LIGHTWEIGHT LATENT REASONING FOR NARRATIVE TASKS

Alexander Gurung, Nikolay Malkin & Mirella Lapata
Department of Informatics
University of Edinburgh
{alex.gurung,nmalkin}@ed.ac.uk, mlap@inf.ed.ac.uk

ABSTRACT

Large language models (LLMs) tackle complex tasks by generating long chains of thought or "reasoning traces" that act as latent variables in the generation of an output given a query. A model's ability to generate such traces can be optimized with reinforcement learning (RL) to improve their utility in predicting an answer. This optimization comes at a high computational cost, especially for narrative-related tasks that involve retrieving and processing many tokens. To this end, we propose LiteReason, a latent reasoning method that can be interleaved with standard token sampling and easily combined with RL techniques. LiteReason employs a lightweight Reasoning Projector module, trained to produce continuous latent tokens that help the model 'skip' reasoning steps. During RL, the policy model decides when to activate the projector, switching between latent and discrete reasoning as needed. Experimental results on plot hole detection and book chapter generation show that our method outperforms latent reasoning baselines and comes close to matching non-latent RL training, while reducing final reasoning length by 77–92%. Overall, LiteReason guides RL training to a more efficient part of the performance-computation tradeoff curve.[1]

1 INTRODUCTION

Reasoning has become a popular paradigm for improving LLM performance across tasks. Originally introduced through prompting methods such as Chain-of-Thought (Wei et al., 2022), which has models generate intermediate tokens (traces) before the final answer, reasoning has since evolved toward RL-based approaches that refine the ability of models to generate high-quality reasoning traces. Tasks in domains where answers can be verified, like math and coding, have seen significant improvement from RL training using the verifier as a reward signal (Shao et al., 2024; Lambert et al., 2025). Recent work has begun extending RL methods to non-verifiable domains like story generation and understanding (Ahuja et al., 2025; Gurung & Lapata, 2025).

However, performance gains from RL often come at increased cost during inference and training as reasoning traces become longer (Shen et al., 2025a). This has motivated efforts to improve the *efficiency* of reasoning, aiming to speed up both RL and inference-time generation (Sui et al., 2025). One such effort is **latent reasoning**, which produces traces formed of *continuous* token embeddings instead of or in addition to discrete tokens. The justification for these latent reasoning methods is twofold: firstly, many tokens are perceived not to contain information useful for predicting the answer; secondly, human reasoning is often done implicitly without verbalization. Keeping thoughts in a non-token space affords flexibility, allowing us to retain more information than lossily decoding to tokens. For example, Hao et al. (2025) argue that continuous embeddings can represent a mixture of reasoning steps, which effectively approximates performing several reasoning rollouts in parallel.

In this work, we introduce LiteReason, a latent reasoning algorithm designed for narrative understanding and generation tasks, where both inputs and outputs often span thousands of tokens. LiteReason adds a Reasoning Projector head to a base LLM, allowing the generation both of text tokens via discrete sampling and of continuous embeddings via **latent reasoning mode**.

[1] Code to replicate results is hosted here.

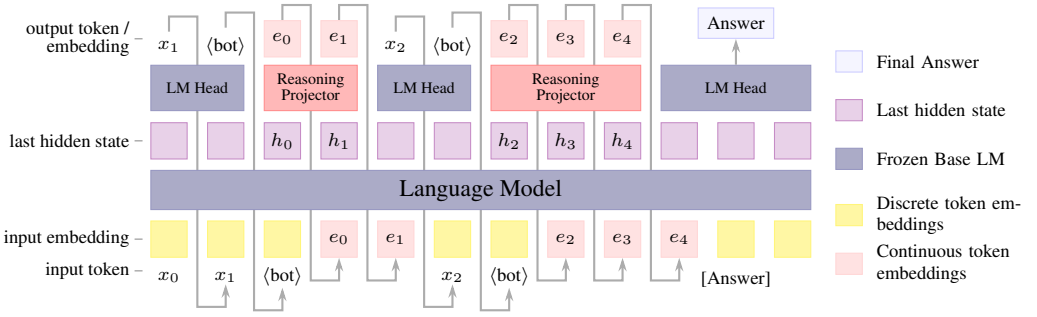

Figure 1: High-level diagram of LiteReason. Discrete sampling (via the LM head) is performed as normal, selecting a token (e.g., x_1) and passing its corresponding discrete token embedding, until we encounter special 'implicit-thought' tags represented here by "$\langle bot \rangle$". We then switch to **latent reasoning mode** and use the Reasoning Projector to directly predict continuous token embeddings (e.g., e_0) for a number of forward passes before switching back to discrete sampling. We can switch between discrete and reasoning mode multiple times before producing the final answer with discrete sampling. When training the Reasoning Projector we 1) randomly replace reasoning steps with the implicit thought tags and 2) freeze the rest of the LLM and apply a cross-entropy loss on the remaining reasoning steps (discrete tokens).

This mode is activated by the production of a special start-of-thought symbol. (Figure 1 for a schematic illustration.)

After pretraining the Reasoning Projector, we perform RL fine-tuning of the base model to encourage efficient reasoning using latent tokens and to improve performance at a given task.

LiteReason is especially well-suited to narrative tasks. In particular, we focus on detecting plot holes in stories (FlawedFictions; Ahuja et al. 2025) and generating book chapters, referred to as Next-Chapter Prediction (NCP; Gurung & Lapata 2025). Both tasks require a variety of story-understanding capabilities, asking models to reason over $>$ 1k tokens of story context to respectively detect plot inconsistencies and plan the next chapter.

They also introduce some key difficulties when adapting existing latent reasoning approaches. Firstly, they lack large datasets of high-quality reasoning traces, and because default model performance is low, collecting such datasets is difficult. Secondly, they require high levels of adaptability, as the input space of stories is very diverse. These challenges guide the design of LiteReason, which balances language model exploration with lower inference costs.

We evaluate LiteReason on both verifiable (Flawed Fictions) and non-verifiable (NCP) tasks and compare against trained (Hao et al., 2025; Tan et al., 2025) and training-free latent reasoning methods (Zhuang et al., 2025; Zhang et al., 2025). We also include RL-trained models without latent reasoning as an upper bound for performance and a benchmark for computational costs. We find that LiteReason outperforms all latent-reasoning baselines and reaches 69–96% of the performance gain of non-latent RL training while generating 50-53% fewer tokens during training and generating 70% fewer tokens during inference. Our core contributions and findings are:

- We introduce LiteReason, a new method that integrates latent reasoning with RL, relying only on the pretrained LLM (and no preexisting dataset of reasoning traces) and designed for low-resource narrative tasks.
- Empirical results on Flawed Fictions (verifiable) and Next-Chapter Prediction (non-verifiable) tasks show high performance relative to latent reasoning and RL baselines.
- Further analysis shows our method results in highly efficient solutions and requires significantly fewer tokens during training and inference than traditional RL.

2 RELATED WORK

Initial work demonstrated that Chain-of-Thought prompting can enhance LLM capabilities by generating intermediate tokens before predicting the final answer (Wei et al., 2022; Kojima et al., 2022). Recent research has successfully applied RL to optimize these reasoning traces, particularly in math and code (Shao et al., 2024; Kimi Team et al., 2025; DeepSeek-AI et al., 2025; Lambert et al., 2025). However, these performance gains often come with increased inference costs as reasoning chains become longer (Shen et al., 2025a). Furthermore, as RL training often requires generating many reasoning traces, the cost of training can also be significant. Consequently there has been interest in making reasoning more *efficient*, exploring the tradeoff between reasoning length and performance (Sui et al., 2025).

Reasoning in the Latent Space Latent reasoning attempts to improve efficiency by performing reasoning in a continuous space instead of sampling discrete tokens. Two high-level strategies have been explored: *training-free*, which modifies the sampling procedure without altering the model itself, and *training-based*, which teaches the model to construct latent representations of reasoning.

Soft Thinking (Zhang et al., 2025) and Mixture of Inputs (Zhuang et al., 2025) are both examples of training-free strategies. The former uses a probability weighted mixture of the top-k token embeddings as the next input and adds an entropy-based stopping mechanism to switch back to discrete sampling. The latter similarly uses a mixture of embeddings, but the weights are determined via a Bayesian estimation method where the sampled token is treated as an observation. Although these methods are lightweight to implement, they typically only improve performance when the model's initial capabilities on the task are high. Both methods apply 27B+ models on datasets like AIME, GSM8K (Cobbe et al., 2021), GPQA Diamond (Rein et al., 2024) and LiveCodeBench (Jain et al., 2024), where pre-existing Chain-of-Thought performance is over 80%.

One influential training-based method is COCONUT (Hao et al., 2025), which uses an LLM's final hidden state as the next input embedding and proposes a training curriculum that iteratively replaces more reasoning steps with these hidden states. Training the entire model to predict the remaining reasoning steps and final answer encourages it to store latent reasoning representations in this final hidden state. COCONUT is built on GPT-2 and trained to perform math and synthetic logical reasoning tasks on large datasets like GSM8K-Aug (Deng et al., 2024). CoLaR (Tan et al., 2025) proposes adding a latent head that predicts token embeddings, supervised by the token embeddings of replaced reasoning steps. After fine-tuning on a large reasoning database (GSM8K-Aug), CoLaR performs RL directly over its latent tokens. Similar to COCONUT, at test-time CoLaR produce a series of latent tokens followed by standard discrete sampling and an answer. CoLaR is scaled up to larger models (Llama 3.2 1-8B Instruct) and is also trained with GSM8K-Aug for math benchmarks.

Other approaches include CoT2 (Gozeten et al., 2025), which learns to produce token mixtures similar to training-free methods, applied to GPT-2 for logical reasoning tasks; CODI (Shen et al., 2025b) uses a teacher-student distillation framework to improve performance on GSM8k (for GPT-2 and Llama 3.1 1B models), and Token Assorted (Su et al., 2025) which learns new token embeddings via a VQ-VAE, also training Llama 3.1 8B for math and logical reasoning tasks.

Our LiteReason approach differs from these trained methods in a few ways. Firstly, we do not rely on a large dataset of reasoning traces like GSM8k-Aug (as such datasets do not exist for our tasks), instead exclusively using traces generated from the model itself. Secondly, we only train a *lightweight Reasoning Projector* for latent reasoning, *not the entire model*, as we hypothesize over-training the base-LLM for token-embedding generation may destabilize the model's token-generation capabilities. Thirdly, we allow interleaving of latent and discrete reasoning, hypothesizing that over the course of RL training the model will learn a useful balance between them. Finally, although many existing approaches compare against Chain-of-Thought prompting and a few of them incorporate RL into their approaches, to our knowledge none of them compare final performance against a non-latent RL-trained model. We believe this baseline to be a more realistic point of comparison for difficult tasks, so include it in our experiments as an upper bound on performance and compute.

Narrative Tasks Although much reasoning work has focused on domains like math and coding, there is a large body of work applying LLMs to narrative understanding and generation tasks (Mostafazadeh et al., 2016; Fan et al., 2018; Huot et al., 2024; Yang et al., 2022; 2023; Huot et al., 2024; Xie & Riedl, 2024; Chhun et al., 2022; Karpinska et al., 2024; Sprague et al., 2023). Many of

these studies prompt models to perform specific sub-tasks such as plot planning or generating character details (Huot et al., 2024; Yang et al., 2023; 2022), eliciting specific reasoning styles. However, these tasks are often very difficult to adapt for RL domain due to their lack of verifiable rewards. We focus in this work on two specific tasks that evaluate narrative-based reasoning in LLMs, while still providing clear reward signals for policy-gradient methods.

Ahuja et al. (2025) constructed the Flawed Fictions benchmark by automatically synthesizing plot holes in human written stories, and tasks models to predict whether the given story contains a plot hole. This task requires various cognitive abilities, including theory of mind, accurate state tracking, and commonsense reasoning. We focus on the verifiable binary prediction task (Yes/No answer: does the story have a plot hole?), where initial performance is only slightly above 50% (ra!ndom).

Gurung & Lapata (2025) introduced the difficult Next-Chapter-Prediction (NCP) task, where LLMs predict *plans* for the next chapter in a book based on information about its story, characters, and previous chapters. During RL, the quality of a plan or reasoning trace is evaluated by how much it improves the likelihood of the true next chapter. This non-verifiable task is especially challenging for existing latent reasoning methods, as the output (plans) are long and linguistically diverse.

3 The LiteReason framework

In this section, we introduce a method that integrates latent reasoning with reinforcement learning (RL) to improve performance while substantially reducing computational cost. This method is designed around key assumptions imposed by the nature of narrative analysis and generation tasks:

1. We lack a significant dataset of high-quality reasoning traces. This limitation is especially common for long-form generation tasks, or analysis tasks where the reasoning-space is very large. This is also often true for tasks where RL is chosen *because* exploration is needed to achieve good performance and distillation is impossible. We do however assume our tasks provide a reward function for evaluating a given response.
2. We want to retain standard language modeling capabilities, as our responses are highly diverse.
3. We assume our model is capable of some reasoning, so we can initialize our method from reasoning traces taken from the model itself.

Our only architectural addition to the base LLM is a small Reasoning Projector, composed of a small stack of MLP layers. Similar to previous latent-reasoning work (e.g., Hao et al. 2025), this projector takes in the last hidden state and predicts a continuous token-embedding vector. This module is only called during latent-reasoning; otherwise, the last hidden state is passed to the language model head and a discrete token is sampled like normal. We refer to these two pathways as **latent reasoning mode** and **discrete mode**, respectively. Figure 1 shows how these modes are interleaved throughout inference and training, and this process is described in more detail below.

Our goal is to train the Reasoning Projector (via supervised fine-tuning; SFT) such that its 'latent thoughts' help the LLM skip reasoning steps, and the LLM (via RL) to both improve at the given task *and* choose when to use the Reasoning Projector.

3.1 Training LiteReason

There are three stages to training LiteReason, described below: (1) Data Collection, (2) SFT Initialization, and (3) RL Training. Our inference procedure is shown in Figure 1.

Data Collection We first wish to collect a small dataset of reasoning traces to initialize the Reasoning Projector to produce 'latent thoughts' (token embeddings) useful for skipping reasoning steps. We construct the initial SFT dataset via rejection sampling, where we sample n reasoning traces for every prompt in the training set and filter out the traces with non-positive reward (Accuracy for Flawed Fictions, Contrastive Improvement for NCP; see §4 for details). We split these reasoning traces into steps by sentences and randomly replace a proportion s_r of them with **implicit thought tags**, which have the structure: `⟨implicit_thought⟩#⟨/implicit_thought⟩` where # is an integer representing the number of steps to take in latent reasoning mode. At each step we pass the final hidden state to the Reasoning Projector and append the predicted token embedding to the

sequence. These predicted 'latent thoughts' are trained to maximize the likelihood of reasoning tokens. We set # as a proportion t_r of the number of tokens in the sentence being replaced. We expect these hyper-parameters, t_r (token-replacement ratio), s_r (sentence-replacement ratio), and n (sample number), to be optimized for new model-task combinations. We tuned these values in preliminary experiments to balance cost and the amount of information latent tokens represent. The chosen values are reported in Table 10.

Supervised Fine-Tuning (SFT) Once we have a dataset of reasoning traces with steps randomly replaced with implicit-thought tags, we aim to train our Reasoning Projector to predict useful token embeddings for 'skipping' reasoning steps. We freeze all model parameters except the projector and train using a standard cross-entropy loss on the remaining reasoning sentences. This encourages the projector to produce embeddings that 'bridge the gap' between explicit reasoning steps when skipping over the masked reasoning step. As later latent reasoning tokens depend on earlier ones, this requires backpropagation through a rollout of several steps of latent reasoning.

RL Training After the Reasoning Projector is initialized, we improve performance on a given task via RL using the reward functions defined in §4. By incorporating latent reasoning into this process, we hope to lower the computational cost of improved performance. Although our RL training is agnostic to the choice of policy-gradient algorithm, we conducted our experiments using Group Relative Policy Optimization (GRPO; Shao et al. 2024) to ensure consistency with previous work on latent reasoning and narrative generation (Gurung & Lapata, 2025; Tan et al., 2025). For each training instance, we sample a group of outputs $\{o_1, o_2, ...o_G\}$ from the old policy π_{old}, where G is the group size (a hyper-parameter). The reward for each output is converted to an advantage via group normalization and used to update the policy.

During RL, we train the whole LLM, but do not compute policy gradients with respect to the prediction of latent thought tokens. In other words, we consider only the token sampling steps as 'actions', some of which are conditioned on past latent thoughts. As a result, the Reasoning Projector is not updated during RL, although the prediction of latent thoughts may change because the discrete token and latent thought predictors share the model body. In experiments, we refer to this variant as 'LiteReason w/o SFT'. However, we hypothesize that during training our Reasoning Projector may fall out-of-date with the current reasoning traces produced by our model.

To alleviate this, after every RL training epoch we add an SFT step that performs the same procedure as before, but on the reasoning traces produced during the epoch. In addition to the latent tokens produced by the model, we also randomly replace a proportion of sentences with latent tokens, just as in the SFT phase. We find this further encourages the model to produce efficient reasoning.

3.2 Inference procedure

During inference we default to **discrete mode**, but when encountering implicit-thought tags we switch to **latent reasoning mode** (see Figure 1). As mentioned earlier, we delineate between discrete and latent reasoning modes via **implicit thought tags**. Note that these tags are not special tokens in the LLM's vocabulary, so instead of training the model to encourage their use we can simply prompt the model. The model also predicts the number #, deciding how many latent tokens are produced. Initial usage of these tags is low, but during early RL training they become much more common.

During latent reasoning mode, for the given number (#) of steps we do not sample a token. Instead we pass the last hidden state to the Reasoning Projector and feed the output as the next token embedding. After the last reasoning step we switch back to sampling tokens as normal. Similar to other methods with latent reasoning heads (e.g., CoLaR; Tan et al. 2025), this incurs a small computational cost when producing latent tokens. In practice, latent tokens are about 8% more expensive than normal tokens, but this cost is far outweighed by savings from producing fewer total tokens.

4 Application to narrative tasks

We briefly present the Flawed Fictions and Next-Chapter Prediction tasks used to evaluate our method. We describe how these tasks are formulated and how a non-latent reasoning model is trained via RL. As described below, both tasks require complex story-driven reasoning that is se-

Method	Flawed Fictions Accuracy (%)	Flawed Fictions Avg # Tokens	NCP Contra Improve (%)	NCP Avg # Tokens
Qwen2.5-7B	57.26 $\pm$ 1.58	400.04 $\pm$ 6.50	0.067 $\pm$ 0.001	831.85 $\pm$ 8.26
RL-Trained	88.71 $\pm$ 0.00 (+31.45%)	114.53 $\pm$ 2.01 (-71.37%)	0.666 $\pm$ 0.01 (+894.03%)	721.33 $\pm$ 6.01 (-15.97%)
LiteReason	87.42 $\pm$ 0.47 (+30.16%)	34.01 $\pm$ 0.38 (-91.50%)	0.478 $\pm$ 0.01 (+613.43%)	193.11 $\pm$ 2.02 (-77.50%)
w/o SFT	85.48 $\pm$ 0.00 (+28.22%)	8.26 $\pm$ 0.17 (-97.94%)	0.560 $\pm$ 0.01 (+735.82%)	622.23 $\pm$ 1.82 (-27.51%)
w/o RP	87.58 $\pm$ 0.76 (+30.32%)	260.81 $\pm$ 3.68 (-34.80%)	0.449 $\pm$ 0.01 (+570.15%)	983.49 $\pm$ 5.30 (+14.57%)

Table 1: We validate LiteReason design variants on Flawed Fictions (left) and Next-Chapter Prediction (right). All variants are trained using GRPO with the same hyper-parameters described in §3. They all drastically improve over the untrained baseline, but the full LiteReason variant best balances high performance with significant token savings. Scores are averages across the test set, with $\pm$SEM denoting the standard error of the mean. Percent differences are relative to the default Qwen 2.5-7B model (first row). Accuracy percent differences are method $-$ default. All other columns are unbounded values, so we compute percent difference as $\frac{\text{method}-\text{default}}{\text{default}}$. Contra Improve refers to our Contrastive Improvement reward metric. RP abbreviates Reasoning Projector.

mantically and stylistically distinct than traditional tasks used by latent-reasoning methods. Table 9 displays example reasoning steps, and Table 7 has dataset statistics.

Flawed Fictions is a benchmark proposed by Ahuja et al. (2025) that tests narrative-based reasoning abilities like theory-of-mind and state-tracking. It is constructed on short stories from Project Gutenberg by FlawedFictionsMaker, an algorithm that controllably *induces* plot holes and continuity errors using LLMs, and filtered with human annotators (see Table 7 for dataset statistics).

Models are given a story and asked to respond Yes or No if the story contains a plot hole. Open-source models like Qwen2.5-32B score around 53%, only slightly better than random (50%). Although the benchmark can be extended to longer stories or specific-line plot hole detection, in this work we focus on the original binary prediction task, reporting Accuracy as the performance metric.

Our reward for RL is a simple binary RLVR-style score $R_{\text{flawed}} = \mathbf{1}[\text{pred} = \text{answer}]$ (Lambert et al., 2025), and we use a slight prompt change that we found improved RL training performance. More details and prompts are presented in Appendix D.

Next-Chapter-Prediction, as proposed by Gurung & Lapata (2025), involves generating a detailed plan $\hat{p}$ (given Story-Information SI_i) for the next chapter of a book c_{i+1}, and then generating the chapter itself based on that plan. This task is challenging, requiring reasoning over 10k+ tokens and producing rich long-form answers (the plans are around 100 tokens). Gurung & Lapata (2025) collate a dataset of 30 books published in or after 2024 (mean length of 139k), which we also use in experiments. To evaluate the generated plan they propose VR-CLI (Verifiable Rewards via Completion Likelihood Improvement), a proxy reward formulation that aims to improve the likelihood of generating the true next chapter, conditioned on both the Story-Information and the plan.

We extend the VR-CLI reward with a **contrastive term** that negatively weights the likelihood of another chapter (c^r) randomly chosen from another book: $R_{\text{contr}} = I(SI_i, c_{i+1}, \hat{p}) - \gamma I(SI_i, c^r, \hat{p})$ The contrastive term improves diversity in the final plans and reasoning traces (Table 8), which we believe is a better test case for LiteReason. We use this reward during training and as our primary automated metric. To avoid confusion, we refer to our reward R_{contr} as **Contrastive Improvement**. We present more details in Appendix B and compare example completions in Table 11.

5 Experimental Setting

We perform all our experiments with Qwen 2.5-7B Instruct (Qwen et al., 2024), as it has shown good performance on the NCP task (Gurung & Lapata, 2025) and is amenable to RL training on the Flawed Fictions dataset (GRPO training leads to a large increase in performance, from 57% to 89%). We call this GRPO baseline RL-Trained, as described in §5.2.

5.1 Testing LiteReason variants

We test each component of LiteReason through experiments on Flawed Fictions and NCP. We use the following LiteReason instantiations which are all RL-trained, but vary in whether they employ

SFT or the proposed Reasoning Projector. We use these hyper-parameters (§3) when applicable: $t_r = 0.2, s_r = (10\%, 25\%)$, and $n = 5$, where $s_r = (10\%, 25\%)$ means a naive equal mixture of datapoints with $s_r = 10\%$ and $s_r = 25\%$, which we found worked well even without a curriculum.

LiteReason Our standard method, where we periodically (once an epoch) run a small SFT training step using the trajectories generated during that epoch. This adds some computational cost, but updates the reasoning-projector with the current model's reasoning. We also use an implicit-thought prompt (Tables 12 and 13) that encourages model usage of the implicit thought tags.

LiteReason w/o SFT This version keeps the implicit prompt and the initialized Reasoning Projector that uses the implicit thought tags to know when to pass new reasoning embeddings to the model. However, this variant does not perform SFT during RL training to update the Reasoning Projector.

LiteReason w/o Reasoning Projector Finally, we evaluate performance post RL without the Reasoning Projector (RP), but with the implicit thought prompt. This method does not use the latent reasoning, and tests if the model performs differently if we prompt for skipping reasoning steps.

5.2 Comparison methods

Drawing from the recent latent reasoning literature, we select a representative set of previous methods to test our design assumptions (in §3). We evaluate these methods on Flawed Fictions and NCP.

Training-Free Latent Reasoning We compare against two recent training-free methods: (1) **Mixture of Inputs** (MoI; Su et al. 2025) blends the sampled token's embedding with those from the remaining token distribution; and (2) **Soft Thinking** (Zhang et al., 2025) also combines token embeddings, but uses an entropy metric to choose when to switch back to discrete sampling.

Trained Latent Reasoning: We also compare against two trained latent reasoning methods: (1) **COCONUT** (Hao et al., 2025) trains the whole LLM to use the last hidden state as a token embedding, and applies a curriculum to slowly learn to predict these embeddings instead of reasoning steps; and (2) **CoLaR** (Tan et al., 2025), which uses a non-deterministic latent head to predict compressed embeddings of the next reasoning step, and also has separate SFT and RL stages.

Finally, we compare against *non-latent* RL, **RL-Trained**, using GRPO (Shao et al., 2024) on our tasks. This approach represents *upper bound* performance as well as its expected token cost.

Hyper-parameters were chosen based on validation performance, and train the full LLM for all methods. RL training used the same hyper-parameters across all methods, with the exception of method-specific choices set to defaults or adapted for the task. More details are in Appendix D.

5.3 NCP human evaluation

Although we optimize next-chapter plans via the **Contrastive Improvement** metric described previously, we are also interested in the quality of the chapters they induce. To evaluate the chapters generated from the plans, we follow the same human evaluation procedure as Gurung & Lapata (2025). We elicit human preferences on the chapters via pairwise comparisons, along the dimensions of Plot, Creativity, Development, Language Use, Characters, and Overall Preference.

Due to the high cost of human annotations, we compare the following representative sample of methods: Default (untrained Qwen model), RL-Trained, Mixture-of-Inputs, CoLaR, and LiteReason. We generate chapters with Qwen 2.5 7B-Instruct based on the plans generated by each method, and collect 20 pairwise comparisons for every method-combination. We compare model performance by Bradley-Terry relative strengths fit to these comparisons. More details are presented in Appendix E.

6 Results

6.1 What is the best LiteReason design?

We first perform ablations on the Flawed Fictions and NCP tasks and report results in Table 1. We benchmark the LiteReason variants against two non-latent baselines built upon the Qwen2.5-7B-Instruct model: a prompted default model and its RL-trained version.

Method	Flawed Fictions Accuracy (%)		Flawed Fictions Avg # Tokens		NCP Contra Improve (%)		NCP Avg # Tokens	
Qwen2.5-7B	57.26 ± 1.58		400.04 ± 6.50		0.067 ± 0.01		831.85 ± 8.26	
MoI	58.06 ± 1.75	(+0.80%)	401.73 ± 6.57	(+0.42%)	0.045 ±0.03	(-32.84%)	829.14 ±3.13	(-2.46%)
Soft Thinking	57.42 ±1.21	(+0.16%)	360.90 ±2.96	(-9.78%)	0.013 ±0.02	(-80.75%)	966.73 ±4.29	(+16.21%)
COCONUT	50.65 ±0.02	(-6.61%)	268.07 ±6.68	(-32.99%)	0.083 ±0.00	(+23.88%)	361.84 ±0.92	(-56.50%)
CoLaR	53.71 ±0.01	(-3.55%)	226.14 ±10.00	(-43.47%)	0.118 ±0.01	(+58.21%)	218.40 ±2.54	(-73.75%)
LiteReason	87.42 ± 0.47	(+30.16%)	34.01 ±0.38	(-91.50%)	0.478 ±0.01	(+613.43%)	193.11 ±2.02	(-77.50%)
RL-Trained	88.71 ± 0.00	(+31.45%)	114.53 ± 2.01	(-71.37%)	0.666 ±0.01	(+894.03%)	721.33 ±6.01	(-15.97%)

Table 2: Comparing method performance on Flawed Fictions (left) and Next-Chapter Prediction (right). The top and bottom block refer to untrained and trained methods respectively. For both tasks LiteReason performs significantly better than all baseline methods, and comes closest to matching non-latent RL performance. Furthermore, LiteReason produces the fewest number of tokens of any method, indicating it has learned a more efficient style of reasoning. Scores are averages across the test set, ±SEM denotes the standard error of the mean. Accuracy percent differences are method - default. Other columns are unbounded values, so we compute percent difference as $\frac{\text{method}-\text{default}}{\text{default}}$. Contra Improve refers to our Contrastive Improvement reward metric.

We find that **the full LiteReason variant best balances high performance with large token reductions**, achieving similar gains to traditional non-latent RL ($\frac{30.16}{31.45} = 95.9\%$ and $\frac{613.43}{894.03} = 68.6\%$ of the performance increase for Flawed Fictions and NCP, respectively) while generating over three times fewer tokens. LiteReason w/o SFT shows mixed performance: on Flawed Fictions it performs slightly worse than LiteReason but with fewer tokens, but on NCP it performs the highest of our ablations but produces significantly more tokens (622 compared to LiteReason's 193). We hypothesize that the usefulness of periodic SFT will depend on task-specific features like difficulty and dataset size. Different LiteReason hyper-parameters (like swap-ratio and how often the Reasoning Projector is updated) may produce results in between these extremes, allowing practitioners to balance performance and efficiency as desired. We also find that LiteReason w/o RP (i.e., just prompting to skip reasoning steps) increases reasoning length and performs even worse than the RL-Trained baseline. Table 4 shows more comparisons with/without the implicit-thought prompt on different models.

We also briefly explored the usage of our latent thoughts throughout training to see if RL is increasing or decreasing their likelihood. We find that at the beginning of RL training there is a sharp upward trend in latent-thought usage, however this behaviour often lowers by the end of training. Future work could explore *inducing* latent reasoning by increasing the likelihood of the implicit-token markers or simply changing the prompt.

6.2 How does LiteReason compare against other latent methods?

Table 2 compares the performance of several reasoning methods on Flawed Fictions and NCP tasks, split into non-trained and trained method blocks. Qwen2.5-7B Instruct is considered 'default' performance, and the last row shows upper bound performance (Qwen2.5-7B-Instruct trained with RL).

Our results show that **LiteReason significantly outperforms existing latent reasoning methods on both tasks.** Relative to the Qwen 2.5 7B-Instruct baseline and all other latent-reasoning methods, LiteReason sees significant performance gains. Accuracy on Flawed Fictions goes up 30.16% and on NCP Contrastive Improvement goes up >6× (Table 2). In contrast, the next-best performing latent-reasoning methods only marginally improve performance (0.8% on Flawed Fictions and 58% on NCP). We hypothesize this is because our method is most comparable in approach to the RL-Trained baseline, and better explores alternate reasoning paths that the base model would not generate.

Unsurprisingly we find that the training-free methods, Mixture-of-Inputs (MoI) and Soft Thinking perform close to the untrained model in performance and token length. On NCP, Soft Thinking is marginally better than the baseline but actually produces slightly more tokens, while MoI performs worse than the baseline with slightly fewer tokens. On Flawed Fictions, the differences to the baseline are within confidence intervals. Future research is needed to make the efficiency and performance gains of training-free approaches more consistent and pronounced across tasks.

Table 2 further examines the quality of generated book chapters on the NCP task through human judgments, comparing the untrained Qwen2.5-7B model against MoI, CoLaR, and a non-latent RL-trained model. We find human judgments (measured as relative strengths) largely follow the

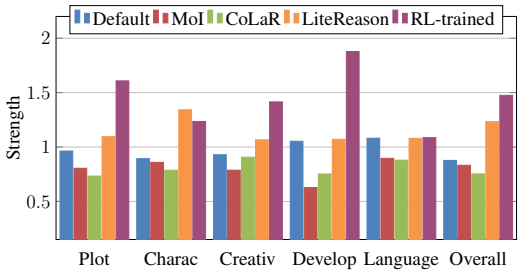

Figure 2: Bradley Terry Relative Strength on NCP task. Default refers to the untrained Qwen2.5-7B model. RL-trained is the upper bound trained with non-latent RL.

Method	FF Tokens	NCP Tokens
RL	61.4M	108.8M
RL+LiteReason	29.0M (-52.8%)	54.9M (-49.5%)

Table 3: Comparing the number of discrete tokens generated during RL (GRPO) training with and without LiteReason. We find that our method produces significantly fewer tokens during RL training, despite achieving comparable performance.

contrastive improvement trends in Table 2 with default RL training performing best, followed by LiteReason, and then a notable gap before the remaining methods.

6.3 Does LiteReason benefit from RL?

We also examine RL's effect on relevant methods: non-latent reasoning, CoLaR, and LiteReason Tables 5 and 6 in the Appendix show performance for these methods before and after RL. As mentioned previously, we consider the standard non-latent RL training as an upper bound on performance.

CoLaR is notable in its use of RL *in the latent space*, as opposed to exclusively token-based RL (LiteReason and standard RL training). However, we find CoLaR fails to significantly benefit from RL training, performing only marginally better than the original SFT initialization in both performance and efficiency (Tables 5 and 6). Both CoLaR variants perform only marginally better than the default model, indicating that they largely compress, not extend, existing model's capabilities. In contrast, **LiteReason benefits significantly from RL**, moving from near default-model performance to close to the RL-trained upper bound.

6.4 Does LiteReason improve RL efficiency?

Table 3 shows the number of tokens generated *during RL training* with and without LiteReason. As can be seen, **LiteReason requires significantly fewer tokens during RL training**. Although we train for the *same* number of steps and samples, LiteReason uses 52.8% fewer tokens on the Flawed Fictions task (29M vs. 61.4M) and 49.5% fewer on the NCP task (54.9M vs. 108.8M).

In addition to requiring fewer tokens during training, Table 2 shows that our method also generates substantially fewer tokens during inference: 77% fewer on the NCP task and 92% fewer on Flawed Fictions, relative to the base model. Taken together, our results demonstrate that **LiteReason guides RL to more efficient solutions**. When compared against the RL-Trained model, LiteReason produces traces 73% and 70% smaller that still achieve 96% of the performance *gains* on the Flawed Fictions task compared to non-latent RL Training, and 69% for NCP. This tradeoff between performance and inference cost is further explored in Appendix C and shown in Figure 3 and Figure 4 for Flawed Fictions and NCP respectively. Compared to other methods, we perform significantly closer to the RL-Trained model, while improving on token efficiency.

7 Conclusion

We presented LiteReason, a latent-reasoning method designed to work in narrative domains and integrate well with RL training. LiteReason trains a lightweight Reasoning Projector to produce useful latent tokens and teaches an LLM to interleave these latent thoughts with discrete sampling.

We evaluate our method against several training-free and trained latent reasoning baselines on two challenging tasks (Flawed Fictions and Next-Chapter Prediction) that require long-context understanding and diverse reasoning. We find that LiteReason achieves the closest performance to traditional RL training while requiring less than half the generated tokens during training and shrinking

the reasoning generated at inference time by 66%. To our knowledge, we are the first to apply continuous latent reasoning to narrative reasoning and the first to combine continuous latent reasoning with discrete token sampling within an RL framework. In the future, we hope to apply our method to other tasks and with different base models. Alternate Reasoning Projector architectures could also be considered, such as recursively updating a single latent thought or injecting latent thoughts at locations other than the token embedding stage.

REFERENCES

Kabir Ahuja, Melanie Sclar, and Yulia Tsvetkov. Finding Flawed Fictions: Evaluating Complex Reasoning in Language Models via Plot Hole Detection. August 2025. URL https://openreview.net/forum?id=ptmgWRCWmu#discussion.

Cyril Chhun, Pierre Colombo, Fabian M. Suchanek, and Chloé Clavel. Of Human Criteria and Automatic Metrics: A Benchmark of the Evaluation of Story Generation. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (eds.), *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 5794–5836, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.509/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, November 2021. URL http://arxiv.org/abs/2110.14168. arXiv:2110.14168 [cs].

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025. doi: 10.48550/ARXIV.2501.12948. URL https://arxiv.org/abs/2501.12948. Publisher: arXiv Version Number: 1.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step, May 2024. URL http://arxiv.org/abs/2405.14838. arXiv:2405.14838 [cs].

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL http://aclweb.org/anthology/P18-1082.

Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous Chain of Thought Enables Parallel Exploration and Reasoning, September 2025. URL http://arxiv.org/abs/2505.23648. arXiv:2505.23648 [cs].

Alexander Gurung and Mirella Lapata. Learning to Reason for Long-Form Story Generation. August 2025. URL https://openreview.net/forum?id=dr3eg5ehR2#discussion.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E. Weston, and Yuandong Tian. Training Large Language Models to Reason in a Continuous Latent Space. August 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3#discussion.

Fantine Huot, Reinald Kim Amplayo, Jennimaria Palomaki, Alice Shoshana Jakobovits, Elizabeth Clark, and Mirella Lapata. Agents' Room: Narrative Generation through Multi-step Collaboration. October 2024. URL https://openreview.net/forum?id=HfWcFs7XLR&noteId=xO9yNXp4VJ.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. October 2024. URL https://openreview.net/forum?id=chfJJYC3iL.

Marzena Karpinska, Katherine Thai, Kyle Lo, Tanya Goyal, and Mohit Iyyer. One Thousand and One Pairs: A "novel" challenge for long-context language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17048–17085, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.948. URL https://aclanthology.org/2024.emnlp-main.948/.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025. doi: 10.48550/ARXIV.2501.12599. URL https://arxiv.org/abs/2501.12599. Publisher: arXiv Version Number: 2.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=e2TBb5y0yFf.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya

Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, 2016. doi: 10.18653/v1/N16-1098. URL http://aclweb.org/anthology/N16-1098. Conference Name: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Place: San Diego, California Publisher: Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. 2024. doi: 10.48550/ARXIV.2412.15115. URL https://arxiv.org/abs/2412.15115. Publisher: arXiv Version Number: 2.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. August 2024. URL https://openreview.net/forum?id=Ti67584b98.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024. doi: 10.48550/ARXIV.2402.03300. URL https://arxiv.org/abs/2402.03300. Publisher: arXiv Version Number: 3.

Si Shen, Fei Huang, Zhixiao Zhao, Chang Liu, Tiansheng Zheng, and Danhao Zhu. Long Is More Important Than Difficult for Training Reasoning Models. 2025a. doi: 10.48550/ARXIV.2503.18069. URL https://arxiv.org/abs/2503.18069. Publisher: arXiv Version Number: 1.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing Chain-of-Thought into Continuous Space via Self-Distillation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Zayne Rea Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the Limits of Chain-of-thought with Multistep Soft Reasoning. October 2023. URL https://openreview.net/forum?id=jenyYQzue1.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token Assorted: Mixing Latent and Text Tokens for Improved Language Model Reasoning. 2025. doi: 10.48550/ARXIV.2502.03275. URL https://arxiv.org/abs/2502.03275. Publisher: arXiv Version Number: 1.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *Transactions on Machine Learning Research*, April 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=HvoG8SxggZ.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. Think Silently, Think Fast: Dynamic Latent Compression of LLM Reasoning Chains. 2025. doi: 10.48550/ARXIV.2505.16552. URL https://arxiv.org/abs/2505.16552. Publisher: arXiv Version Number: 4.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. October 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Kaige Xie and Mark Riedl. Creating Suspenseful Stories: Iterative Planning with Large Language Models. arXiv, 2024. doi: 10.48550/ARXIV.2402.17119. URL https://arxiv.org/abs/2402.17119. Version Number: 1.

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. Re3: Generating Longer Stories With Recursive Reprompting and Revision. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4393–4479, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.296. URL https://aclanthology.org/2022.emnlp-main.296/.

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. DOC: Improving Long Story Coherence With Detailed Outline Control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3378–3465, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.190. URL https://aclanthology.org/2023.acl-long.190.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft Thinking: Unlocking the Reasoning Potential of LLMs in Continuous Concept Space. October 2025. URL https://openreview.net/forum?id=ByQdHPGKgU.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Text Generation Beyond Discrete Token Sampling. 2025. doi: 10.48550/ARXIV.2505.14827. URL https://arxiv.org/abs/2505.14827. Publisher: arXiv Version Number: 2.

Method	Implicit Prompt	Accuracy (%)		Avg # Tokens	
Qwen2.5-7B-Instruct	×	57.26 ±1.58		400.04 ±6.50	
Qwen2.5-7B-Instruct	✓	50.65 ±1.85	(-6.61%)	499.21 ±9.20	(+24.79%)
RL-Trained	×	88.71 ±0.00	(+31.45%)	114.53 ±2.01	(-72.46%)
RL-Trained	✓	90.32 ±0.00	(+33.06%)	144.88 ±2.03	(-63.78%)
LiteReason	✓	87.42 ±0.47	(+30.16%)	34.01 ±0.38	(-91.50%)
LiteReason	×	89.68 ±0.36	(+32.42%)	105.80 ±8.46	(-73.55%)
w/o Periodic SFT	✓	85.48 ±0.00	(+28.22%)	8.26 ±0.17	(-97.94%)
w/o Periodic SFT	×	87.26 ±0.16	(+30.00%)	30.57 ±1.98	(-92.36%)
w/o Reasoning Projector	✓	87.58 ±0.76	(+30.32%)	261.69 ±3.72	(-34.58%)
w/o Reasoning Projector	×	86.61 ±0.76	(+29.35%)	258.92 ±3.35	(-35.28%)

Table 4: Comparing LiteReason variants on the Flawed Fictions task, with and without the implicit-thought prompt. Flawed Fictions accuracy percent differences are new − old, while other metrics percent differences are $\frac{\text{new}-\text{old}}{\text{old}}$. We find slight increases in performance for LiteReason variants that use the Reasoning Projector when we switch to the non-implicit-thought prompt, at the cost of much longer reasoning traces. Future work could leverage this prompting ability or sampling to encourage desired performance-compute tradeoff at test-time.

	Flawed Fictions			
Method	Accuracy (%)		Avg # Tokens	
Qwen2.5-7B-Instruct	57.26 ±1.58		400.04 ±6.50	
RL-Trained	88.71 ±0.00	(+31.45%)	114.53 ±2.01	(-71.37%)
LiteReason pre-RL	52.10 ±2.50	(-5.16%)	567.63 ±13.86	(+41.89%)
LiteReason	87.42 ±0.47	(+30.16%)	34.01 ±0.38	(-91.50%)
CoLaR	50.89 ±0.02	(-6.37%)	226.88 ±6.33	(-43.29%)
CoLaR-PostRL	53.71 ±0.01	(-3.55%)	226.14 ±10.00	(-43.47%)

Table 5: Comparing performance on Flawed Fictions before and after RL. LiteReason pre-RL refers to using the implicit-thought prompt and the pretrained Reasoning Projector, immediately after its SFT initialization. We find that for both the base model and LiteReason, RL improves performance and reduces tokens. However, CoLaR-PostRL achieves essentially the same performance and length for Flawed Fictions and worse performance and length for NCP, indicating that it is not able to use RL effectively. Flawed Fictions accuracy percent ± are new − old, while other metrics percent ± are $\frac{\text{new}-\text{old}}{\text{old}}$.

A Further Ablations

Table 4 compares performance with and without the implicit-thought prompt that encourages models to 'skip' reasoning steps with the special tokens. Tables 5 and 6 compares performance before and after RL for the relevant methods.

B Extending VR-CLI with Contrastive Reward

In this section we build on the VR-CLI objective proposed in Gurung & Lapata (2025) to add a contrastive term, which we use as our primary automated metric for our results.

We use the formalisms introduced in the original paper: at a given chapter index i, a *reasoning model* $\pi_\theta^{\mathcal{R}}$ reasons over Story-Information SI_i and predicts a plan $\hat{p} \leftarrow \pi_\theta^{\mathcal{R}}(SI_i)$ for the next-chapter. SI_i contains a global story sketch, previous chapter plot summary and character sheets, the immediately preceding chapter, and a high-level synopsis of the next chapter. A *story-generator model* $\pi^{\mathcal{G}}$ then takes this plan and the Story-Information and predicts the next-chapter $\hat{c}_{i+1} \leftarrow \pi^{\mathcal{G}}(SI_i, \hat{p})$. Both reasoning and generator models are initialized with the same base model (Qwen 2.5 7B-Instruct) but only the reasoning model is trained.

In the original paper, models are trained to with a 'VR-CLI' or "Improvement" (I) based reward. From Gurung & Lapata (2025): the improvement is defined as the *percent improvement in per-token*

Method	NCP Contra Improve		NCP Avg # Tokens	
Qwen2.5-7B-Instruct	0.067 ± 0.01		829.14 ± 3.13	
RL-Trained	0.666 ± 0.01	(+894.03%)	721.33 ± 6.01	(-15.97%)
LiteReason pre-RL	0.041 ± 0.03	(-38.80%)	946.47 ± 31.36	(+13.78%)
LiteReason	0.478 ± 0.01	(+613.43%)	193.11 ± 2.02	(-77.50%)
CoLaR	0.118 ± 0.01	(+58.21%)	218.40 ± 2.54	(-73.75%)
CoLaR-PostRL	0.118 ± 0.01	(+58.21%)	218.40 ± 2.54	(-73.75%)

Table 6: Comparing performance on the Next-Chapter Prediction task before and after RL. LiteReason pre-RL refers to using the implicit-thought prompt and the pretrained Reasoning Projector, immediately after its SFT initialization. We find that for both the base model and LiteReason, RL improves performance and reduces tokens. However, CoLaR-PostRL achieves essentially the same performance and length for Flawed Fictions and never improves on the SFT model for NCP (and thus the 'best' model is the same), indicating that it is not able to use RL effectively. Flawed Fictions accuracy percent $\pm$ are new $-$ old, while other metrics percent $\pm$ are $\frac{\text{new}-\text{old}}{\text{old}}$

perplexity (PPL) when generating y from $\pi^{\mathcal{G}}$, conditioned on (x, a), relative to the perplexity of y conditioned only on x. This measures how much the plan, generated from the reasoning model, increases the likelihood of the true next chapter, according to the story-generator model.

$$I_{\pi^{\mathcal{G}}}(x, y, a) = [\frac{PPL_{\pi^{\mathcal{G}}}(y|x) - PPL_{\pi^{\mathcal{G}}}(y|x, a)}{PPL_{\pi^{\mathcal{G}}}(y|x)}] \times 100 = [1 - \frac{PPL_{\pi^{\mathcal{G}}}(y|x, a)}{PPL_{\pi^{\mathcal{G}}}(y|x)}] \times 100$$

When investigating the reasoning traces produced in the original work (Gurung & Lapata, 2025) we found significant repetitions in the final plans and in the reasoning traces. We suspect this arises from a kind of mode-collapse, where the policy model learns to generate generic writing advice (e.g. 'the chapter is interesting') instead of making specific claims/plans for the next chapter.

These kinds of reasoning traces would not serve as a useful test-bed for latent reasoning, as they could be easily compressed (and writing advice could be folded into the prompt). We propose a simple addition to the VR-CLI formulation to produce significantly more diverse reasoning traces after training. We call our addition a **contrastive term**, which consists of subtracting a (weighted) improvement term for another chapter from the original improvement.

For each group we randomly select a chapter from a different book (c^r), and get the likelihood improvement of that chapter given the generated plan. We subtract this term to penalize plans that give general advice that could be applicable to any chapter. Because each element within a group receives the same random chapter, we do not find destabilizing effects.

$$R_{\text{old}} = I(SI_i, c_{i+1}, \hat{p})$$
$$R_{\text{contr}} = I(SI_i, c_{i+1}, \hat{p}) - \gamma I(SI_i, c^r, \hat{p})$$

As shown in Table 8, training with this approach produces more diverse plans as measured by type-token ratios. Unless otherwise specified, our NCP results use this contrastive-improvement as the objective of interest for evaluation and training. We provide example plans produced from models trained with and without the contrastive reward in Table 11. Future work could further explore this approach, modifying the weight (we always use $\gamma = 0.5$) and changing the chapter-selection process. For example, down-weighting the likelihood of other chapters in the same book may also have a positive effect by requiring the model to generate plans specific to the current events, but may also discourage the model from referencing true facts about shared characters.

C Plotting performance-cost curves

We plot models by their performance (i.e., reward on the given task) and cost (measured by the generated token length). Our goal is to show that all methods lie somewhere in this tradeoff space, and that LiteReason pushes the Pareto front by achieving comparable performance to non-latent RL

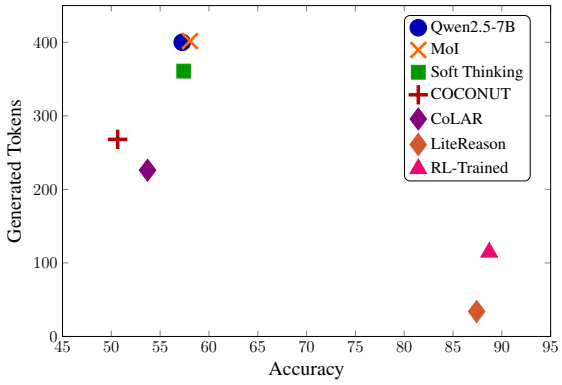

Figure 3: Generated tokens vs Accuracy for the Flawed Fictions benchmark, by model type. Note that the goal is to be to the right (more accurate) and lower (fewer tokens). We find that LiteReason performs significantly more efficiently than the standard RL-Trained model, with only slightly worse performance. Aside from RL-Trained, there is a significant gap in both accuracy (about 20%) and efficiency (about 100 tokens) between LiteReason and the next best method. Thus, we claim our model sits on the same Pareto frontier as the RL-Trained baseline.

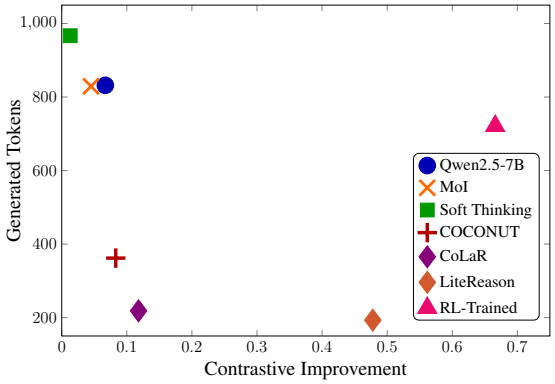

Figure 4: Generated tokens vs Contrastive Improvement for the Next-Chapter Prediction task, by model type. The goal is to be to the right (higher Contrastive Improvement) and lower (fewer tokens). We find that LiteReason performs significantly more efficiently than the standard RL-Trained model, with only slightly worse performance. Aside from RL-Trained, there is a significant gap in both Contrastive Improvement and efficiency between LiteReason and the next best method. Thus, we claim our model sits on the same Pareto frontier as the RL-Trained baseline.

methods while being significantly less costly. Figure 4 shows this tradeoff curve for the NCP task, and Figure 3 for Flawed Fictions.

D Training details and hyper-parameter sweeps

Dataset size and provenance for the two tasks considered (Flawed Fictions and NCP) is presented in Table 7. Example reasoning traces for the tasks are presented in Table 9.

Prompts with and without implicit-prompt instructions are provided for Flawed Fictions in Table 12 and Next-Chapter Prediction in Table 13.

Hyper-parameters were chosen from prior work and initial experiments. Due to their long-context, training on these tasks is very computationally expensive; we hope future work gives more guidance on hyper-parameter selection.

Dataset	# Examples	# Tokens	Source
Flawed Fictions	414	≈ 900	Project Gutenberg
NCP	1,347	≈ 6.5k	Recent Books

Table 7: Size and provenance for the Flawed Fictions (Ahuja et al., 2025) and Next-Chapter Prediction (Gurung & Lapata, 2025) datasets. # Tokens describes the mean number of *input* tokens for each example. # Examples is the total number of datapoints in the dataset, we use the prescribed train/test/val split for NCP and a random 70/15/15 split for Flawed Fictions.

Reward	Reasoning TTR	Final Plan TTR
Default	0.072	0.046
Contrastive	0.087 (+20.8%)	0.136 (+200.2%)

Table 8: Comparing the Type-Token Ratio (TTR) of the reasoning and final plans on the NCP task, with models trained using the default objective (VR-CLI) and our Contrastive objective (Contrastive VR-CLI). Although we observe only a slight increase in reasoning diversity, we see a significant increase in final plan diversity. As shown in the examples in Table 11, our reasoning and final plans are more specific to the current chapter.

Dataset	Example Reasoning Step
GSM8K-Aug (Cobbe et al., 2021)	*The helmet costs $15x 2 = $30.*
ProsQA (Hao et al., 2025)	*Every bompus is a wumpus.*
ProntoQA (Saparov & He, 2023)	*Each vumpus is mean.*
Flawed Fictions (Ahuja et al., 2025)	*The continuity error occurs because the story earlier establishes that the little girl was very poor and had no room to live in or bed to sleep in, but later it states that she returned to her small bed in the shelter with her newfound wealth.*
Next-Chapter-Prediction (Gurung & Lapata, 2025)	*⟨citation⟩Source A (Character Sheet: Rose) says X Rose is rebellious, disobedient, and has a sarcastic sense of humor.⟨/citation⟩, therefore ⟨reasoning⟩Rose will likely continue to challenge authority figures and express her opinions, possibly provoking Miss Wellwood and leading to a confrontation.⟨/reasoning⟩*

Table 9: Example reasoning steps taken from reasoning traces in previously tested datasets (top) compared to our tasks (below). Our tasks involve significantly more varied and complex reasoning steps, which we hypothesize is a more difficult and realistic setting for latent reasoning. Steps for Flawed Fictions and NCP were taken from Qwen 2.5 7B-Instruct. For Flawed Fictions the model compares two story events implicitly, and for NCP the model explicitly cites and reasons over the given Story Information.

For training-free methods we perform a small hyper-parameter sweep across temperature, top-p, and top-k on the validation set, following previous work (Su et al., 2025; Zhang et al., 2025). Training hyper-parameters were selected from the default configurations presented in earlier work (Hao et al., 2025; Tan et al., 2025), and when applicable adjusted to full-parameter fine-tuning.

For higher confidence reported metrics are averages across k runs on the entire test set, where $k = 5$ for NCP and $k = 10$ for Flawed Fictions (as the relatively smaller context length and inexpensive evaluation meant inference on Flawed Fictions was relatively inexpensive). $\pm$SEM scores are run-level estimates of the uncertainty in the test-set mean metric, computed as the standard deviation of the k run-level means divided by $\sqrt{k}$. This is meant to represent the run-level variability induced by LLM sampling.

Reported token counts include the generated latent tokens with the exception of during-training token counts in Table 3, as we did not log every reasoning trace produced during training. Note that COCONUT latent tokens are slightly computationally cheaper than those from CoLaR and LiteReason due to the lack of a latent head or Reasoning Projector. A potential alternative efficiency measurement could be to measure total FLOPs or another memory-utilization metric. We chose average tokens due to significant implementation differences that misrepresent true computational cost. For example, our method has been combined with VLLM which makes generation significantly more efficient than the other trained latent methods. Similarly, there are further optimizations to our method that would make it more comparable to non-latent reasoning: for example we recompute the latent tokens during gradient calculation to avoid significantly re-engineering VLLM's API, which makes our method appear less FLOP efficient than traditional RL despite generating fewer tokens.

Justifications for the chosen hyper-parameters are below.

CoLaR	FF Value	NCP Value
Learning Rate (SFT)	1e-4	1e-4
Learning Rate (RL)	5e-7	5e-7
Compression Factor	5	5
Epochs (SFT)	5	2
Epochs (RL)	30	20
Train batch size	48	64
Max-Gen.-Length	2048	2048
# Samples in group	16	8
Max-Latents	64	64

COCONUT	FF Value	NCP Value
Learning Rate	5e-5	5e-5
Epochs	14	14
Train batch size	48	64
Continuous thought	1	1
Epochs per stage	2	2
Latent stages	10	10

LiteReason	FF Value	NCP Value
Learning Rate (SFT)	1e-4	1e-4
Learning Rate (RL)	5e-7	5e-7
Epochs (SFT)	1	2
Epochs (RL)	30	20
Train batch size	48	64
Max-Gen.-Length	2048	2048
# Samples in groupt	16	8
Token-repl. ratio t_r	0.2	0.2
Sent.-repl. ratio s_r	(10%, 25%)	(10%, 25%)

VR-CLI	NCP Value
Contra. weight γ	0.5

Table 10: Hyper-parameters and their values, adapted from the original work (and modified for fairness). Models were selected by validation performance.

CoLaR: learning rates, batch size, generation length, and samples per prompt were all set the same as our method. Compression factor, max-latents, and other hyper-parameters (e.g. epochs) were left as defaults based on the configurations provided in the codebase.

COCONUT: batch size, generation length, and samples per prompt were all set the same as our method. Continuous thought was set to 1 as Hao et al. (2025) uses this value for logical reasoning, the most similar task. The original work trains on significantly smaller models (GPT-2) and with much larger datasets, and we found significant problems with overfitting and answer-mode-collapse so we adapted other hyper-parameters (e.g. learning rate) to prevent overfitting.

Mixture-of-Inputs: We evaluated performance on the validation set while sweeping the β hyper-parameter $[0.25, 0.5, 1.0, 2.0, 4.0]$ based on the advice presented in Zhuang et al. (2025). We selected the best performing β for test-set evaluation (0.5 for Flawed Fictions, 4.0 for NCP).

E Next-chapter prediction: Human evaluation

We evaluate the chapters on the following dimensions (1) **Plot:** Does the chapter exhibit events and turns that move the plot forward logically? (2) **Creativity:** Does the continuation have engaging characters, themes, and imagery, and avoid overly cliched characters and storylines? (3) **Development:** Does the chapter introduce characters and settings with appropriate levels of detail and complexity? (4) **Language Use:** Is the language varied and rich, exhibiting rhetorical, linguistic,

Reasoning & Plan w/o Contrastive Reward

Character Developments:
⟨citation⟩Source A says X⟨/citation⟩, therefore ⟨reasoning⟩Louise's motivations and emotions become more complex as the story progresses.⟨/reasoning⟩
⟨citation⟩Source A says X⟨/citation⟩, therefore ⟨reasoning⟩Louise begins to question her relationship with John Paul and her feelings for Lee Towson.⟨/reasoning⟩

In the next chapter, Louise's situation is revealed as she remains in a holding cell and is questioned by an officer. The chapter ends with the officer telling Louise of John Paul's assertion that she is involved in Barbara's disappearance. The chapter ends with Louise's thoughts and feelings about her relationship with John Paul and her own secrets. The chapter also concludes with the investigation into Barbara's disappearance and the revelation of John Paul's involvement. The chapter ends with Judy's decision to go home, and the chapter ends with the story near its conclusion. The chapter is marked by themes of family trauma, mental health, and survival, and the chapter ends with the discovery of Barbara's whereabouts. The chapter is a important chapter in the story, as it reveals the truth about Barbara's disappearance and the camp's dark past. The chapter ends with the story near its conclusion, and the chapter ends with the reader eager to know what will happen next. The chapter is a important chapter in the story, as it reveals the truth about Barbara's disappearance and the camp's dark past. The chapter ends with the reader eager to know what will happen next.

Reasoning & Plan w/ Contrastive Reward

Revelation of John Paul's Claim:
⟨citation⟩Next chapter synopsis says the officer tells Louise of John Paul's assertion that Louise is involved in Barbara's disappearance.⟨/citation⟩ Therefore, the officer will reveal that John Paul has accused Louise of being involved in Barbara's disappearance, which will be a pivotal moment in the chapter.

Chapter 44 will begin with Louise being taken to a holding cell, where she will be interrogated by an officer who looks down on her. The officer will reveal that John Paul has accused Louise of being involved in Barbara's disappearance, which will be a significant moment of tension and conflict. Louise will respond defensively and sarcastically, using her wit and intelligence to try to deflect suspicion and protect herself. Throughout the scene, Louise will express her concern for her family, particularly her brother, Jesse, and her determination to protect him from their mother's neglect. The chapter will build on the complex relationships and motivations of the characters, particularly Louise, and set the stage for further revelations and developments in the story.

Table 11: Example reasoning (snippet) and plans taken from models trained with and without the Contrastive Reward on the NCP task. We find that optimizing our contrastive reward produce are more specific to the given Story-Information, and include less generic writing advice. The reasoning is also less formulaic, and obeys the prompts format (with citation and reasoning tags) more faithfully.

and literary devices? (5) **Characters:** Does the chapter feature believable and conceptually consistent characters, including reasonable character arcs and development? (6) **Overall Preference:** Which of the two continuations did you prefer?

We generate chapters with Qwen2.5 7B-Instruct-1M. Relative strengths are shown in Table 2. Across the majority of categories we find our LiteReason method to produce preferred chapters over the other baselines, best matching non-latent RL performance.

E.1 Human annotation details

Annotators were recruited via Prolific, restricted to native English speakers employed in creative writing. In addition to the two test chapters, we show annotators the same Story-Information as the models see, i.e., a global story sketch, previous chapter summaries, character sheets, the previous

Normal Prompt

You are tasked with detecting the presence of continuity errors in a short story. A continuity error occurs when an event or detail in the story contradicts or is incompatible with previously established information about the story's world or characters.

Is there a continuity error in the provided story? Think step by step to answer this question. End your response with \boxed{Yes} if you find a continuity error, and \boxed{No} if you do not find a continuity error. If you respond \boxed{Yes}, you should also provide the lines that introduce the continuity error and the lines from earlier in the story that are contradicted by the error.

Format these lines as follows:
⟨contradicted_lines⟩
[If applicable, quote the lines from earlier in the story that are contradicted by the error]
⟨/contradicted_lines⟩

\boxed{answer}

Here is the story to analyze:

⟨story⟩
{story}
⟨/story⟩

Think carefully and check your work.

Implicit Thought Prompt

You are tasked with detecting the presence of continuity errors in a short story. A continuity error occurs when an event or detail in the story contradicts or is incompatible with previously established information about the story's world or characters.

Is there a continuity error in the provided story? Think step by step to answer this question. End your response with \boxed{Yes} if you find a continuity error, and \boxed{No} if you do not find a continuity error. If you respond \boxed{Yes}, you should also provide the lines that introduce the continuity error and the lines from earlier in the story that are contradicted by the error.

Format these lines as follows:
⟨contradicted_lines⟩
[If applicable, quote the lines from earlier in the story that are contradicted by the error]
⟨/contradicted_lines⟩

\boxed{answer}

In addition to your normal reasoning, you may format your reasoning with implicit thought tags of the following format:

⟨implicit_thought⟩number⟨/implicit_thought⟩
Where number is a number between 1 and 5, and corresponds to the complexity of the thought. When you use this tool, you can skip the sentence that you would have said and move on with your reasoning. This will allow you to think a lot more about the story, while skipping over obvious conclusions/reasoning steps. We recommend doing this in the center of your reasoning, and not at the beginning or end.

Here is the story to analyze:

⟨story⟩
{story}
⟨/story⟩

Think carefully and check your work.

Table 12: Flawed Fictions normal and implicit-thought prompts adapted from the original provided by Ahuja et al. (2025) to improve performance during RL.

chapter, and a next-chapter synopsis. Instructions were adapted from Gurung & Lapata (2025). We paid annotators £14 per three datapoints or an estimated £9.33 per hour.

Thought Prompt

Instructions: You will be given the most recent chapter of the story, a high-level plan of the entire story, a summary of the previously written chapters, character sheets for the three main characters and a brief synopsis of what should happen in the next chapter. You will also be given a chapter header for the next chapter, containing the chapter's title and any other epigraph-type text. You will first reason about the given story and about what should come next. Next, you will write the next chapter of the story.
High-Level Story Summary/Plan: ### {global_story_sketch}
Summary of Already Written Chapters: ### {previous_chapter_summaries}
Character Sheets: ### {character_sheets}
Previous Chapter: ### {previous_chapter}
Next Chapter Synopsis: ### {next_chapter_synopsis}
Next Chapter Header: ### {chapter_header}
Instructions: ### Instructions: Based on the next chapter's synopsis and header, please reason step by step to come up with a more detailed plan for the next chapter. Format your reasoning with "⟨citation⟩source A says X⟨/citation⟩, therefore ⟨reasoning⟩reasoning⟨/reasoning⟩" pairs, where the sources can be the character sheets, the high-level story plan, the previous-chapters summary, the next chapter synopsis, and the previous few chapters. Add and modify your conclusions as you remember more information. End your response with a detailed paragraph explaining your reasoning as to how next chapter will unfold (including plot and character points), beginning this paragraph with "In summary: ".

Implicit Thought Prompt

Instructions: You will be given the most recent chapter of the story, a high-level plan of the entire story, a summary of the previously written chapters, character sheets for the three main characters and a brief synopsis of what should happen in the next chapter. You will also be given a chapter header for the next chapter, containing the chapter's title and any other epigraph-type text. You will first reason about the given story and about what should come next. Next, you will write the next chapter of the story.
High-Level Story Summary/Plan: ### {global_story_sketch}
Summary of Already Written Chapters: ### {previous_chapter_summaries}
Character Sheets: ### {character_sheets}
Previous Chapter: ### {previous_chapter}
Next Chapter Synopsis: ### {next_chapter_synopsis}
Next Chapter Header: ### {chapter_header}
Instructions: ### Instructions: Based on the next chapter's synopsis and header, please reason step by step to come up with a more detailed plan for the next chapter. In addition to your normal reasoning, format your reasoning with two types of tags: 1) ⟨citation⟩A says X⟨/citation⟩, therefore ⟨reasoning⟩B⟨/reasoning⟩ pairs, where the sources A can be the character sheets, the high-level story plan, the previous-chapters summary, the next chapter synopsis, and the previous few chapters. X is the relevant information within A, and B is your reasoning based on A and X. 2) ⟨implicit_thought⟩number⟨/implicit_thought⟩ tags, where number is a number between 1 and 5, and corresponds to the complexity of the thought. When you use this tool, you can skip the sentence that you would have said and move on with your reasoning. This will allow you to think a lot more about the story, while skipping over obvious conclusions/reasoning steps. We recommend doing this in the center of your reasoning, and not at the beginning or end. Add and modify your conclusions as you remember more information. End your response with a detailed paragraph explaining your reasoning as to how next chapter will unfold (including plot and character points), beginning this paragraph with "In summary: ".

Table 13: Next-Chapter Prediction normal prompt and implicit-thought prompt. The prompts are almost exactly the same, but with an additional explanation for how to use the 'implicit thought tags.'