpt of a Wikipedia article on the topic relevant to the particular general knowledge question. This dataset therefore tests a model’s ability to answer general knowledge questions given additional information, which in practice could be provided by—for instance—a basic RAG system which retrieves Wikipedia article excerpts based on similarity to the input prompt. An example prompt, context and correct (gold) answer is provided in Table 1.

2.2.2 Metrics

The main metric used in the results of Section 2.3 are reported with an *exact match* (EM) score. A more relaxed *substring match* score is also used toward the end of the section. Both methods ignore character case, punctuation, articles and extra whitespace, and are detailed below.

Exact Match (EM) score We adopt a multi-reference exact match score, following the widely-used definition introduced by Rajpurkar et al. (2016) and its implementation in the official evaluation script³. A score of 1 is assigned if the model’s output exactly matches *any* of the gold reference answers, and 0 otherwise. This metric is chosen as it matches the metric employed by the original CAD paper (Shi et al., 2024) on the same MemoTrap and Natural Questions benchmarks.

Formally, let $G^{(q)} = \{g_1^{(q)}, g_2^{(q)}, \dots, g_n^{(q)}\}$ be the set of *normalised*⁴ gold answer strings for question q of a particular dataset, and let $a^{(q)}$ be the normalised string of the model’s decoded predicted answer. For a Q -question dataset, we define a multi-reference exact-match score for question q as follows:

$$\text{Match}^{(q)} = \begin{cases} 1 & \text{if } a^{(q)} \in G^{(q)} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

And the overall EM score across the dataset:

$$\text{AvgMatch} = \frac{1}{Q} \sum_{q=1}^Q \text{Match}^{(q)} \quad (12)$$

If there is only *one* gold answer $g^{(q)}$ for each question in the dataset, the multi-reference exact match score collapses into the single-reference EM score for one gold answer, with the EM match for each question computed as:

$$\text{Match}^{(q)} = \begin{cases} 1 & \text{if } a^{(q)} = g^{(q)} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

³See <https://github.com/rajpurkar/SQuAD-explorer/blob/master/evaluate-v2.0.py#L98> for the exact implementation.

⁴Lowercased string with punctuation, articles and extra whitespace removed.

and the overall accuracy similarly computed as the mean of these match scores across all Q questions in the dataset.

Since there are multiple gold answers for some questions in the Natural Questions dataset, the EM score for Natural Questions is computed as in Equation 11.

However, since there is only one gold answer for each question in the MemoTrap dataset, the simplification in Equation 13 is applied.

Substring match score The substring match score is a more relaxed score which allows for other characters in an LLM answer to surround an otherwise correct substring. For example, if the normalised gold answer string $g^{(q)}$ to question q in the dataset is '1971', then this metric would mark the normalised LLM answer $a^{(q)}$ of 'year 1971' as correct.

Formally, let $G^{(q)} = \{g_1^{(q)}, g_2^{(q)}, \dots, g_n^{(q)}\}$ be the set of normalised gold answer strings for question q of a particular dataset, and let $a^{(q)}$ be the normalised string of the model’s decoded predicted answer. For a Q -question dataset, we define the substring match score as follows:

$$\text{Match}^{(q)} = \begin{cases} 1 & \text{if any } g_i^{(q)} \text{ is a} \\ & \text{substring of } a^{(q)} \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

And the overall accuracy score is computed as the mean of these scores across the entire question dataset, as in Equation 12.

2.2.3 Models

The primary language model used in this study is Meta’s LLaMA-2-7B (Touvron et al., 2023), selected for its public availability and relatively low computational inference cost. LLaMA-2-13B is employed in Section 2.3.7 to align with the implementations of the studies discussed therein.

2.2.4 Other Experimental Details

Temperature We set the temperature to 1 throughout the experiments on the two distributions that are input into the CAD or Additive

CAD formula, as in the original CAD paper (Shi et al., 2024). We leave the study of the effects of varying the temperature of these distributions to future work.

Sampling Method For all results reported in this section, we use *max-probability decoding* to sample from the final CAD or Additive CAD distribution—that is, selecting the token with the highest probability at each step.

Since this deterministic sampling approach is performed on the final modified distribution resulting from the contrastive method under consideration (e.g., CAD, Additive CAD), the whole decoding process is deterministic, and the accuracy figures reported are not subject to sampling variance (in contrast to the stochastic sampling methods described in Section 3).

Again, we leave the study of the effects of stochastic sampling of these distributions to future work.

Prompts For the MemoTrap benchmark, we require no additional prompting other than simple concatenation of the context c and question x to form the input for the with-context LLM, and simply the question x alone as input for the no-context LLM. Hence the prompt templates used are as provided in Table 7 (Appendix A.1) and match those of the original CAD paper.

For the Natural Questions benchmark, however, we wish to simulate a realistic RAG system, by constructing an input prompt with clearly-formatted separation of the contextual information from the question itself. Hence the prompt template for the LLM output distribution with context clearly instructs the LLM to first read the context and then answer the corresponding question, clearly separated by a new line and “Q:” to signify the end of the context and the beginning of the question. The prompt for the LLM output distribution without context is designed to match the wording closely but without any reference to the context. The full template is provided in Table 10 (appendix A.1) and is the prompt used for Natural Questions evaluations unless stated otherwise.

Section 2.3.7 investigates how prompt style affects performance of CAD on the Natural

Questions benchmark.

Stopping Criterion In order to prevent the CAD output from continuing indefinitely, and to save computing costs, well-defined stopping criterion on the generation are required. Unless stated otherwise, the maximum number of generated tokens in the experiments of Section 2 per question is 20.

Additionally, we define stopping symbols so that the CAD generation doesn’t continue unnecessarily. Any ancillary explanation or follow-up after a correctly-generated answer would prevent an exact match with the gold answer. Hence, generation on the MemoTrap benchmark stops on a ' . ' token, and generation on the Natural Questions benchmark stops once a ' . ' or a newline appears in the decoded output.

Handling of infinite float representations

Since the logit values of an LLM output are occasionally a float representation of positive or negative infinity, errors arise when attempting to add or subtract finite-valued logit values from these non-finite values. The solution employed in this paper is to preemptively set positive infinite floats to 1,000 and negative infinite floats to $-1,000$ before performing any floating-point arithmetic. We find that this setting is sufficient in magnitude to avoid any potential unwanted side-effects of the truncation.

2.3 Results and Discussion

The full numerical results corresponding to the graphs in this section are provided in Appendix B.1.

2.3.1 Reproduction of CAD Results on the MemoTrap Benchmark

Figure 1 shows the effect of varying the CAD coefficient α on the EM accuracy of LLaMA-2-7B evaluated on the MemoTrap benchmark. Consistent with the findings of the original CAD paper (Shi et al., 2024), we observe a substantial improvement in accuracy when applying CAD with $\alpha = 1$ compared to standard decoding ($\alpha = 0$).

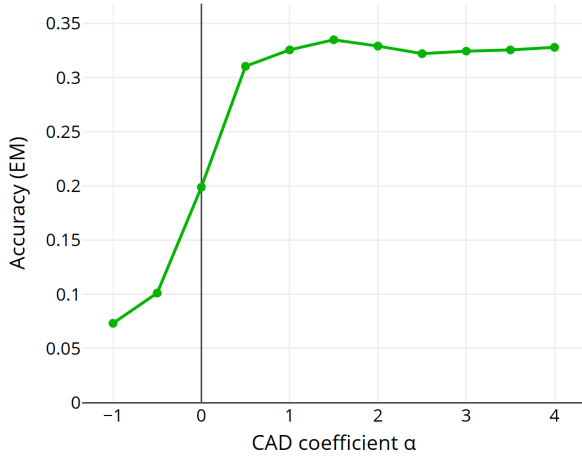


Figure 1: Accuracy of LLaMA-2-7B on the MemoTrap benchmark with varying CAD coefficient α .

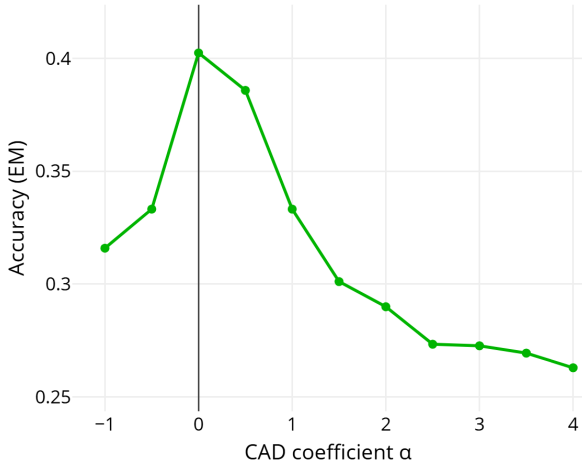


Figure 2: Accuracy of LLaMA-2-7B on the Natural Questions benchmark with varying CAD coefficient α .

While precise EM accuracy values are not directly comparable due to differences in model architecture (the original CAD paper used LLaMA-2-13B but not LLaMA-2-7B), the trend observed here closely mirrors that reported for other models in the original paper. Specifically, accuracy rises sharply from $\alpha = 0$ to $\alpha = 1$, remains stable or improves slightly at $\alpha = 2$, and begins to degrade at higher values (e.g., $\alpha = 4$).

2.3.2 Reproduction of CAD Results on the Natural Questions Benchmark

Figure 2 illustrates the effect of varying the CAD coefficient α on the EM accuracy of LLaMA-2-7B on the Natural Questions benchmark. In contrast to the original CAD pa-

per (Shi et al., 2024), we observe a notable *decrease* in accuracy when CAD is applied with $\alpha = 1$, compared to standard decoding ($\alpha = 0$).

This drop in performance on the Natural Questions benchmark has also been reported by other studies (Yuan et al., 2024; Wang et al., 2024), suggesting that the effectiveness of CAD may be sensitive to task setup. We investigate this discrepancy further in Section 2.3.7. In Section 2.3.9, we even show that under a particular settings configuration—which included use of the prompt template used in the original CAD paper—it is possible to yield results that mirror their findings. However, we also demonstrate that slight modifications to the prompt template can lead to even greater increases in accuracy than those achieved by CAD.

2.3.3 CAD-DoLa

Figure 3 illustrates the effect of varying the CAD coefficient on the EM accuracy of LLaMA-2-7B on the MemoTrap and Natural Questions benchmarks, under various CAD-DoLa configurations.

On the Natural Questions benchmark, we observe that DoLa improves LLM accuracy when no context is provided ($\alpha = -1$), supporting findings from the original DoLa paper that DoLa can amplify factual knowledge already encoded in the model’s layers. However, a much larger accuracy gain is observed when context *is* provided ($\alpha = 0$), while the addition of DoLa in this setting leads to only marginal improvement. This suggests that DoLa is most beneficial when the prompt lacks informative context, and less so when helpful context is available.

Across CAD coefficient values $0.5 \leq \alpha \leq 4$, applying DoLa to the with-context distribution provides a modest improvement in CAD accuracy on the Natural Questions benchmark, partially offsetting the performance drop caused by CAD. In contrast, the same intervention yields minimal effect on CAD performance in the MemoTrap setting.

Applying DoLa to the no-context distribution, however, substantially degrades CAD per-

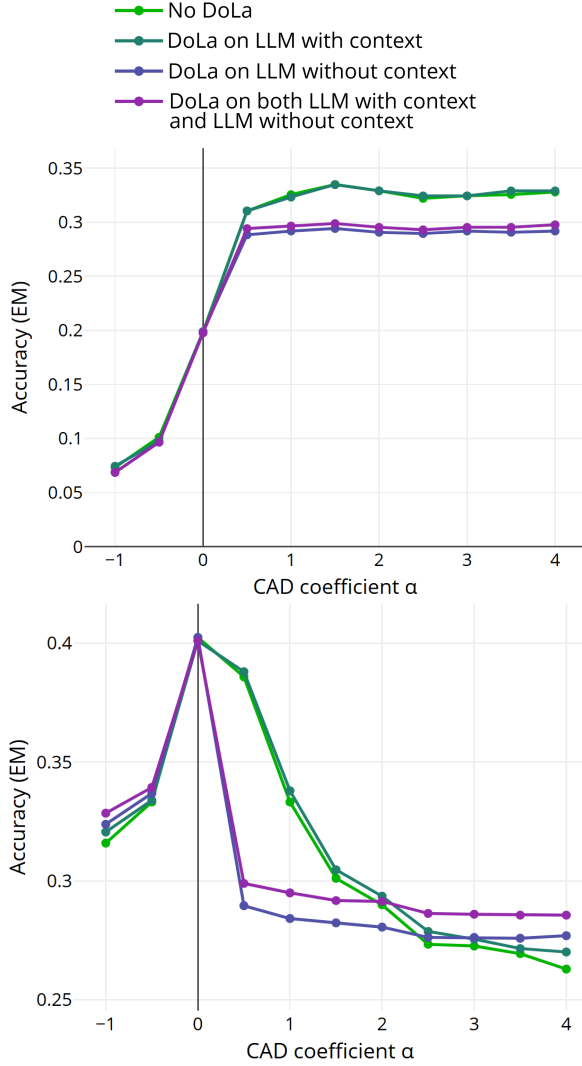


Figure 3: Accuracy of LLaMA-2-7B on the MemoTrap benchmark (top) and Natural Questions benchmark (bottom) for various CAD-DoLa schemes with varying CAD coefficient α .

formance on both benchmarks.

The findings on the Natural Questions benchmark suggest that by improving the truthfulness (through DoLa) of the distribution we are contrasting *against*—the no-context distribution—CAD responses become less truthful. This is an intuitive result, as the contrasting signal formed by the ratio of probability distributions may no longer be as strong for correct tokens that now have high probabilities for both the with-context and no-context distributions—increasing the likelihood that some other nonsensical token probability is amplified.

Similarly, the results suggest that by improv-

ing the truthfulness (through DoLa) of the distribution we are contrasting *in favour of*—the with-context distribution—CAD responses become more truthful. In this case, the contrasting signal formed by the ratio of probability distributions may be even stronger for correct tokens that now have higher probabilities for the with-context distribution—increasing the likelihood that these correct token probabilities are amplified.

The case study presented in Section 2.3.4 visually demonstrates how shifting both of the CAD input distributions through DoLa can undermine the effectiveness of CAD by altering the contrastive signal in other unintended ways.

Without-Context Accuracy It may seem counterintuitive that for $\alpha = -1$, which corresponds to the no-context distribution $P(y_t | \mathbf{x}, y_{<t})$, that the reported accuracies for configurations “no DoLa” and “DoLa applied to the with-context LLM” differ. This discrepancy arises because the implementation consistently applies the Adaptive Plausibility Constraint (APC; see Section 2.1.6) using the with-context distribution, regardless of the α setting. This design choice ensures that only the effect of α is varied in isolation, maintaining experimental consistency across configurations.

2.3.4 Probability Simplex Visualisation

We propose a visualisation technique using the probability simplex that we demonstrate to be particularly valuable when examining the effects of CAD on the base model. This approach offers a useful framework for visualising parts of a high-dimensional probability distribution.

To project a distribution over the full vocabulary (with size $V = 32,000$ for LLaMA-2-7B) into a three-dimensional probability mass function (PMF), the following method is used. First, we select any three tokens of interest. Next, we zero out the probabilities of all other tokens in the distribution. The remaining three probabilities are then renormalised by dividing each by their total sum, ensuring the result is a valid PMF. This reduced distribution

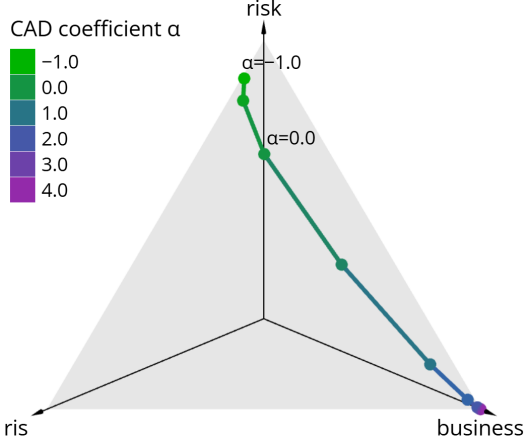


Figure 4: Probability distribution of top three output tokens of LLaMA-2-7B with CAD applied. Context: *Write a quote that ends in the word "business"*. Prompt: *Advisers run no*

can then be visualised as a point on the 2-simplex in three-dimensional space, where coordinates (x, y, z) satisfy the constraints of a PMF: $x + y + z = 1$ and $x, y, z \in [0, 1]$.

In the MemoTrap examples that follow, the three tokens chosen for the visualisation are:

1. The highest token probability of the no-context distribution (often the common ending to the proverb in question).
2. The highest token probability of the $\alpha = 1$ CAD distribution (often the correct ending given the instruction in the context).
3. Some other token of high probability.

Successful CAD An example of a successful trajectory through the probability simplex induced by CAD is shown in Figure 4. As expected, the distribution without context ($\alpha = -1$) assigns high probability to the token 'risk', which completes the well-known proverb in the prompt "Advisers run no...". Even with context ($\alpha = 0$), the distribution *continues* to favour 'risk' over 'business', despite the prompt now *explicitly instructing* that the quote be completed with 'business'. In this case, the model's prior belief overrides the contextual correction. The figure illustrates how increasing α in CAD shifts probability mass toward the contextually appropriate token 'business'. For all $\alpha \geq 1$, the model's

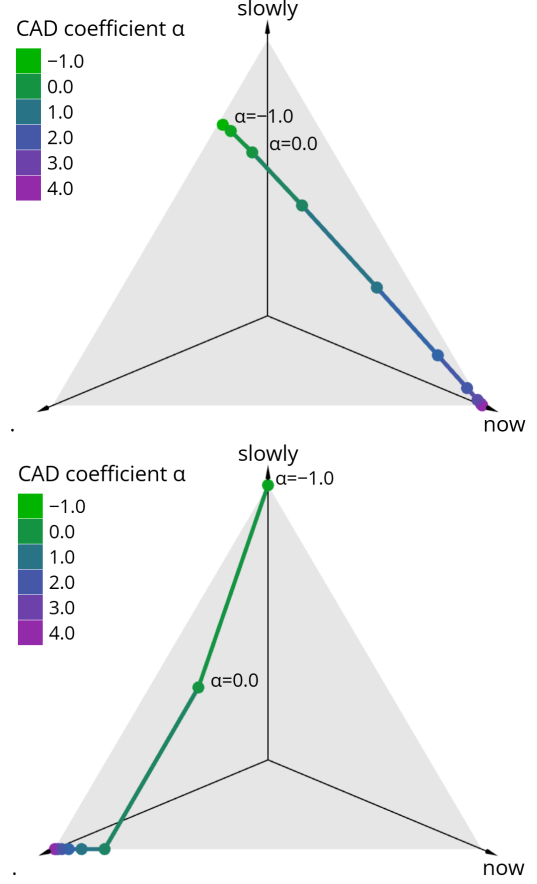


Figure 5: Probability distributions of top three output tokens of LLaMA-2-7B with CAD applied. Top: no DoLa. Bottom: DoLa applied to both LLM with context and LLM without context. Context: *Write a quote that ends in the word "now"*. Prompt: *Make haste*

response is classified as correct.

In a similar manner, the trajectory in the top graph of Figure 5 shows another successful CAD outcome. The distribution without context ($\alpha = -1$) assigns high probability to the token 'slowly', which completes the familiar proverb in the prompt "Make haste...". Even with context ($\alpha = 0$), the model continues to favour 'slowly' over 'now', despite the prompt now explicitly instructing that the quote be completed with 'now'. As in the previous example, increasing α in CAD shifts probability mass toward the contextually appropriate token 'now'.

Unsuccessful CAD-DoLa An example of an unsuccessful trajectory through the probability simplex resulting from CAD-DoLa is shown in the bottom graph of Figure 5. The top graph of

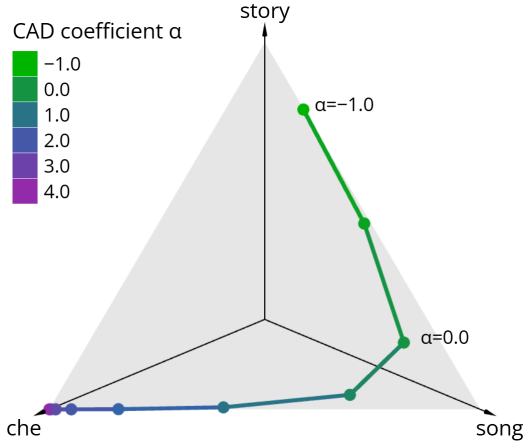


Figure 6: Probability distribution of top three output tokens of LLaMA-2-7B with CAD applied. DoLa applied to both LLM with context and LLM without context.

Context: *Write a quote that ends in the word "song".*
 Prompt: *Applause is the echo of a*

the figure shows that the response would have been marked as correct at $\alpha = 1.0$ if DoLa had not been incorporated. However, by applying DoLa to both the LLMs with and without context, the corresponding output distributions ($\alpha = -1.0$, $\alpha = 0.0$) are shifted to new positions. Specifically, the distribution without context assigns significantly higher probability to the token 'slowly', reflecting how DoLa amplifies knowledge encoded in the model's layers. Meanwhile, the distribution with context is shifted slightly to reduce the probability of 'slowly' in favour of token '.'. As a result, the proportional increase in probability for '.' across the two distributions outweighs the increase for the correct answer 'now'. Consequently, CAD selects '.', and the response is marked as incorrect.

This is an example which corroborates the notable decrease in MemoTrap accuracy observed when DoLa is applied to both the with-context and no-context LLMs.

Motivation for Additive CAD A particularly illustrative failure case for CAD is shown in Figure 6, where amplification of an unlikely token leads to an incorrect response. This pattern, observed across several examples, motivates the Additive CAD method introduced in Section 2.1.2, which modifies the contrastive

mechanism to better preserve alignment with the context.

In this example, the distribution without context ($\alpha = -1$) assigns high probability to the token 'story', which completes the familiar proverb in the prompt "Applause is the echo of a...". With context ($\alpha = 0$), the model *correctly* favours 'song' over 'story', without requiring CAD. However, when CAD is applied, the proportional increase in probability for the incorrect token 'che' between the two distributions outweighs that of the correct answer 'song'. Consequently, CAD selects 'che', and the response is marked as incorrect.

At first glance, this failure of CAD may seem counterintuitive, especially since the with-context distribution appears much closer—at least in Euclidean distance—to the correct answer than the no-context distribution. The issue stems from the fact that even a small absolute increase in probability can become a large *proportional* increase if the no-context value is small. This can cause CAD to amplify the wrong tokens, resulting in a trajectory that moves *away from* the correct answer.

This insight motivates the introduction of *Additive CAD*. We denote the probability distribution at $\alpha = -1$ as the vector \vec{P}_{-1} and the distribution at $\alpha = 0$ as \vec{P}_0 . Intuitively, adding a portion of the difference vector ($\vec{P}_0 - \vec{P}_{-1}$) to \vec{P}_0 shifts the distribution further in the direction informed by context. Additive CAD is thus expressed as:

$$\vec{P}_{\text{AddCAD}} = \vec{P}_{-1} + \gamma(\vec{P}_0 - \vec{P}_{-1}) \quad (15)$$

This formulation is an equivalent vectorised expression of the definition presented in Section 2.1.2.

Applying this new formulation to the example visualised in Figure 6, we can imagine how for $\gamma > 1$, we would move from \vec{P}_0 into the direction formed by drawing a straight line from \vec{P}_{-1} to \vec{P}_0 , resulting in a distribution that is *closer* to the correct token 'song'. This prevents the token 'che'—which has low probability in both input distributions—from being amplified.

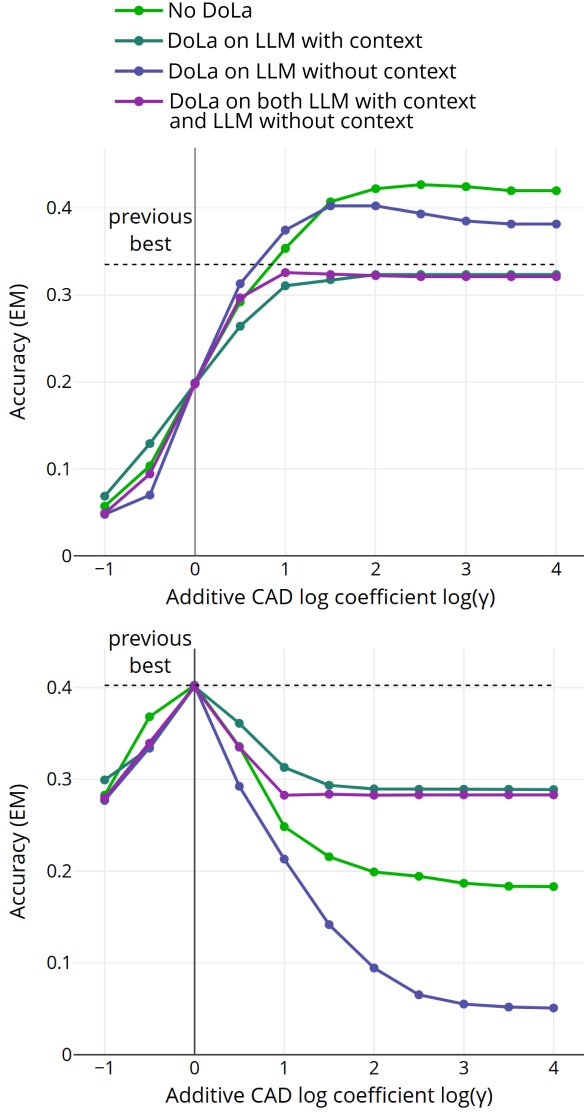


Figure 7: Accuracy of LLaMA-2-7B on the MemoTrap benchmark (top) and Natural Questions benchmark (bottom) for various Additive CAD-DoLa schemes with varying Additive CAD log-coefficient $\log_{10}(\gamma)$.

Note that if γ is sufficiently large, \vec{P}_{AddCAD} may leave the probability simplex and no longer represent a valid probability distribution. To address this, the vector is renormalised—as described in Section 2.1.2—by dividing each element by the sum of all elements, projecting it back onto the simplex.

2.3.5 Additive CAD

The “No DoLa” curves in Figure 7 show the effect of varying the Additive CAD log-coefficient $\log_{10}(\gamma)$ on the EM accuracy of LLaMA-2-7B across the MemoTrap and Nat-

ural Questions benchmarks. Here, $\log \gamma = 0$ corresponds to standard with-context decoding without any additive adjustment.

On the MemoTrap benchmark (top graph), Additive CAD yields a substantial improvement over regular CAD, achieving a peak accuracy of 42.7% at $\log \gamma = 2.5$, compared to the prior best of 33.5% under standard CAD with $\alpha = 1.5$. This demonstrates the strong potential of Additive CAD as a more effective approach for reinforcing critical context and countering harmful model priors—whilst avoiding amplification of nonsensical tokens.

However, consistent with findings for standard CAD, applying Additive CAD to the Natural Questions benchmark results in a clear *drop* in performance. In fact, Additive CAD proves even more detrimental than regular CAD in this setting: across the range $1 \leq \log \gamma \leq 4$, no configuration outperforms the accuracy of any CAD setting with $1 \leq \alpha \leq 4$. This highlights a lack of robustness and poor performance for Additive CAD in with-context QA settings.

The differing results on MemoTrap and Natural Questions are rooted in the nature of the benchmarks. In MemoTrap, the context is *essential*—without it, the model defaults to completing a familiar quote incorrectly. This stark misalignment between the prior (no-context) and the context-informed distributions allows Additive CAD to be highly effective: the correct token often receives a substantial probability boost only when context is introduced.

In comparison, context in Natural Questions is often helpful but not strictly necessary. The distributions with and without context are typically more similar, meaning their difference vector may not point meaningfully toward the correct answer. As a result, Additive CAD may shift probability mass away from the right token, leading to degraded performance.

2.3.6 Additive CAD-DoLa

Figure 7 illustrates the effect of varying the Additive CAD log-coefficient $\log \gamma$ on the EM accuracy of LLaMA-2-7B on the MemoTrap and Natural Questions benchmarks, under sev-

Table 2: Reported EM accuracy of LLaMA-2-13B on the Natural Questions benchmark, across different publications. Note that decoding and evaluation settings vary across each paper.

Paper	Regular ($\alpha = 0$)	CAD ($\alpha = 1$)
CAD (Shi et al., 2024)	0.223	0.336
COIECD (Yuan et al., 2024)	0.731	0.675
AdaCAD (Wang et al., 2024)	0.443	0.379

eral Additive CAD-DoLa configurations.

As seen in the top graph, applying DoLa to either distribution does not improve the peak accuracy of Additive CAD on MemoTrap. Introducing DoLa to the LLM without context provides a modest boost for $\log \gamma \in \{0.5, 1.0\}$, but the resulting curve saturates earlier and drops more sharply at higher values of $\log \gamma$. In contrast, applying DoLa to the LLM with context worsens performance across the range—in contrast to regular CAD-DoLa.

For the Natural Questions benchmark, however, DoLa partially mitigates the accuracy degradation introduced by Additive CAD. Consistent with findings for regular CAD, the most effective configuration is to apply DoLa to the with-context LLM. Applying DoLa to the no-context LLM alone is the most harmful—leading to an eightfold reduction in accuracy at higher values of $\log \gamma$.

This pattern is intuitive: as shown in Section 2.3.3, DoLa improves the factual accuracy of the LLM without context. By enhancing the factuality of the distribution we are trying to contrast *against*, Additive CAD is more likely to shift probability mass in an unhelpful direction—away from the contextually appropriate answer.

2.3.7 Addressing Inconsistencies Across Papers

As demonstrated in Section 2.3.2, our experiments show a decrease in accuracy when applying CAD on the Natural Questions benchmark

compared to standard with-context decoding. This result contradicts the findings reported in the original CAD paper (Shi et al., 2024), but aligns with subsequent studies: ‘COIECD’ (Yuan et al., 2024), and ‘AdaCAD’ (Wang et al., 2024). Table 2 summarises the reported EM accuracies on the Natural Questions benchmark using LLaMA-2-13B from these three works.

Despite consistencies in LLM and evaluation metrics used, both the baseline (regular decoding) and CAD-modified EM scores vary substantially between papers, indicating major inconsistencies in experimental conditions.

The coded evaluation scripts employed by each paper⁵ were analysed in detail. Two primary factors that likely contribute to these discrepancies are as follows:

- Dataset variants:** The specific version of the Natural Questions dataset used in the original CAD paper is not disclosed. COIECD evaluates on a 4,289-example subset containing both short (1-2 word) and long (≥ 3 word) answers. AdaCAD uses a different 3,230-example subset with a similar composition. Our implementation employs a filtered 2,773-example variant that includes only short-form answers. These differences in dataset construction impact the difficulty of the questions and, consequently, reported accuracies.
- Prompt templates:** Each study uses distinct prompting styles (see Appendix A.1). Notably, the original CAD paper simply concatenates the context and question without any additional prompting, whereas COIECD, AdaCAD, and our work employ prompts that more clearly delineate the input components and include an explicit task instruction. As we demonstrate in Section 2.3.8, the prompting scheme alone can significantly affect the relative success of CAD.

⁵CAD: <https://github.com/xhan77/context-aware-decoding>
COIECD: <https://github.com/Stacy027/COIECD>;
AdaCAD: <https://github.com/HanNight/AdaCAD>

Additional implementation-level differences—particularly in decoding constraints—may also account for discrepancies in reported EM scores. For instance, the AdaCAD implementation allows up to 100 generated tokens per response, whereas COIECD constrains output length to exactly match that of the gold answer. This difference can have a substantial impact on EM accuracy: in AdaCAD, longer generations would increase the likelihood of producing additional tokens beyond the correct answer, thereby reducing the chances of achieving an exact match, even if the core answer is correct. This likely contributes to the consistently higher EM scores reported by COIECD compared to AdaCAD.

The lack of consistency in reported accuracies highlights the sensitivity of benchmark results to changes in experimental conditions, and how different conclusions may be drawn depending on these conditions. Section 2.3.8 and 2.3.9 show how small changes to experimental conditions can cause entirely different trends to be observed with varying CAD coefficient.

2.3.8 The Effect of Prompt on Natural Questions Performance

Figure 8 compares CAD (top) and Additive CAD (bottom) performance across four prompting schemes—those used in this work, CAD (Shi et al., 2024), COIECD (Yuan et al., 2024), and AdaCAD (Wang et al., 2024). These prompts are provided in Appendix A.1 in Tables 10, 7, 8 and 9, respectively. All other settings, including dataset and decoding configuration, are held constant.

The results indicate the following:

- Our prompt achieves the highest performance under regular decoding without context.
- The AdaCAD prompt performs best when context is provided.
- The original CAD prompt consistently underperforms across all configurations.
- For none of the four prompts does either CAD nor Additive CAD yield an improve-

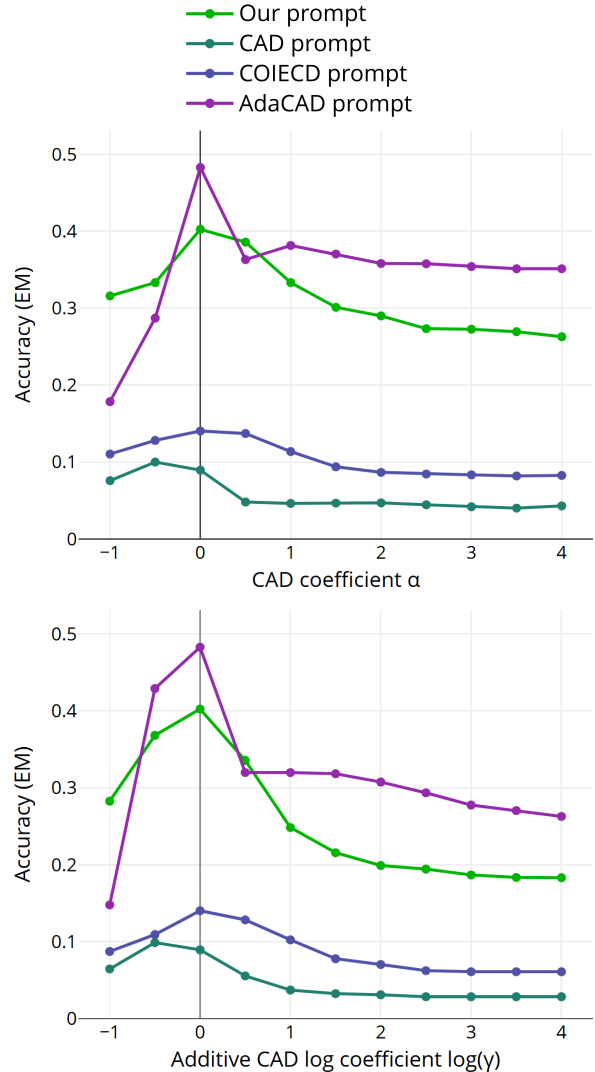


Figure 8: Accuracy of LLaMA-2-7B on the Natural Questions benchmark for regular CAD (top) and Additive CAD (bottom), with prompt styles from different papers: this paper, CAD (Shi et al., 2024), COIECD (Yuan et al., 2024), and AdaCAD (Wang et al., 2024). Each prompt template is provided in Appendix A.1.

ment in EM accuracy over standard decoding.

These results suggest that previously reported CAD gains with use of the CAD prompting scheme on Natural Questions are highly contingent on prompt format and may not generalise across implementations.

While these results are not intended to precisely replicate each original paper’s findings, they offer a controlled comparison under standardised conditions. They underscore the extent to which prompting—often treated as a

Table 3: Substring match accuracy of LLaMA-2-13B on the Natural Questions benchmark, by input prompt style (each of which is provided in Appendix A.1), with constant decoding and evaluation settings.

Input prompt style (paper)	Regular ($\alpha = 0$)	CAD ($\alpha = 1$)
Ours	0.771	0.762
CAD (Shi et al., 2024)	0.341	0.417
COIECD (Yuan et al., 2024)	0.780	0.687
AdaCAD (Wang et al., 2024)	0.785	0.745

secondary consideration—can dictate the effectiveness of contrastive decoding methods.

2.3.9 Sensitivity to Decoding and Evaluation Settings

Throughout our experiments, we maintain fixed decoding and evaluation parameters to isolate the effects of CAD and Additive CAD. Specifically, we have used exact match (EM) as the evaluation metric, LLaMA-2-7B as the model, a generation limit of 20 tokens, and stopping conditions based on the characters ' . ' and newline.

In this final analysis, we relax these constraints to explore whether CAD can produce

accuracy gains under a different set of conditions. We adjust the setup as follows:

We use the more lenient substring match as the accuracy metric instead of EM. Next, the underlying model is changed to LLaMA-2-13B, and we limit text generation to a maximum of 15 tokens, with decoding terminated at the end-of-sequence (EOS) token otherwise.

Table 3 presents the results under these revised conditions. With the original CAD prompt, we now observe a notable improvement in accuracy for $\alpha = 1$ relative to regular decoding, echoing the trend reported by the original CAD paper. However, even in this configuration, the performance remains below that achieved with any of the other three prompting styles under regular decoding, *obfuscating the need for CAD*. These findings reinforce the conclusion that CAD—provided that a prompting scheme other than simple concatenation is used—decreases performance on the Natural Questions benchmark.

The findings also reinforce the conclusion that performance of CAD is highly sensitive to evaluation and decoding settings. Minor changes in generation length, token constraints, or evaluation criteria can materially alter the conclusions drawn—raising important questions about reproducibility and the generalisability of CAD across diverse tasks and settings.

3 Wider Exploration of Idea Space with Doubt Injection

This section explores an alternative research direction focused on recently-developed reasoning models—a distinct category from those investigated so far in this report. These models achieve superior performance across a range of tasks as a result of a generation process that produces a lengthy Chain-of-Thought (CoT) reasoning process before concluding with a final answer. We investigate the performance of these models in situations where informative context isn’t available, and instead the LLM must rely on internal reasoning to produce final answers.

3.1 Method

This report introduces *Doubt Injection*, a method for subtly influencing the generative trajectory of the CoT of large language models by introducing uncertainty cues at key points during text generation.

The method operates within the standard autoregressive generation process (as described in Section 1.2.1), but selectively intervenes at natural discourse boundaries—such as paragraph breaks—by probabilistically inserting linguistic markers of hesitation (e.g., the phrase “But”). These inserted tokens are intended to guide the model toward exploring alternative ideas, without modifying the original user prompt.

Algorithm 1 outlines an example implementation. Tokens y_t are generated by sampling from the LLM output distribution $P(y_t \mid \cdot)$ given the input prompt x and generated sequence so far y . The algorithm checks whether the sampled token y_t corresponds to a string from the set \mathcal{S} —a non-exhaustive set of string patterns typically indicating paragraph breaks (i.e., sequences ending in two newline characters). Although this set does not capture all possible paragraph transitions (such as consecutive newline tokens), it was found to be effective in practice at detecting new paragraphs for the models evaluated in this work.

Algorithm 1 Doubt Injection of string “But” during LLM text generation.

Require: Input prompt x , maximum generation steps N , injection probability p

Ensure: Generated sequence y

Initialize $y \leftarrow []$

Let $\mathcal{S} \leftarrow \{“. \backslash n \backslash n”, “. \backslash n \backslash n”, “ \backslash n \backslash n”, “ \backslash n \backslash n”\}$

for $t = 1$ to N :

 Sample $y_t \sim P(y_t \mid x, y)$

 Append y_t to y

 Let $s \leftarrow \text{decode}(y_t)$

if $s \in \mathcal{S}$:

 Sample $u \sim \mathcal{U}(0, 1)$

if $u < p$:

 Append $\text{tokenise}(\text{“But”})$ to y

if y_t is the end-of-sequence token :

break

return y

Whenever a new paragraph is detected, the algorithm appends—with probability p —a tokenised representation of the chosen injection string (in this case “But”) to the sequence of generated tokens so far, y . Then, the algorithm repeats by sampling the next token in the sequence.

Some minor implementation details are omitted from Algorithm 1 for brevity. In particular, our implementation in practice uses a boolean variable, `in_cot`, to indicate whether generation is still in the Chain-of-Thought reasoning phase. For the DeepSeek-R1 models used in this study, the thinking process is confined by the model to be within the tokens `<think>` and `</think>`. Once the `</think>` token is detected in the generated output, the `in_cot` flag is set to `false` to indicate that generation is no longer in the CoT, and subsequent tokens are generated without any further application of Doubt Injection. This design confines the intervention to the internal reasoning process, ensuring that the final answer maintains natural flow.

An example of LLM generation with Doubt Injection applied is provided in Appendix A.3, where each injection is clearly shown.

3.2 Experimental Setup

The evaluation scripts used in this section are available on GitHub⁶.

3.2.1 Datasets

Reasoning and social intelligence SimpleBench (Philip and Hemang, 2024) is a multiple-choice benchmark test (6 choices per question) on which humans with high school knowledge consistently outperform state-of-the-art LLMs. The questions are designed to cover ‘spatio-temporal reasoning’, ‘social intelligence’, and ‘linguistic adversarial robustness (or trick questions)’. An example of a question from this dataset is provided in Table 4.

Only a small portion (10 questions) are made available to the public. During initial pilot experiments, a small portion of these questions were found to never produce a correct answer and hence to save limited computational resources, the subset of questions whose results are shown (referred to herein as ‘the subset of SimpleBench questions’) are questions 1, 2, 3, 8, 9 and 10 of the publicly available questions.

Mathematical reasoning The American Invitational Mathematics Examination (AIME) is a prestigious, invite-only mathematics competition for high-performing high-school students. The AIME 2024 benchmark (MAA, 2024) contains 30 challenging mathematics problems, which each typically require around 10 steps each to solve and have an answer which is an integer in the range 0 to 999. An example of a question from this dataset is provided in Table 4.

This dataset holds the major advantage over other popular mathematics benchmarks in that performance on it has not yet become saturated. For instance, given that most new models now readily achieve over 90% accuracy on the frequently-used MATH-500 benchmark, differences in model performance are difficult to ascertain. Due to the increased difficulty and multi-step process required to

⁶See <https://github.com/stephen1cowley/deepseek-experiments>.

Table 4: Example question and correct (gold) answer from the SimpleBench (Philip and Hemang, 2024) (question 9) and AIME 2024 (MAA, 2024) datasets.

SimpleBench
Question: Agatha makes a stack of 5 cold, fresh single-slice ham sandwiches (with no sauces or condiments) in Room A, then immediately uses duct tape to stick the top surface of the uppermost sandwich to the bottom of her walking stick. She then walks to Room B, with her walking stick, so how many whole sandwiches are there now, in each room? A. 4 whole sandwiches in room A, 0 whole sandwiches in Room B B. no sandwiches anywhere C. 4 whole sandwiches in room B, 1 whole sandwich in Room A D. All 5 whole sandwiches in Room B E. 4 whole sandwiches in Room B, 1 whole sandwiches in room A F. All 5 whole sandwiches in Room A Gold answer: A
AIME 2024
Question: Find the largest possible real part of $(75 + 117i)z + \frac{96 + 144i}{z}$ where z is a complex number with $ z = 4$. Gold answer: 540

solve the problems in the AIME benchmarks however, state-of-the-art models do not yet consistently achieve accuracies over 90% on the dataset.

Due to the low number of questions in both benchmarks discussed above, multiple runs were made through each dataset, with the final accuracy computed as specified in Section 3.2.2. Since the implementations in this section—for optimal performance—use stochastic sampling of tokens with non-zero

temperatures (unlike the experiments in Section 2), increasing the number of runs increases the precision of the results and reduces the size of the error bars (calculated through the method outlined in Section 3.2.5).

3.2.2 Metrics

Accuracy Calculation To account for the variance in model performance due to stochastic decoding, we report the average pass@1 score across multiple runs. Specifically, for each run, the model is evaluated on the full benchmark dataset by generating a single response per question, and the pass@1 metric is computed as the proportion of questions for which the model’s first response is correct. Given Q questions in the dataset and by letting $c_q \in \{0, 1\}$ indicate whether the answer to question q is correct (1 if correct, 0 otherwise), the pass@1 score for a single run is calculated:

$$\text{Pass@1} = \frac{1}{Q} \sum_{q=1}^Q c_q \quad (16)$$

The average pass@1 is then defined as the mean of these pass@1 scores across all n independent runs:

$$\text{AvgPass@1} = \frac{1}{n} \sum_{i=1}^n \text{Pass@1}_i \quad (17)$$

This is mathematically equivalent to the overall proportion of correct answers across all runs combined, and is herein referred to as “accuracy”.

3.2.3 Models

The experiments conducted on Doubt Injection are required to be carried out with models that allow for token-by-token control. This would be highly inefficient with the use of closed-source models via API (due to network latency and lack of caching with single-token requests). Moreover, many closed-source reasoning model providers such as OpenAI’s o1 do not provide access to the full Chain-of-Thought, where Doubt Injection is intended to operate.

The models chosen are two distilled ver-

sions of DeepSeek-R1 (Guo et al., 2025; DeepSeek-AI, 2025): DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Qwen-1.5B. These models satisfy the requirement of being open source models that produce Chains-of-Thought. Specifically, the models were trained to contain a reasoning process within `<think>` `</think>` tags and final answer within `<answer>` `</answer>` tags.

The distilled (32 and 1.5 billion parameter) models are selected over the original 671 billion parameter model because of their lower inference costs as a result of fewer required floating point operations during inference. In addition, the distilled models perform strongly on benchmarks compared to alternative models of similar size (Guo et al., 2025).

3.2.4 Other Experimental Details

Nucleus Sampling (top-p) Setting In alignment with the evaluation setup of the original DeepSeek-R1 paper (Guo et al., 2025), we set the top-p (nucleus sampling) value to 0.95 in our experiments. Introduced by Holtzman et al. (2019), nucleus sampling selects tokens from the smallest possible subset of the vocabulary whose cumulative probability exceeds a threshold p . Given a model’s predictive distribution $P(w_t \mid \mathbf{w}_{<t})$ over the vocabulary \mathcal{V} , top-p sampling defines $\mathcal{V}^{(p)} \subset \mathcal{V}$ as the smallest set such that:

$$\sum_{w \in \mathcal{V}^{(p)}} P(w_t = w \mid \mathbf{w}_{<t}) \geq p \quad (18)$$

and samples the next token from $\mathcal{V}^{(p)}$. This ensures that the sampled tokens remain sufficiently relevant.

Generation Limits To prevent indefinite generation, we impose explicit token limits. Once the number of generated tokens exceeds the specified threshold, the process is halted, and the response is marked as incorrect. The choice of token limit reflects a trade-off between maximising the number of correct responses captured and minimising unnecessary computational cost. Accordingly, we use a limit of 20,000 tokens for the motivating re-

sults in Section 3.3.1, reduce the limit to 10,000 tokens for subsequent experiments, and explore the effects of increasing the limit in Section 3.3.6.

Prompting The prompt template used on SimpleBench questions is provided in Table 12 of Appendix A.2. The LLM is instructed to contain its reasoning within `<think>` and `</think>` tags and its final answer within `<answer>` and `</answer>` tags, in line with the training prompts of the original DeepSeek paper (Guo et al., 2025). For evaluation, the string within the final answer tags is extracted, stripped of any leading and trailing whitespace and compared to the correct answer which is one of A, B, C, D, E, or F.

The prompt used on AIME 2024 questions is provided in Table 13 of Appendix A.2. The LLM is instructed to reason step-by-step and put the final answer within `\boxed{}`, following the prompting recommendations of DeepSeek-AI (2025). The enclosed string is extracted, whitespace-trimmed, and converted to an integer using Python’s `int` function to accommodate variations such as `073` or `73.0` that should be treated as correct. If `int` raises an error, indicating a non-integer response, the answer is marked as incorrect.

Following further recommendations by DeepSeek-AI (2025), we prepend “`<think>\n`” to the input to encourage the model to engage in the intended reasoning process, rather than skipping directly to an answer. As detailed in Appendix A.2, this technique is applied exclusively to experiments involving the AIME 2024 benchmark, based on findings that the model often omits reasoning without such guidance.

An additional prompting consideration concerns the formatting of the Doubt Injection string. Early pilot experiments used injection strings that ended with a space (e.g., “But wait, let me think again. ”), which frequently resulted in awkward outputs, such as:

But wait, let me think
again. maybe I missed something.

The unintended double space and lowercase continuation stem from the model’s tokenisation behavior, which typically does not include a trailing space at the end of words. Consequently, all injection strings used in our experiments omit the trailing space, resulting in more natural output (see Appendix A.3).

3.2.5 Calculation of Confidence Intervals

Many evaluation studies of LLMs on question-answering datasets omit error bars in performance graphs. This omission can be misleading, as it fails to capture the inherent uncertainty in reported accuracy scores. For example, consider evaluating two LLMs at non-zero temperature (non-deterministic generation) on a 200-question dataset, with observed accuracies of 72.5% for LLM A and 73.0% for LLM B. A naïve interpretation might assert that LLM B is more accurate. However, a more rigorous statistical analysis reveals that there is only a 54% chance that LLM B is truly superior on this dataset—barely better than random guessing. Such marginal differences are often overinterpreted as definitive, despite being statistically inconclusive.

This result arises when we model the accuracy of an LLM as a probability distribution over possible values, conditioned on observed outcomes. Let the true (but unknown) accuracy of a model on a particular dataset be denoted by $a \in [0, 1]$, representing the probability that the model produces a correct response to a randomly chosen question in the dataset.

If we now produce a set \mathcal{D} of n independent observations— N correct answers and M incorrect answers on the dataset (where $n = N + M$), then the probability of making this exact set of observations is:

$$P(\mathcal{D}|a) = a^N (1 - a)^M \quad (19)$$

Using Bayes’ theorem, the posterior distribution of the accuracy given the observed data is:

$$p(a|\mathcal{D}) \propto P(\mathcal{D}|a)p(a) \quad (20)$$

Assuming a uniform prior over accuracy, $p(a) = 1$ for $a \in [0, 1]$ (equivalent to a

Beta(1, 1) prior), the posterior simplifies to:

$$p(a|\mathcal{D}) \propto a^N (1-a)^M \quad (21)$$

This posterior is the Beta distribution with parameters $\alpha = N + 1$ and $\beta = M + 1$:

$$p(a|\mathcal{D}) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} a^{\alpha-1} (1-a)^{\beta-1} \quad (22)$$

where Γ is the gamma function. The mode of this distribution—the most probable accuracy under the posterior—is the maximum a posteriori (MAP) estimate. For a uniform prior, this is also the maximum likelihood estimate (MLE) and is given by:

$$\hat{a} = \frac{N}{N + M} \quad (23)$$

This is the observed accuracy over all questions of all trials and corresponds to the average pass@1 accuracy metric defined in Section 3.2.2.

To quantify uncertainty, we compute a 95% Bayesian credible interval by finding the values of a for which the cumulative distribution function (CDF) of the Beta distribution reaches 2.5% and 97.5%. These form the lower and upper bounds of the interval, respectively, as shown in the example in Figure 9. Although technically a credible interval, we refer to it as a “confidence interval” in keeping with common ML terminology.

Beyond computing confidence intervals, we can also estimate the probability that one model is genuinely more accurate than another. We approximate the probability that B is more accurate than A via simple Monte Carlo sampling: the two Beta distributions are sampled a large number of times, and we compute the proportion of times the accuracy of B is higher than the accuracy of A for each pair of samples.

Illustrative Example The top plot in Figure 10 compares two hypothetical LLMs evaluated on 30 questions, with scores of 19 and 20 respectively—accuracies of 63.3% and 66.7%. Despite this apparent difference, the posterior distributions overlap substantially, and Monte

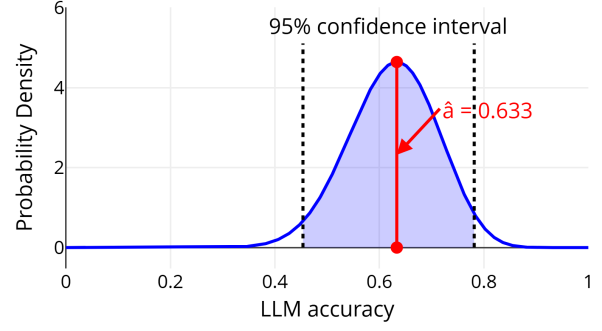


Figure 9: Posterior probability density function of the true accuracy a for a hypothetical LLM with score of 19/30. 95% confidence interval and distribution mode \hat{a} are shown.

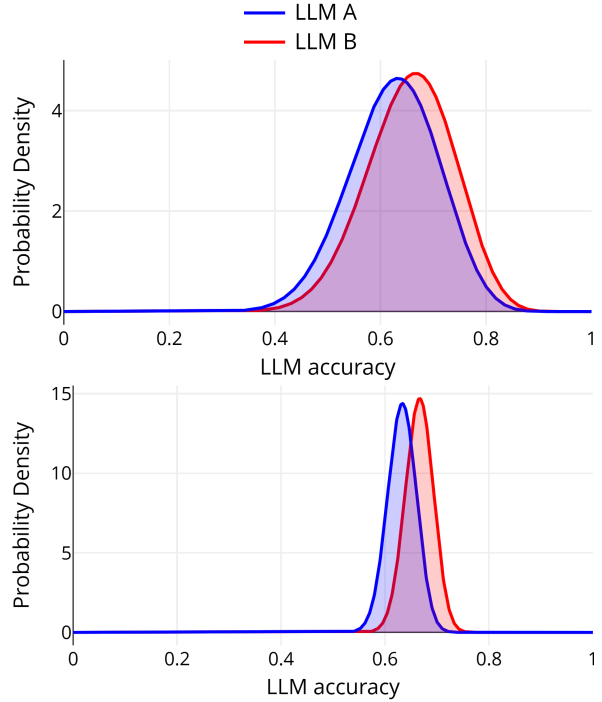


Figure 10: Posterior probability density functions of the true accuracy a for two hypothetical LLMs: LLM A and LLM B. Top: scores of 19/30 vs. 20/30. Bottom: 190/300 vs. 200/300.

Carlo sampling reveals only a 60% chance that LLM B is truly more accurate than LLM A.

The bottom plot shows the same accuracy difference, but based on 10 times more data (300 questions). Here, the posterior distributions are narrower due to reduced sampling variance, and the probability that LLM B is more accurate increases to 80%. This illustrates how statistical confidence in relative performance improves with larger sample sizes.

From the central limit theorem for a large number of samples taken from a binomial dis-

Table 5: Prompt and typical answers (the portion of the LLM response enclosed within <answer> and </answer> tags) of DeepSeek-R1-Distill-Qwen-32B for the altered river crossing problem.

Prompt:

Solve the following puzzle: A man and a goat are on the left side of a river. There is a wolf and a cabbage on the right side of the river. The man has a boat. The boat can carry only the farmer and a single item. How can the farmer get the goat to the right side of the river?

Example correct answer:

The farmer takes the goat across the river to the right side.

Example incorrect answer:

1. The farmer takes the goat across the river to the right side.
 2. The farmer returns alone to the left side.
 3. The farmer takes the wolf across to the right side.
 4. The farmer brings the goat back to the left side.
 5. The farmer takes the cabbage to the right side.
 6. The farmer returns alone to the left side.
 7. The farmer takes the goat to the right side.
- All three—the goat, wolf, and cabbage—are safely across the river.

tribution, $n \rightarrow \infty$, the posterior probability distribution approaches that of a Gaussian with a standard deviation (distribution width) that scales with $\sim \frac{1}{\sqrt{n}}$.

3.3 Results and Discussion

The numerical benchmark results corresponding to the graphs in this section are provided in Appendix B.2.

3.3.1 Motivating Results (the Altered River Crossing Problem)

Setup To motivate Doubt Injection as a potentially desirable method, we conduct a brief case study using a carefully constructed question designed to break from typical LLM train-

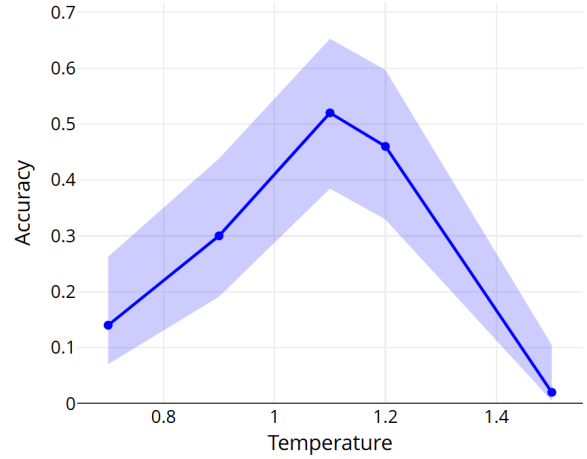


Figure 11: Accuracy of DeepSeek-R1-Distill-Qwen-32B on the altered river crossing puzzle, for different temperatures. 50 runs for each temperature. Shaded regions are 95% confidence intervals.

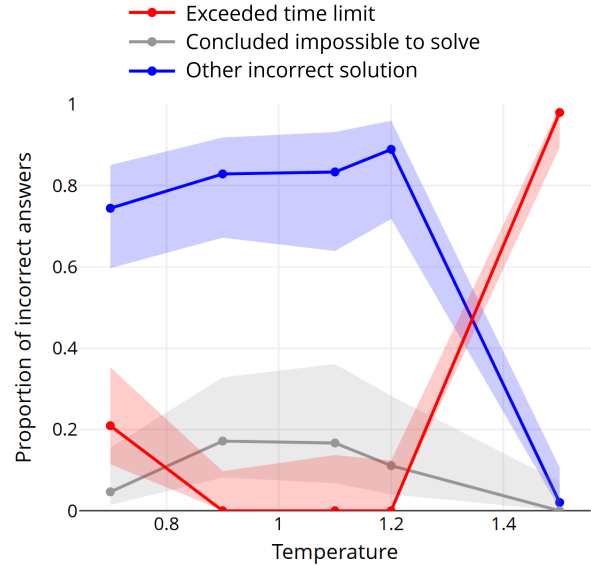


Figure 12: Proportional makeup of incorrect responses of DeepSeek-R1-Distill-Qwen-32B on the altered river crossing puzzle, for different temperatures. 50 runs for each temperature. Shaded regions are 95% confidence intervals.

ing data. This tests the model’s ability to reason through unusual problems without relying on prior exposure to similar examples. The question, along with example responses labeled as correct or incorrect, is provided in Table 5. It is based on the classic river-crossing puzzle, where solving the original version typically requires a seven-step process involving strategic item transport to avoid undesirable interactions between various entities.

However, in our modified version, while the

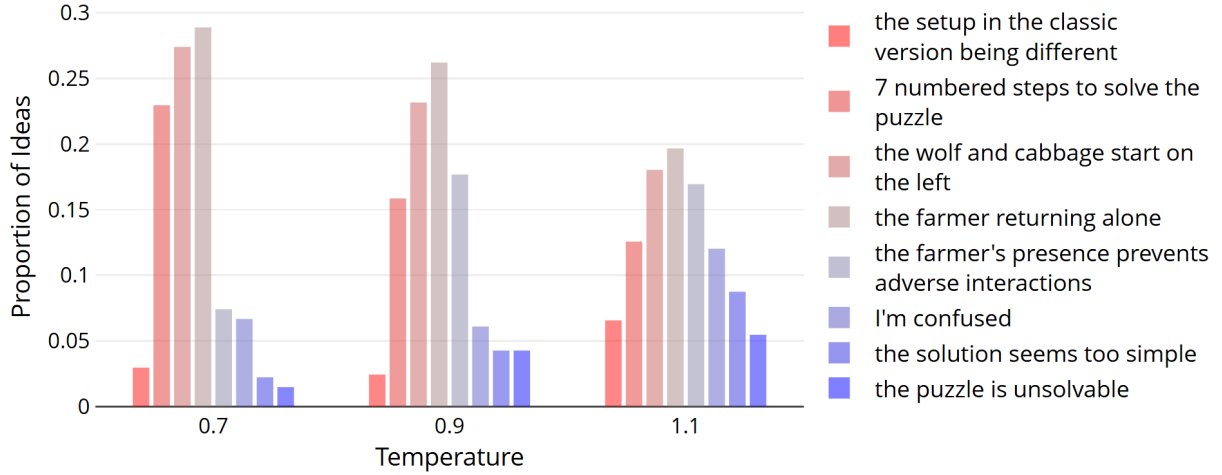


Figure 13: The proportional makeup of ‘ideas’ explored in the Chain-of-Thought of DeepSeek-R1-Distill-Qwen-32B in answering the altered river crossing puzzle, across different temperatures.

wording mimics the style of the original, the problem has been made trivially solvable in a single step. This subtle subversion serves as a diagnostic tool to reveal whether the model defaults to the memorised multi-step procedure—a major hallucination—or whether it can adapt its reasoning to the actual constraints of the problem.

The full input prompt is designed to elicit Chain-of-Thought reasoning with clear formatting, based on the prompt style used in training Deepseek-R1-Zero (Guo et al., 2025). The prompt is provided in the appendix in Table 11.

Results We manually labeled 50 model responses at various temperature settings, due to the variability in the wording of plausible correct answers. Accuracy results are shown in Figure 11, and the distribution of error types among incorrect responses is shown in Figure 12.

At a temperature of $T = 0.7$, accuracy is low at just 14%. The vast majority of incorrect responses conclude with the canonical 7-step solution typical of the classical river-crossing problem—demonstrating that the LLM is heavily influenced by memorised training data. This is consistent with the hypothesis that the model has encountered the standard problem but has not been exposed to this altered, trivially solvable version.

Approximately 21% of the incorrect responses at $T = 0.7$ are due to exceeding the

generation time limit (12 minutes), with many of these exhibiting an apparent infinite loop: the model repeatedly generates the same sentence. This failure mode aligns with behaviors documented by DeepSeek-AI (2025).

Accuracy peaks at 52% at $T = 1.1$. While many incorrect answers at this setting still default to the classical 7-step solution, none fall into looping behavior, suggesting that this intermediate temperature setting helps the model escape repetitive generation traps.

Interestingly, these intermediate temperatures ($T = 0.9$ and $T = 1.1$) also yield a greater proportion of incorrect answers that conclude the puzzle is impossible, compared to $T = 0.7$ (see Figure 12). This indicates that the model begins to explore more diverse reasoning paths at these temperatures, even if they do not always lead to the correct conclusion.

However, at $T = 1.5$, accuracy drops sharply. At this level, outputs frequently mix languages, become nonsensical, and consistently hit the time limit. As shown in Figure 12, nearly all errors at this temperature are due to these runaway generations.

Exploration of Idea Space An evaluator LLM (Qwen2.5-32B at a zero temperature setting) was used to analyse the Chain-of-Thoughts of the altered river crossing problem. After splitting up the Chain-of-Thought into overlapping segments, we query the evaluator LLM on each segment as to whether a particu-

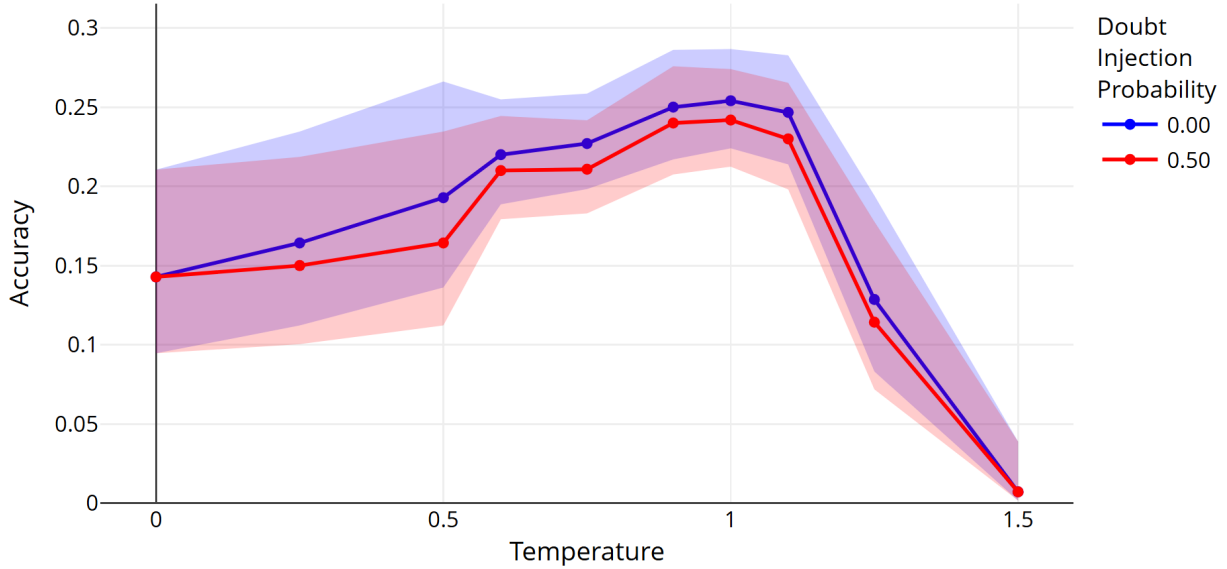


Figure 14: Accuracy of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions, with and without Doubt Injection of “But wait, let me think again.” Shaded regions are 95% confidence intervals. Varying confidence interval widths are as a result of varying number of runs at each temperature.

lar idea is mentioned in the text.

This results in a true/false for each idea, for each response. This is then averaged over the 50 responses, before being normalised to form a ‘distribution over ideas’ at each temperature, shown in Figure 13.

We can see from this figure that the vast majority of ideas explored in the Chain-of-Thought at lower temperatures are concentrated in a small number of (mostly incorrect) ideas. However, as the temperature rises, the distribution flattens, and the variation in ideas explored becomes richer.

This result is expected from the fact that higher temperature results in higher randomness in individual tokens. If a wider variety of tokens are explored, it follows that a wider variety of ‘ideas’ would result in the output as a whole.

Relation to accuracy The idea of “the setup in the classic version being different” (an important realisation in correctly solving the puzzle) shows an increase in prevalence as the temperature rises. This coincides with the increase in accuracy seen over this temperature range in Figure 11. This suggests that a higher temperature allows this and other under-explored correct ideas to be accessed more often by the

Chain-of-Thought, resulting in a higher accuracy and therefore reduced hallucination in the final answer.

Statements of doubt The mention of “I’m confused” also increases in prevalence as the temperature rises. This provides a motivation for Doubt Injection. If the Chain-of-Thought can be made to explore a *richer set of ideas*, perhaps it will be more likely to mention the correct idea, explore it further, and ultimately conclude with it. The following sections test whether the injection of a statement encouraging a sense of doubt, for instance “But wait, let me think again.” can cause a wider variety of ideas to be explored, and ultimately increase the accuracy of LLMs.

3.3.2 Doubt Injection: Effect of Temperature

We examine the effect of sampling temperature on model accuracy, comparing standard generation ($p = 0$) with generation using Doubt Injection—specifically, the phrase “But wait, let me think again.” inserted with probability $p = 0.5$ at each new paragraph. Results on the subset of SimpleBench questions using the DeepSeek-R1-Distill-Qwen-32B model are shown in Figure 14.

Across both configurations, accuracy in-

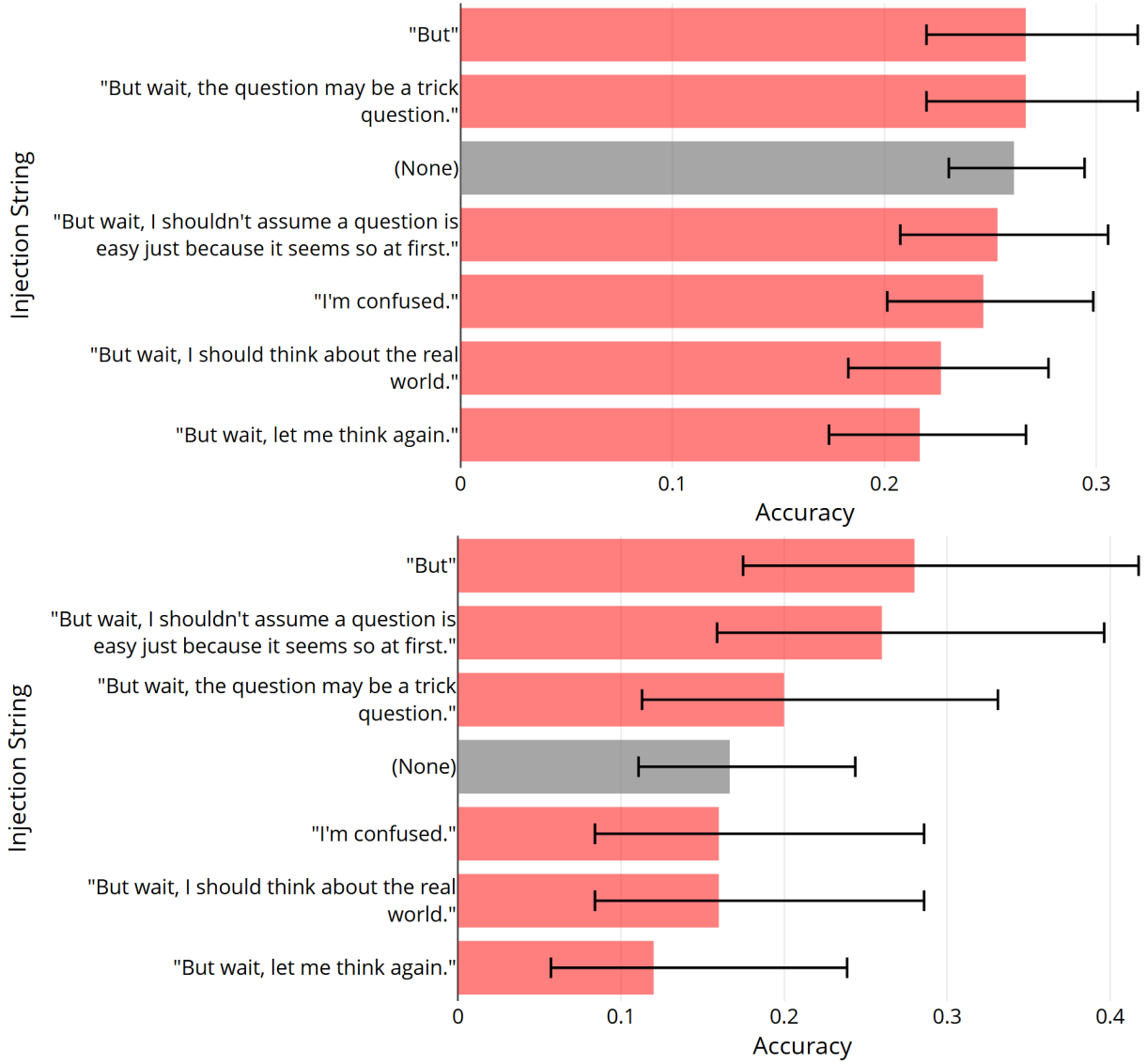


Figure 15: Accuracy at a temperature of 1.0 of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions (top), and question 9 only (bottom), for different Doubt Injection strings. 50 runs were made for injections with $p = 0.25$ (red), and 120 runs for no injection string (grey). Error bars are 95% confidence intervals.

creases steadily between temperature values $T = 0.0$ and $T = 1.0$, before sharply declining beyond $T = 1.1$. In line with our motivating results and findings by Guo et al. (2025), degradation in response quality at higher temperatures is primarily due to language mixing and more frequent generations past the token limit. The accuracy curves for $p = 0$ and $p = 0.5$ closely follow one another in shape, indicating that the general temperature–performance trend is preserved under Doubt Injection.

Interestingly, although DeepSeek-AI (2025) recommend a sampling temperature of $T =$

0.6, we observe peak accuracy on this dataset at $T = 1.0$. Moreover, we find that using $T = 1.0$ significantly reduces the probability of exceeding the 10,000-token generation limit—by a factor of approximately 20 compared with $T = 0.6$.

Initial results using a broader range of injection probabilities (omitted here for visual clarity but provided in Figure 23 of Appendix B.2) suggested that $p = 0.5$ may yield slightly better performance than $p = 0$ for temperature settings around $T = 1$. However, when the sample size was greatly increased in this range,

the results converged to the trend shown in Figure 14. For a temperature of $T = 1.0$, we estimate model accuracies of 24.2% with Doubt Injection and 25.4% without. A comparison of the resulting posterior distributions (as described in Section 3.2.5) indicates a 71% probability that Doubt Injection at $p = 0.5$ results in lower accuracy at this temperature. This result is not conclusive, and larger sample sizes would be required to reduce uncertainty. Additionally, outcomes on this dataset may differ under higher token generation limits, which were not explored in this study—primarily due to the computational cost constraints of running large-scale evaluations.

3.3.3 Effect of Injection String

The effect of the specific Doubt Injection string on accuracy for the subset of SimpleBench questions is shown in the top graph of Figure 15. The injection of “But” results in the highest observed accuracy—26.7%—compared to 26.1% with no injection. However, this difference is marginal. Comparing the posterior probability distributions of LLM accuracy for each accuracy (see Figure 16), we observe significant overlap in probability mass. The posterior for $p = 0$ has a sharper peak due to a larger sample size (120 dataset runs, versus 50 for $p = 0.25$). Monte Carlo sampling from both distributions estimates only a 58.1% probability that Doubt Injection actually improves accuracy, indicating that more data is required to reach a conclusive result.

Nevertheless, some individual questions show a much clearer benefit from Doubt Injection. The bottom graph of Figure 15 focuses on question 9 (whose wording is provided in Table 4). For this question, accuracy increases from 16.7% to 28.0% by randomly injecting the string “But” into the Chain-of-Thought. A comparison of posterior distributions indicates a 95.4% probability that Doubt Injection improves performance on this question.

Out of the 6 questions analyzed, accuracy improves for 3 under Doubt Injection of “But”, remains unchanged for 1, and worsens for 2. Notably, question 8 is the only question

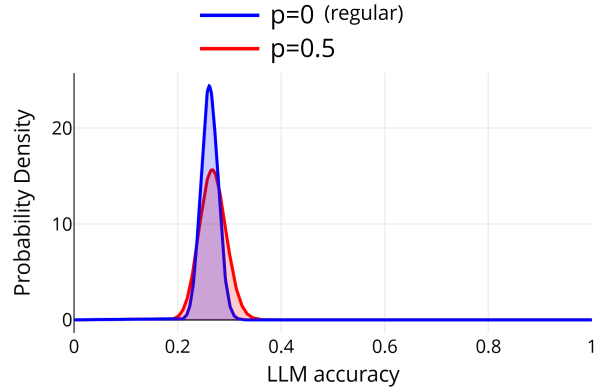


Figure 16: Posterior probability density functions of true accuracy of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions. Comparing Doubt Injection string “But” with $p = 0.0$ (120 runs, 26.1% accurate) and $p = 0.25$ (50 runs, 26.7% accurate).

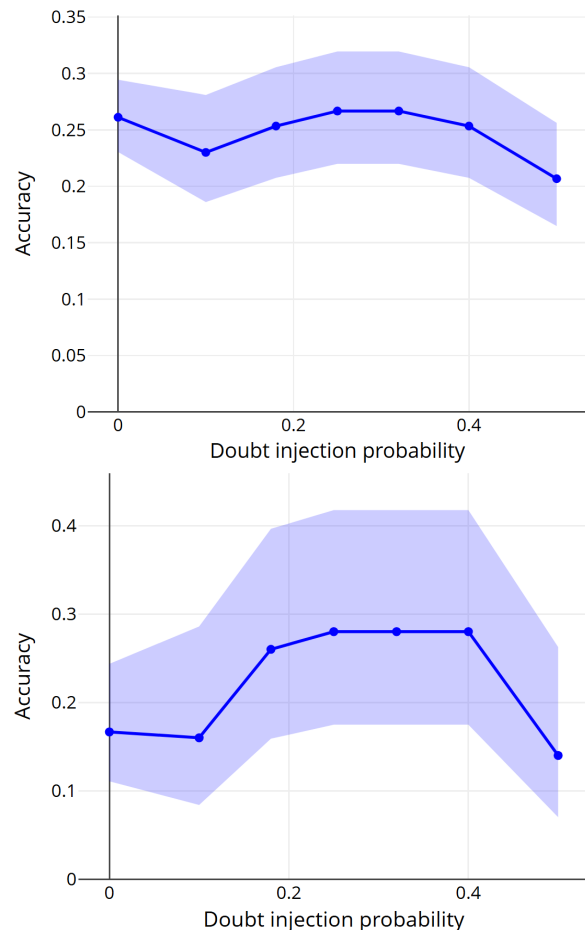


Figure 17: Accuracy of Doubt Injection string “But” on the subset of SimpleBench questions (top) and question 9 only (bottom) against probability of injection. DeepSeek-R1-Distill-Qwen-32B, temperature 1.0. Shaded regions are 95% confidence intervals.

for which none of the tested Doubt Injection strings make an improvement over no injection, implying that for a specific question, there likely exists *some specific injection string* which improves the performance. In general, the performance of Doubt Injection—and the best-performing injection string—is found to be highly dependent on the specific question asked. A more comprehensive study into the effects of different injection strings and probabilities—across diverse question types and datasets—may yield further insights into what settings are most effective.

3.3.4 Effect of Injection Probability

We evaluate the effect of varying the Doubt Injection probability for the injection string “But”. The results are shown in the top graph of Figure 17. Performance initially declines at low injection probabilities, then peaks gently around $p \approx 0.3$, before degrading again by $p = 0.5$.

As with earlier experiments, the increase in overall accuracy on the SimpleBench subset due to Doubt Injection remains statistically insignificant. However, a more focused analysis of question 9—plotted in the bottom graph of Figure 17—reveals consistent and substantial performance gains for injection probabilities $p \in \{0.25, 0.32, 0.4\}$. Posterior probability distribution comparisons indicate a 95.4% confidence in actual improvement in LLM accuracy at each of these settings—again highlighting the question-dependent nature of the performance of Doubt Injection.

3.3.5 Effect of Injection Probability on the AIME 2024 Benchmark

The AIME 2024 dataset comprises 30 challenging multi-step mathematical problems. We study the effect of varying the injection probability of the string “But” into the Chain-of-Thought. Each setting is evaluated over four passes through the dataset using the prompt structure provided in Table 13 of Appendix A.2, with temperature fixed at 0.6—consistent with the DeepSeek-R1 setup (Guo et al., 2025).

As shown in the top graph of Figure 18,

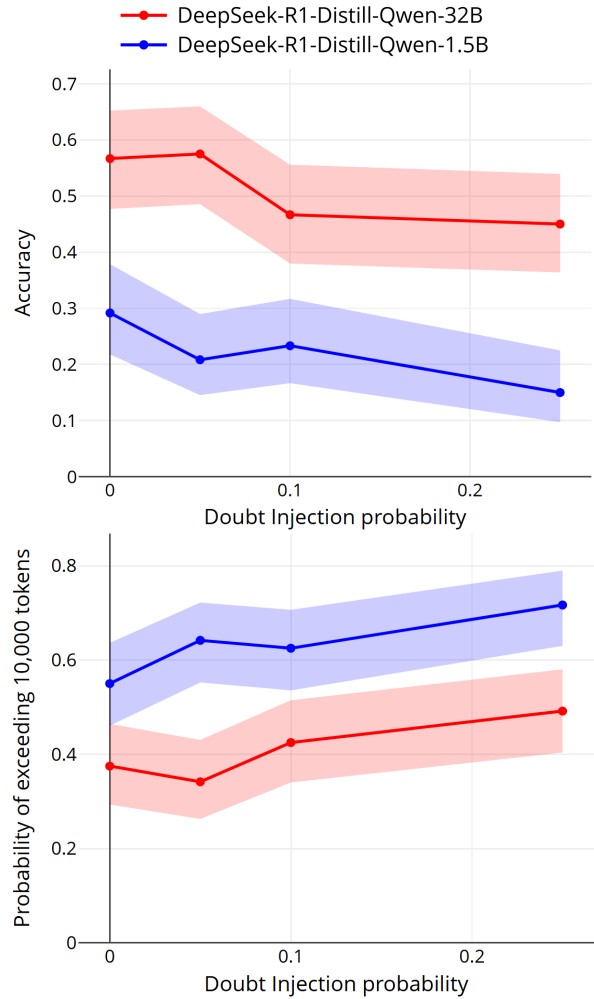


Figure 18: Top: Accuracy and 95% confidence intervals of different LLMs on the AIME 2024 benchmark with Doubt Injection of “But” at $T = 0.6$, against injection probability p . Bottom: Corresponding probability of exceeding the 10,000-token generation limit. Four independent runs were conducted per configuration.

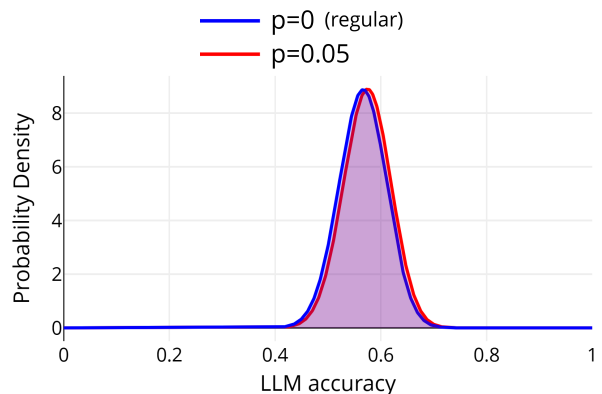


Figure 19: Posterior probability density functions of true accuracy of DeepSeek-R1-Distill-Qwen-32B on four runs of AIME 2024 benchmark, comparing Doubt Injection string “But” with $p = 0.0$ (56.7% accurate) and $p = 0.05$ (57.5% accurate).

the accuracies obtained in our experiments without Doubt Injection do not exactly match those reported by Guo et al. (2025), who achieve pass@1 accuracies of 28.9% and 72.6% with DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-32B, respectively. In contrast, our results are 29.2% and 56.7%. This discrepancy is likely due to our use of a shorter token generation limit (10,000 tokens) compared to their 32,768-token limit. This constraint led to 55% and 38% of our responses respectively being truncated and thus marked as incorrect for the two models.

As shown in Figure 18, a slight increase in accuracy is observed for the 32B model at an injection probability of $p = 0.05$ (57.5%) compared to $p = 0.0$ (56.7%). However, this difference is again not statistically significant. The overlap of the posterior Beta distributions in Figure 19 reveals that there is only a 55% probability that the true accuracy with Doubt Injection exceeds that without. More runs through the dataset would be required to increase the certainty in the result. This demonstrates the importance of uncertainty-aware evaluation, as many papers might misinterpret such a difference as evidence of improvement.

Overall, the trend shown in Figure 18 for both 32B and 1.5B models is a decrease in accuracy as injection probability increases, though the 32B model exhibits a less pronounced decline. This suggests greater robustness to perturbations in the reasoning process for larger models.

A contributing factor to the decline in accuracy with increasing Doubt Injection probability could be the increase in probability of exceeding the 10,000 generation limit, as seen in the bottom graph of Figure 18. This suggests that at higher injection probabilities, there is lower probability that the LLM will end the Chain-of-Thought with the `</think>` token even if the generated response is significant in length.

Nonetheless, the general trend is highly dependent on the specific question. Certain problems show consistent improvement with Doubt Injection, while others degrade sharply. For

Table 6: AIME 2024 accuracy of DeepSeek-R1-Distill-Qwen-1.5B with and without Doubt Injection of “But” at $T = 0.6$. 4 dataset runs for 10,000 token limit, 16 dataset runs for 32,768.

Generation limit (tokens)	Regular ($p = 0$)	Doubt Injection ($p = 0.1$)
10,000	0.292	0.233
32,768	0.292	0.296

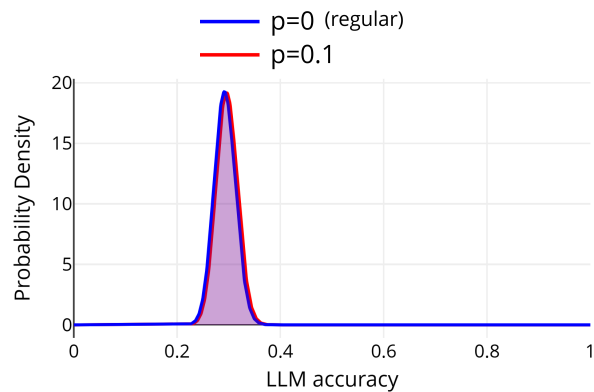


Figure 20: Posterior probability density functions of true accuracy of DeepSeek-R1-Distill-Qwen-1.5B on 16 runs of AIME 2024 benchmark, comparing Doubt Injection string “But” with $p = 0.0$ (29.2% accurate) and $p = 0.05$ (29.6% accurate).

example, for problem 11, there is a posterior probability of 91.7% that Doubt Injection genuinely improves accuracy.

Despite these isolated gains, there is no consistent pattern in question topic or structure that predicts whether accuracy will improve or degrade under Doubt Injection. This variability across problems reinforces the claim that the impact of Doubt Injection is highly problem-specific. As observed in the SimpleBench evaluation, we find no convincing evidence that this particular injection strategy yields a reliable improvement in model performance across the dataset.

3.3.6 Increasing the Generation Limit on the AIME 2024 Benchmark

To assess whether the 10,000-token generation limit constrained the performance of Doubt Injection, we increase the generation cap to 32,768 tokens, matching the setup used in

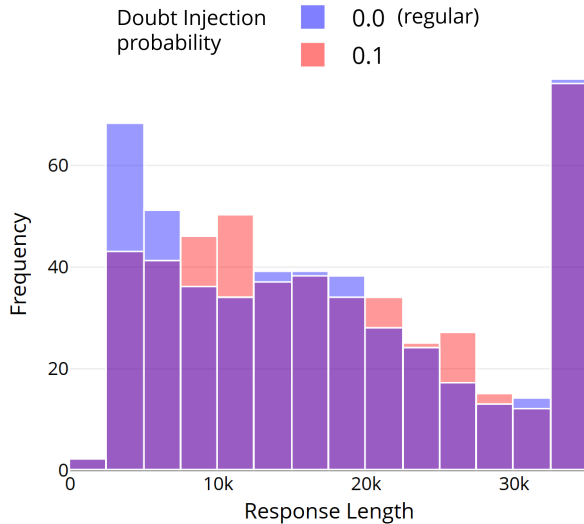


Figure 21: Histogram of different response lengths (in tokens) of DeepSeek-R1-Distill-Qwen-1.5B on 16 runs of the AIME 2024 benchmark with Doubt Injection of “But” at $T = 0.6$. Generation was limited to 32,768 tokens.

DeepSeek’s original implementation. In Table 6, we compare standard generation to Doubt Injection with probability $p = 0.1$ for DeepSeek-R1-Distill-Qwen-1.5B.

The accuracy of standard generation remains unchanged despite the increased generation length, implying that responses exceeding 10,000 tokens were largely incorrect and thus truncated prematurely without loss of correct completions. In contrast, the accuracy penalty previously observed with Doubt Injection is now eliminated—and in fact, performance slightly surpasses that of standard gen-

eration. However, due to the limited number of benchmark runs (16), the posterior distributions for accuracy with and without Doubt Injection still largely overlap (Figure 20). Monte Carlo sampling yields only a 55% posterior probability that Doubt Injection offers a genuine accuracy benefit.

On average, Doubt Injection increases response length from 16,847 to 17,620 tokens. The distribution of response lengths provided in Figure 21 shows that standard generation more frequently terminates in the 2.5k–5k token range, while Doubt Injection responses tend to extend into the 10k–12.5k range. This suggests that Doubt Injection *prevents early conclusions from being drawn*, and instead cause the model to produce longer Chains-of-Thought—as a result of more perceived uncertainty cues being present in the text.

Importantly, the histogram reveals that the original 10,000-token limit would have disproportionately truncated more Doubt Injection responses. At the new 32,768-token ceiling, both standard and injected generations have a similar likelihood of reaching the maximum length. We can therefore conclude that the previous 10k-token constraint was likely a significant limiting factor in the underperformance of Doubt Injection on the AIME 2024 benchmark. Future investigations should ensure that generation limits do not introduce bias when investigating methods like Doubt Injection.

4 Conclusions

This report explored methods to improve the factual reliability of LLMs through contrastive decoding and intervention in the CoT reasoning process.

In the first part of the work, we proposed *CAD-DoLa*, a technique that combines CAD (Shi et al., 2024), which contrasts outputs with and without context, and DoLa (Chuang et al., 2023), which contrasts internal layer states. While CAD-DoLa did not notably improve performance in knowledge-conflict scenarios—where external context contradicts model priors—it partially mitigated drawbacks of CAD in general knowledge settings, where relevant context is supplied (i.e., RAG-like scenarios).

Building on insights from visualising how CAD shifts probability mass through the output distribution, we proposed *Additive CAD*, which redefines CAD by summing scaled differences between contextual and non-contextual token probabilities instead of multiplying by their ratios. This additive approach significantly improves performance in knowledge-conflict settings (from 33.5% to 42.7%), but negatively impacts performance in general knowledge scenarios where context is available—highlighting a trade-off between aggressiveness in overriding model priors and preserving general knowledge accuracy.

Our experiments also revealed strong sensitivity to prompt phrasing, which may account for inconsistencies in CAD performance reported across the literature. We showed that a specific decoding and evaluation setup can make CAD beneficial in RAG settings—as reported by the original CAD paper—but that these benefits can be surpassed by simpler changes to the prompting scheme.

In the second part of the work, we examined how the Chain-of-Thought (CoT) reasoning process—in reasoning LLMs—behaves under adversarial questioning. We found that at higher temperatures (close to 1), the model explored a broader idea space and produced more correct intermediate steps, improving overall reasoning performance.

Motivated by this observation, we proposed a technique called *Doubt Injection*, in which statements that imply doubt or hesitation (e.g., “But”) are inserted during CoT generation at new paragraphs to subtly redirect the reasoning process. Although this method produced clear improvements on specific logical and arithmetic questions, the overall gains (from 56.7% to 57.5% on arithmetic questions) were not statistically significant at the evaluated sample sizes.

With smaller reasoning models on arithmetic reasoning questions, our research showed that the 10,000-token generation cap implemented initially had suppressed the observed effectiveness of Doubt Injection compared with regular decoding. Increasing the generation cap to 32,768 brought performance of Doubt Injection above that of regular decoding, but again statistically insignificant at the sample size used.

Overall, this work highlights both the potential and the limitations of contrastive decoding and Chain-of-Thought interventions as strategies for mitigating hallucinations in LLMs.

4.1 Future Work

The Additive CAD method introduced in this report demonstrates potential as a more effective alternative to the standard multiplicative contrastive decoding scheme. Since DoLa and other contrastive methods such as Contrastive Decoding (Li et al., 2022a) also rely on multiplicative ratios of distributions, future work could explore whether these other contrastive decoding use cases benefit from an additive formulation similar to that proposed here.

A natural extension of Additive CAD might involve generalising from a single pairwise contrast to a weighted combination of multiple distributional differences. For instance, instead of comparing just one with-context and no-context distribution pair, it may be possible to combine several distributions generated under different prompts. Learning or optimising the weights in such combinations might improve accuracy, but it may also introduce additional complexity and potentially reduce

generalisability.

For the Doubt Injection technique, the current experiments only covered a limited subset of possible parameter settings. Constraints on computational resources prevented broader exploration of factors such as model size, injection position, temperature, and the diver-

sity of injection strings. Future work could investigate sampling from a richer set of injection phrases and placing them at alternative discourse boundaries (e.g., sentence ends, or based on the number of tokens generated since the last injection) rather than only randomly at paragraph breaks.

References

- Anthropic. 2025. [Reduce hallucinations](#). Accessed: 2025-05-23.
- Molly Bohannon. 2023. [Lawyer used chatgpt in court—and cited fake cases. a judge is considering sanctions](#). *Forbes*. Accessed: 2025-05-26.
- Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.
- DeepSeek-AI. 2025. Deepseek-r1-distill-qwen-32b. <https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B>. Accessed: 2025-05-25.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022a. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022b. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.
- Alisa Liu and Jiacheng Liu. 2023. The memotrap dataset. <https://github.com/inverse-scaling/prize/blob/main/data-release/README.md>. GitHub repository.
- MAA. 2024. American invitational mathematics examination - aime. <https://maa.org/student-programs/amc/>. American Invitational Mathematics Examination - AIME 2024.
- OpenAI. 2025a. [Introducing openai o1](#). Accessed: 2025-05-24.
- OpenAI. 2025b. [Safety evaluations hub](#). Accessed: 2025-05-23.
- Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.
- Philip and Hemang. 2024. Simplebench: The text benchmark in which unspecialized human performance exceeds that of current frontier models. <https://simple-bench.com/>. Retrieved from <https://simple-bench.com/>.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Vipula Rawte, Swagata Chakraborty, Agnibh Pathak, Anubhav Sarkar, SM_Towhidul Islam Tonmoy, Aman Chadha, Amit Sheth, and Amitava Das. 2023. The troubling emergence of hallucination in large language models-an extensive definition, quantification, and prescriptive remediations. *Association for Computational Linguistics*.

- Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Adacad: Adaptively decoding to balance conflicts between contextual and parametric knowledge. *arXiv preprint arXiv:2409.07394*.
- Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Kevin Wu, Eric Wu, and James Y Zou. 2024. Clashes: Quantifying the tug-of-war between an llm’s internal prior and external evidence. *Advances in Neural Information Processing Systems*, 37:33402–33422.
- Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. 2024. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. *arXiv preprint arXiv:2402.11893*.

Appendix

A Prompts and Responses

A.1 CAD Prompt Templates

Table 7: Prompt templates of the original CAD paper (Shi et al., 2024). “{context}” and “{question}” denote the where specific context c and prompt x (as in Section 2.2.1) are inserted.

Prompt for $P(y_t c, x, y_{<t})$:
{context} {question}
Prompt for $P(y_t x, y_{<t})$:
{question}

Table 8: Prompt templates of the COIECD paper (Yuan et al., 2024). “{context}” and “{question}” denote the where specific context c and prompt x (as in section 2.2.1) are inserted.

Prompt for $P(y_t c, x, y_{<t})$:
Given the following information: {context}
Answer the following question based on the given information with one or few words: {question}
Answer:
Prompt for $P(y_t x, y_{<t})$:
Answer the following question based on your internal knowledge with one or few words: {question}
Answer:

Table 9: Prompt templates of the AdaCAD paper (Wang et al., 2024). “{context}” and “{question}” denote the where specific context c and prompt x (as in Section 2.2.1) are inserted.

Prompt for $P(y_t c, x, y_{<t})$:
{context} Using only the references listed above, answer the following question:
Question: {question}.
Answer:
Prompt for $P(y_t x, y_{<t})$:
Answer the following question: Question: {question}. Answer:

Table 10: CAD and Additive CAD Prompt template used in this paper. “{context}” and “{question}” denote the where specific context c and prompt x (as in Section 2.2.1) are inserted.

Prompt for $P(y_t c, x, y_{<t})$:
Instruction: read the given information and answer the corresponding question.
{context}
Q: {question}
A:
Prompt for $P(y_t x, y_{<t})$:
Instruction: answer the corresponding question.
Q: {question}
A:

A.2 Doubt Injection Prompts

Table 11: Full input prompt for the altered river crossing problem.

Full input prompt:
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User:
Solve the following puzzle: A man and a goat are on the left side of a river.
There is a wolf and a cabbage on the right side of the river.
The man has a boat.
The boat can carry only the farmer and a single item.
How can the farmer get the goat to the right side of the river?
Assistant:

Table 12: Prompt template for the SimpleBench test. “{question}” denotes the location of where the specific SimpleBench question is inserted.

Prompt template:

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags, respectively, i.e., `<think>` reasoning process here `</think>` `<answer>` answer X here `</answer>` where X is one of the letters A, B, C, D, E, or F. Do not include additional formatting in the final answer. User:

{question}
Assistant:

Table 13: Prompt template for the AIME 2024 benchmark test. Note that the begin-of-sentence, user and assistant tokens provided here aren’t the exact unicode representations of those tokens. “{question}” denotes the location of where the specific AIME 2024 question is inserted.

Prompt template:

`<|begin_of_sentence|><|User|>`You will be given a problem. Please reason step by step, and put your final answer within `\boxed{ }`: {question}`<|Assistant|><think>`

A.3 Doubt Injection Example

The following is an example input and partial response of DeepSeek-R1-Distill-Qwen-1.5B with doubt injection of $p = 0.25$ for an AIME 2024 benchmark question. Injections of “But” are highlighted. Special tokens of the start and end of sequence are shown, in line with the prompt template of Table 13.

Input:

`<|begin_of_sentence|><|User|>`You will be given a problem. Please reason step by step, and put your final answer within `\boxed{ }`:
Alice and Bob play the following game. A stack of n tokens lies before them. The players take turns with Alice going first. On each turn, the player removes

either 1 token or 4 tokens from the stack. Whoever removes the last token wins. Find the number of positive integers n less than or equal to 2024 for which there exists a strategy for Bob that guarantees that Bob will win the game regardless of Alice’s play.`<|Assistant|><think>`

Response:

Okay, so Alice and Bob are playing this game with a stack of n tokens. Alice goes first, and each turn a player can remove either 1 or 4 tokens. The person who takes the last token wins. I need to figure out how many positive integers $n \leq 2024$ Bob can force a win by using a strategy, regardless of what Alice does. Hmm, this seems like a combinatorial game theory problem.

But I need to think through it step by step.

First, the grid is 5×5 , so 25 cells in total. Each cell can have either a white chip, a black chip, or be empty. But with the constraints given, I need to make sure that in every row and every column, all the chips are the same color. That means if I have a row with some white chips, all the chips in that row must be white, and similarly for columns. The same applies to black chips.

But wait, $n=1$: Alice takes 1 token and wins. So Bob can’t win when $n=1$.

But $n=2$: Alice can take 1 token, leaving 1 for Bob, who then takes it and wins. Or she can take 4, but $n=2$ is too small. So if $n=2$, Alice can win by taking 1 token. Wait, but actually, if $n=2$, Alice takes 1, leaves 1 for Bob, who takes it and wins. So Bob doesn’t win. So $n=2$: Alice wins.

Wait, that seems a bit confusing. Let me rephrase. For each n , if the current player can force a win, then that player has a winning strategy. So, for $n=1$: Alice takes 1 and wins. For $n=2$: Alice can take 1, leaving 1 for Bob, who then takes it and wins. So Bob can’t win because Alice can take 1 and leave 1 for Bob, who then takes it and wins. So for $n=2$, Bob doesn’t have a winning strategy.

Wait, maybe I need to think about winning and losing positions. In combinatorial game theory, a position is a losing position if every move from that position leads to a winning position for the opponent. Conversely, a position is a winning position if there’s at least one move that leads to a losing position for the opponent.

...

B Additional Experimental Results

B.1 CAD-DoLa and Additive CAD-DoLa

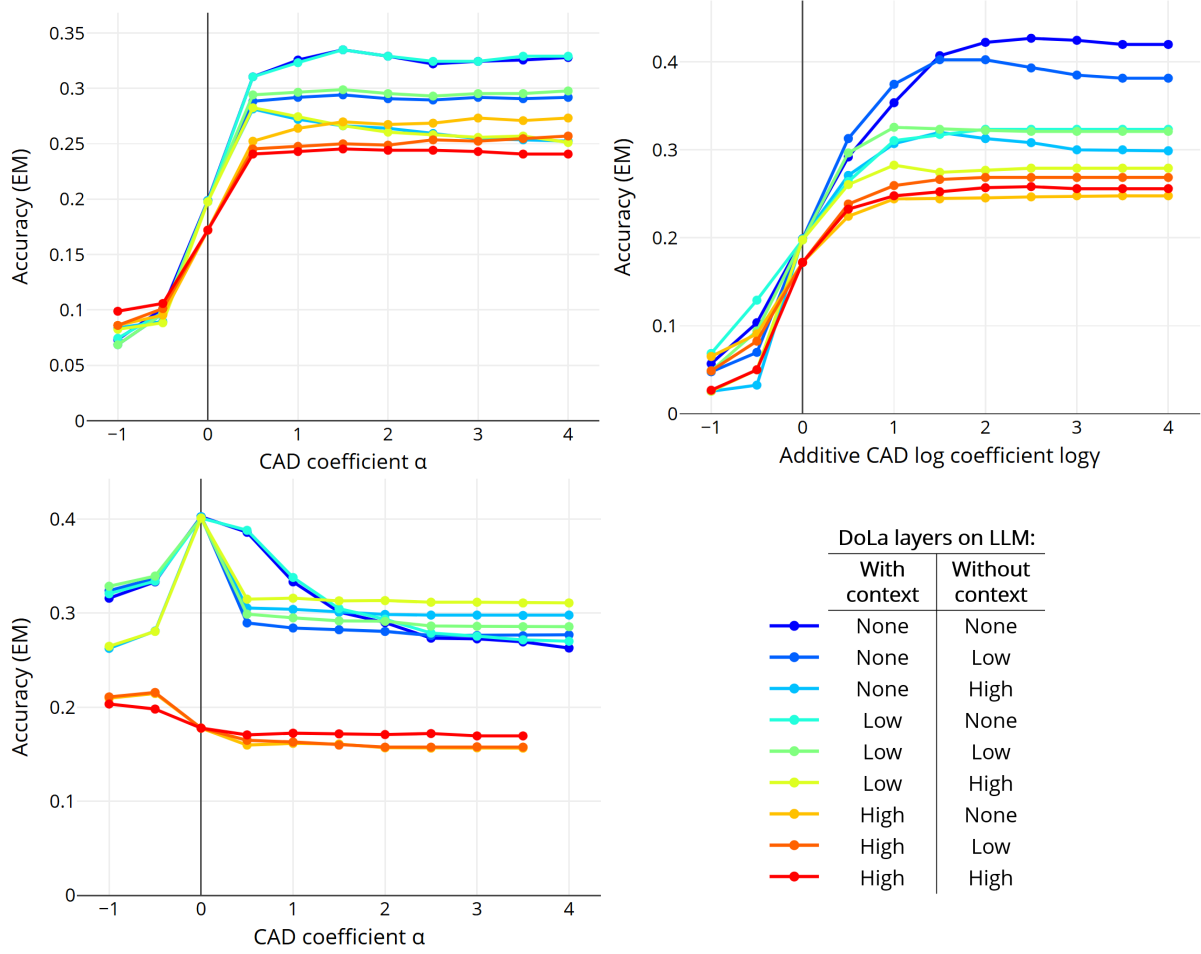


Figure 22: EM accuracy of LLaMA-2-7B for regular/Additive CAD-DoLa schemes. DoLa inactive: “none”. DoLa restricted to upper transformer layers: “high”; lower layers: “low”. Top left: CAD-DoLa on Memotrap. Top right: Additive CAD-DoLa on Memotrap. Bottom left: CAD-DoLa on Natural Questions.

Table 14: Natural Questions EM scores (/2773) of LLaMA-2-7B for Additive CAD-DoLa schemes. DoLa inactive: “none”. DoLa restricted to upper transformer layers: “high”; lower layers: “low”.

DoLa layers on LLM:		Additive CAD log-coefficient $\log_{10}(\gamma)$											
With context	Without context	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
None	None	784	1021	1116	931	689	598	552	539	518	509	508	
None	Low	830	926	1112	1001	868	814	803	802	802	802	801	
Low	None	768	928	1116	811	591	393	262	181	153	144	141	
Low	Low	775	941	1112	929	784	787	784	785	785	785	785	

Table 15: MemoTrap EM scores (/860) of LLaMA-2-7B for CAD-DoLa schemes. DoLa inactive: “none”. DoLa restricted to upper transformer layers: “high”; lower layers: “low”.

DoLa layers on LLM:		CAD coefficient α											
With context	Without context	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
None	None	63	87	171	267	280	288	283	277	279	280	282	
None	Low	59	83	171	248	251	253	250	249	251	250	251	
None	High	72	77	171	242	234	229	227	223	219	218	217	
Low	None	64	84	170	267	278	288	283	279	279	283	283	
Low	Low	59	83	170	253	255	257	254	252	254	254	256	
Low	High	71	76	170	243	236	229	224	222	220	221	216	
High	None	74	82	148	217	227	232	230	231	235	233	235	
High	Low	74	87	148	211	213	215	214	218	217	219	221	
High	High	85	91	148	207	209	211	210	210	209	207	207	

Table 16: MemoTrap EM scores (/860) of LLaMA-2-7B for Additive CAD-DoLa schemes. DoLa inactive: “none”. DoLa restricted to upper transformer layers: “high”; lower layers: “low”.

DoLa layers on LLM:		Additive CAD log-coefficient $\log_{10}(\gamma)$											
With context	Without context	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
None	None	49	89	171	251	304	350	363	367	365	361	361	
None	Low	41	60	171	269	322	346	346	338	331	328	328	
None	High	22	28	171	233	264	275	269	265	258	258	257	
Low	None	59	111	170	227	267	273	278	278	278	278	278	
Low	Low	42	81	170	255	280	278	277	276	276	276	276	
Low	High	22	43	170	224	243	236	238	240	240	240	240	
High	None	56	78	148	193	210	210	211	212	213	213	213	
High	Low	42	71	148	205	223	229	231	231	231	231	231	
High	High	23	43	148	200	213	217	221	222	220	220	220	

Table 17: Natural Questions EM scores (/2773) of LLaMA-2-7B for CAD-DoLa schemes. DoLa inactive: “none”. DoLa restricted to upper transformer layers: “high”; lower layers: “low”.

DoLa layers on LLM:		CAD coefficient α											
With context	Without context	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
None	None	876	924	1116	1070	924	835	804	758	756	747	729	
None	Low	898	934	1116	803	788	783	778	766	766	765	768	
None	High	728	779	1116	847	843	835	828	826	826	825	826	
Low	None	889	926	1112	1076	937	845	814	773	764	753	749	
Low	Low	911	941	1112	829	818	809	808	794	793	792	792	
Low	High	734	778	1112	873	876	868	869	864	864	862	862	
High	None	581	595	493	443	448	446	435	434	435	434	-	
High	Low	585	598	493	457	452	443	437	437	438	437	-	
High	High	564	549	493	473	478	475	474	477	470	470	-	

Table 18: Natural Questions EM scores (/2773) of LLaMA-2-7B across different CAD prompting schemes.

Input prompt style (paper)	CAD coefficient α											
	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
Ours	876	924	1116	1070	924	835	804	758	756	747	729	
CAD (Shi et al., 2024)	210	277	248	133	128	129	130	123	116	111	119	
COIECD (Yuan et al., 2024)	306	355	389	380	315	260	240	234	231	227	229	
AdaCAD (Wang et al., 2024)	495	796	1339	1007	1058	1027	993	992	982	974	974	

Table 19: Natural Questions EM scores (/2773) of LLaMA-2-7B across different Additive CAD prompting schemes.

Input prompt style (paper)	Additive CAD log-coefficient $\log_{10} \gamma$											
	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	
Ours	784	1021	1116	931	689	598	552	539	518	509	508	
CAD (Shi et al., 2024)	179	274	248	154	103	90	86	79	79	79	79	
COIECD (Yuan et al., 2024)	242	303	389	356	284	216	195	173	169	169	169	
AdaCAD (Wang et al., 2024)	784	1021	1116	931	689	598	552	539	518	509	508	

B.2 Doubt Injection

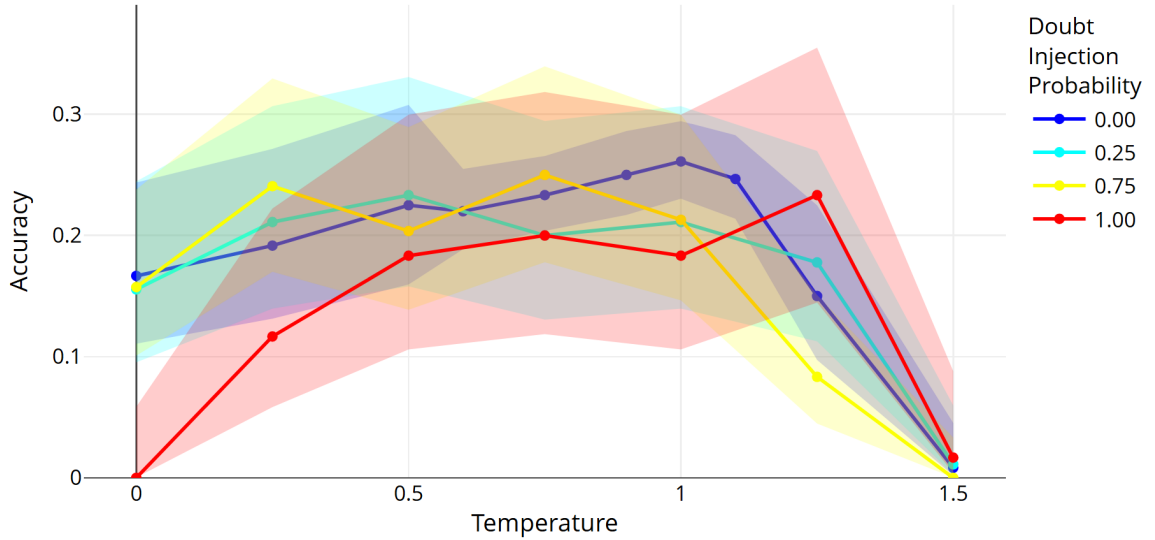


Figure 23: Accuracy of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions, with Doubt Injection of “But wait, let me think again.” Number of runs vary for each configuration. See Figure 14 for $p = 0.5$ curve.

Table 20: Correct responses/total responses of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions, with Doubt Injection of “But wait, let me think again.” Number of runs vary for each configuration.

Doubt injection probability	Temperature									
	0.0	0.25	0.5	0.6	0.75	0.9	1.0	1.1	1.25	1.5
0.0	20/120	23/120	27/120	132/600	168/720	150/600	188/720	148/600	18/120	1/120
0.25	14/90	19/90	21/90	-	18/90	-	19/90	-	16/90	1/90
0.5	20/120	21/120	23/120	126/600	155/720	144/600	179/720	138/600	16/120	1/120
0.75	17/108	26/108	22/108	-	27/108	-	23/108	-	9/108	0/108
1.0	0/60	7/60	11/60	-	12/60	-	11/60	-	14/60	1/60

Table 21: Correct responses/total responses of DeepSeek-R1-Distill-Qwen-32B at temperature 1.0 on the subset of SimpleBench questions, with Doubt Injection of different strings. Number of runs vary for each configuration.

Injection string	Correct/total
"But"	80/300
"But wait, the question may be a trick question."	80/300
(None)	188/720
"But wait, I shouldn't assume a question is easy just because it seems so at first."	76/300
"I'm confused."	74/300
"But wait, I should think about the real world."	68/300
"But wait, let me think again."	65/300

Table 22: Correct responses/total responses of DeepSeek-R1-Distill-Qwen-32B at temperature 1.0 on the subset of SimpleBench questions, with Doubt Injection of “But”. Number of runs vary for each configuration.

Doubt injection probability	0.0	0.1	0.18	0.25	0.32	0.4	0.5
SimpleBench score	188/720	69/300	76/300	80/300	80/300	76/300	62/300

Table 23: Correct responses (out of 120) of different models on the AIME 2024 benchmark with Doubt Injection of “But” at $T = 0.6$. Four independent runs of the dataset were conducted per configuration.

Model	Doubt Injection Probability			
	0.0	0.05	0.1	0.25
DeepSeek-R1-Distill-Qwen-32B	68	69	56	54
DeepSeek-R1-Distill-Qwen-1.5B	35	25	28	18

C Risk Assessment Retrospective

No notable hazards presented an issue during the course of this project.