# INTACT: INDUCING LATENT REASONING ON A FROZEN BACKBONE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Latent reasoning replaces natural-language chains of thought with compact continuous "soft-token" representations, reducing the number of intermediate tokens that must be decoded autoregressively. We use a small external encoder to induce latent reasoning in a frozen backbone model. The encoder iteratively maps the backbone's hidden states to new latent blocks, constructing a latent reasoning prefix, after which the frozen model writes its remaining reasoning and its answer as text. Our method, INTACT, trains this encoder through distillation to reproduce the effect of written reasoning steps while staying close to the frozen model's behavior, and then reinforces it on answer correctness. INTACT raises GSM8K accuracy from 75.8% to 79.3% on LLAMA-3.2-3B and from 77.6% to 79.7% on QWEN2.5-3B, and cuts end-to-end latency per problem by 19%. Accuracy improves on all four backbones without updating any backbone parameter, and INTACT outperforms the evaluated latent-reasoning baselines at the 3B and 8B scales. A separate encoder trained on a multi-domain corpus improves the same frozen backbone on twelve held-out mathematics, science and commonsense benchmarks without task-specific adaptation.

## 1 INTRODUCTION

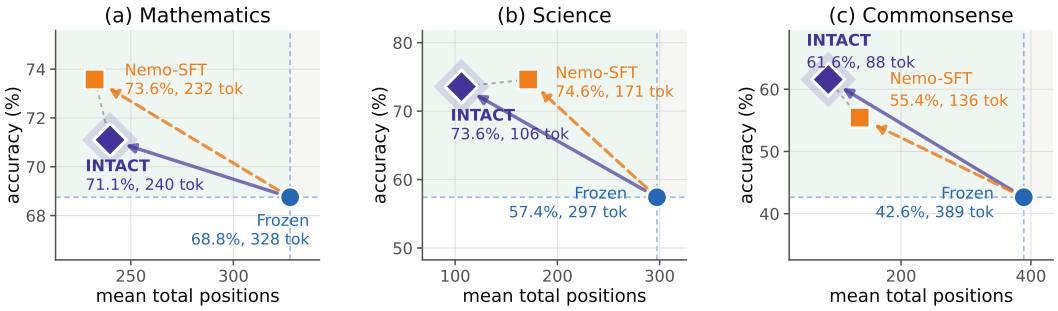


Figure 1: **Accuracy and total positions across three reasoning domains.** The same encoder is evaluated on mathematics, science and commonsense benchmarks. INTACT and Nemo-SFT are both trained on the Llama-Nemotron mixture. INTACT trains only the encoder and leaves LLAMA-3.2-3B frozen, whereas Nemo-SFT updates the language model via supervised fine-tuning. Section 4.4 describes the datasets and evaluation setup.

Language models are increasingly applied to complex problems across domains such as mathematics, code, and commonsense reasoning, where the answer follows from a sequence of dependent steps rather than from recall (Cobbe et al., 2021; Hendrycks et al., 2021b; Jain et al., 2024). In such settings, chain-of-thought prompting draws out the intermediate steps in natural language and substantially improves accuracy (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023). It buys that accuracy with compute, generating more intermediate text as problems grow harder. However, those steps are paid for in the output sequence, since every intermediate token must be decoded autoregressively before the answer appears, and the cost grows with reasoning depth (Chen et al., 2024; Snell et al., 2024).



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

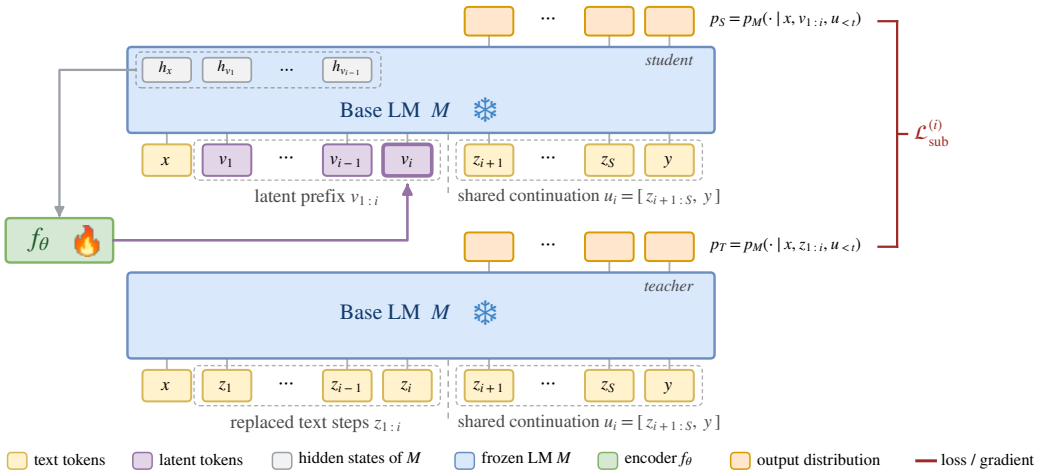


Figure 2: **Stage 1, progressive substitution via distillation.** The language model $M$ is frozen and only the encoder $f_\theta$ is trained. At depth $i$, the encoder produces $\mathbf{v}_i$ from $M$'s states over $\mathbf{x}$ and the preceding blocks $\mathbf{v}_{1:i-1}$, completing the student prefix $[\mathbf{x}, \mathbf{v}_{1:i}]$ against the teacher's $[\mathbf{x}, \mathbf{z}_{1:i}]$. The substitution loss $\mathcal{L}_{\text{sub}}$ distills $p_T$ into $p_S$ over the shared continuation $\mathbf{u}_i$, without directly aligning $\mathbf{v}_j$ and $\mathbf{z}_j$. **Stage 2, latent-policy refinement**, not shown, then refines the same encoder for answer correctness using group-relative policy gradients, leaving $M$ frozen.

*Latent reasoning* proposes to move part of this computation out of language space, replacing some or all of the written steps with continuous vectors that the model consumes directly (Hao et al., 2024; Shen et al., 2025). These vectors may therefore provide compact representations of longer textual traces, and recent work has learned such representations by distilling from an explicit reasoner (Shen et al., 2025; Wei et al., 2026), by multiplexing several reasoning threads into one sequence (Suleymanzade et al., 2026), or by looping a transformer over its own latent state (Fan et al., 2026; Saunshi et al., 2025).

Yet most latent-reasoning methods update the backbone alongside the latent mechanism (Hao et al., 2024; Shen et al., 2025; Fan et al., 2026; Suleymanzade et al., 2026). On capable backbones, such updates risk eroding existing reasoning abilities, including those needed for tasks beyond the fine-tuning data (Xu et al., 2025). Freezing the backbone leaves the original model intact and recoverable, but places the burden on the latent interface to produce inputs the frozen model can use. One response to this constraint is SoftCoT, which derives soft thoughts from the question with a small assistant (Xu et al., 2025). We instead condition a small encoder on the backbone's hidden states over the question and preceding latent blocks. This leads to our central question: *Can we induce latent reasoning on a frozen backbone, improving both its accuracy and its efficiency?*

We propose **INTACT**, illustrated in Figure 2. An encoder reads the frozen language model's final-layer states and produces $K$ latent blocks of $c$ vectors each ($K = 4$ and $c = 4$ in our experiments). During training, the first $j$ blocks are trained to replace the first $j$ reasoning steps. Each block is generated from the question representation together with the frozen model's representation of blocks $1, \ldots, j-1$. At inference, the resulting $Kc$ vectors are inserted after the question in place of the corresponding initial textual reasoning, after which the frozen model generates a solution in its own words.

Training proceeds in two stages. Stage 1 trains each block to reproduce the effect of the written reasoning step it replaces, while a preservation term keeps the latent-conditioned model's behavior close to the frozen model's own. Stage 2 then refines the encoder against answer correctness using group-relative advantages. Across four backbones from 1B to 8B, INTACT improves upon the frozen backbone while reducing the number of generated tokens, and it surpasses the evaluated latent-reasoning methods at the 3B and 8B scales. We summarize our contributions as follows:

1. **Frozen-backbone latent inference.** INTACT prefills the question and constructs a latent prefix over $K$ rounds. In each round, the encoder reads the frozen model's hidden states

and emits a block that the model then processes. The frozen model subsequently generates its own textual reasoning and answer, without further encoder calls.

2. **Two-stage encoder training.** Progressive substitution distills the effect of successive written reasoning steps into latent blocks, while on-policy preservation constrains their effect on generated continuations. A second stage refines the encoder on answer correctness using group-relative policy gradients.
3. **Accuracy with fewer tokens.** Under *mathematics training on GSM8K-Aug*, INTACT improves frozen-model accuracy while generating fewer tokens across four Llama and Qwen backbones from 1B to 8B. It also outperforms the evaluated latent-reasoning baselines at the 3B and 8B scales (Section 4.2).
4. **Multi-domain transfer.** Under *multi-domain training on Llama-Nemotron*, a separate encoder improves accuracy and reduces total positions on twelve held-out benchmarks, including commonsense tasks absent from training and without benchmark-specific adaptation (Figure 1, Section 4.4).

## 2 Related Work

**Chain-of-thought and test-time scaling.** Prompting or training models to produce intermediate steps before answering improves accuracy on hard problems (Wei et al., 2022; Kojima et al., 2022), and the gains grow with the language-space computation spent, whether by aggregating sampled chains (Wang et al., 2023), scaling test-time compute (Snell et al., 2024; Muennighoff et al., 2025), or reinforcement learning that elicits extended traces (DeepSeek-AI et al., 2025). The cost of these tokens has in turn motivated work on overthinking and reasoning efficiency (Chen et al., 2024).

**Latent and continuous reasoning.** A complementary line moves reasoning into non-linguistic computation. Pause and filler tokens add computation without content (Goyal et al., 2024; Pfau et al., 2024), and implicit-CoT methods distill the trace away entirely (Deng et al., 2023; 2024). Coconut feeds last hidden states back as input embeddings, unrolling a chain of continuous thoughts (Hao et al., 2024), which may encode superpositions of parallel search paths (Zhu et al., 2025a). CODI compresses the chain by self-distillation (Shen et al., 2025) and SIM-CoT adds step-level supervision (Wei et al., 2026), while Soft Thinking replaces sampled tokens with probability-weighted mixtures at inference (Zhang et al., 2025), although the benefits of this training-free approach have been contested (Wu et al., 2025). Others make continuous chains trainable with reinforcement learning (Butt et al., 2025), move the reasoning loop inside the network (Saunshi et al., 2025; Geiping et al., 2025), or pretrain with continuous concepts (Tack et al., 2025). Zhu et al. (2025b) survey the area. Many of these methods update the backbone, and most stop the model from writing its own reasoning.

**Compression and frozen-backbone latents.** Text-space compression shortens prompts and traces while keeping the medium fixed (Jiang et al., 2023). Compressed chain-of-thought generates dense contemplation tokens as trace surrogates (Cheng & Van Durme, 2024), Token Assorted mixes discrete latent codes with text (Su et al., 2025), and latent diffusion methods denoise reasoning steps in a learned space that is co-trained for noise tolerance (Kang et al., 2026; Li et al., 2022). SoftCoT prepends soft tokens from a small assistant model to a frozen backbone and still generates language-space reasoning afterwards (Xu et al., 2025). Appendix A compares INTACT with individual prior methods.

## 3 Method

**Notation.** Let $M$ be a language model with vocabulary $\mathcal{V}$ and input-embedding dimension $d_M$, and let $\mathcal{V}^*$ denote sequences of tokens in $\mathcal{V}$. For a token string $\mathbf{x} \in \mathcal{V}^*$, let $E(\mathbf{x})$ denote the corresponding sequence of token embeddings. We consider inputs to $M$ that mix a token sequence $\mathbf{x}$ with a sequence $\mathbf{v}$ of non-token vectors, and we write such inputs in embedding space, so that the concatenation of the two is $[E(\mathbf{x}), \mathbf{v}]$. We write $H(\cdot)$ for the final-layer hidden states $M$ produces on a mixed input of this kind, so $\mathbf{h} = H([E(\mathbf{x}), \mathbf{v}])$ denotes its states over the whole sequence. Where the embedding map is clear from context we abbreviate $[E(\mathbf{x}), \mathbf{v}]$ as $[\mathbf{x}, \mathbf{v}]$, and we write

**Algorithm 1** INTACT inference. The language model $M$ is frozen and the encoder $f_\theta$ is fixed.

**Require:** question $\mathbf{x}$; frozen LM $M$; trained encoder $f_\theta$; $K{=}4$ blocks of $c{=}4$ vectors
$H(\cdot)$ denotes the final-layer states $M$ produces on an embedding-space input, as in Section 3

**for** $j = 1, \ldots, K$ **do**
  $\mathbf{v}_j \leftarrow f_\theta^{(j)}\big(H([E(\mathbf{x}), \mathbf{v}_{1:j-1}])\big)$ ▷ one block per call, $c$ vectors at a time
**end for**
$\hat{\mathbf{y}} \sim p_M\big(\cdot \mid [E(\mathbf{x}), \mathbf{v}_{1:K}]\big)$ ▷ no further encoder calls
**return** $\mathrm{Ans}(\hat{\mathbf{y}})$

conditioning on such an input as $p_M(\cdot \mid \mathbf{x}, \mathbf{v})$. A training example is a triple $(\mathbf{x}, \mathbf{z}_{1:S}, \mathbf{y})$, where $\mathbf{x}$ is a question, $\mathbf{z}_{1:S} = (\mathbf{z}_1, \ldots, \mathbf{z}_S)$ is a reference reasoning trace of $S$ steps with $\mathbf{z}_i \in \mathcal{V}^{n_i}$, and $\mathbf{y}$ is the final answer. At inference, the unmodified backbone generates a continuation $\hat{\mathbf{y}} \sim p_M(\cdot \mid \mathbf{x})$ containing its own reasoning and a final answer $\mathrm{Ans}(\hat{\mathbf{y}})$, requiring an autoregressive decoding step for every token in it. Table 5 collects the remaining symbols.

### 3.1 Latent Encoder $f_\theta$

The latent encoder $f_\theta$ maps a sequence of hidden states $\mathbf{h}$ to a block of $c$ continuous vectors. Vectors within a block are generated jointly in a single encoder pass, whereas blocks are constructed sequentially. To produce block $j$, the encoder reads the frozen model's final-layer states over the question and the preceding latent blocks:

$$\mathbf{v}_j = f_\theta^{(j)}\big(H([E(\mathbf{x}), \mathbf{v}_{1:j-1}])\big) \in \mathbb{R}^{c \times d_M}.$$

For $j = 1$, the input consists only of the question states, $H(E(\mathbf{x}))$. After each block is appended, the frozen model processes the extended prefix, providing the hidden states used to generate the next block. This recurrence is mediated by the frozen model's representations of the preceding blocks, rather than by feeding their raw vectors directly back to the encoder. The encoder therefore builds the latent prefix from representations supplied by $M$, without learning a separate question encoder.

*Example.* For a 60-token question, the frozen model supplies the states $H(E(\mathbf{x}))$ to the encoder, which returns the first block $\mathbf{v}_1 = f_\theta^{(1)}(H(E(\mathbf{x})))$ of $c = 4$ vectors. These are appended to the input, so the model reads $[E(\mathbf{x}), \mathbf{v}_1]$ and produces $H([E(\mathbf{x}), \mathbf{v}_1])$. The encoder reads the sequence of 64 hidden states to produce $\mathbf{v}_2 = f_\theta^{(2)}(H([E(\mathbf{x}), \mathbf{v}_1]))$, and so on:

$$\text{Prefix positions:} \quad 60 \xrightarrow{+\mathbf{v}_1} 64 \xrightarrow{+\mathbf{v}_2} 68 \xrightarrow{+\mathbf{v}_3} 72 \xrightarrow{+\mathbf{v}_4} 76.$$

After $K = 4$ blocks the prefix holds 76 positions, 60 question tokens and 16 latent vectors, and the model generates its reasoning and answer with no further encoder calls.

**Encoder architecture.** The encoder first projects the context states from $d_M$ to a hidden dimension of $d = 1024$. It then applies six Transformer layers, each consisting of cross-attention from the $c$ queries to the context, self-attention among the queries, and a feed-forward network, with pre-normalization and residual connections throughout. A final linear layer projects each output to $d_M$. Each output vector is normalized and rescaled by a shared learned scalar. All parameters are shared across blocks except for the queries $q_j$.

**Inference.** Given a question $\mathbf{x}$, the encoder is invoked $K$ times, each call appending a block to the input the frozen model reads. Once the latent prefix is constructed, $M$ generates the textual reasoning and a final answer autoregressively from $[E(\mathbf{x}), \mathbf{v}_{1:K}]$, without further encoder calls. Algorithm 1 states the procedure.

### 3.2 Training

Training proceeds in two stages, neither of which updates the language model. Every parameter of $M$ remains fixed, and although gradients pass through it, only $\theta$ is updated, which separates learning the latent interface from relearning the reasoning behavior of the backbone. Stage 1 trains through

distillation to replace written reasoning steps with latent blocks, and Stage 2 then refines the same encoder against answer correctness. Algorithm 2 summarizes both stages.

**Stage 1: progressive substitution.** Stage 1 trains successive latent blocks to reproduce the effect of corresponding written reasoning steps. At substitution depth $i$, the *teacher context* contains the first $i$ reference steps, $C_i^{\mathrm{T}} = [\mathbf{x}, \mathbf{z}_{1:i}]$, while the *student context* replaces them with latent blocks, $C_i^{\mathrm{S}} = [\mathbf{x}, \mathbf{v}_{1:i}]$. Both are followed by the same continuation $\mathbf{u}_i = [\mathbf{z}_{i+1:S}, \mathbf{y}]$. For two contexts $A$ and $B$, we define the continuation-level behavioural divergence

$$D(A \| B; \mathbf{u}) = \frac{1}{|\mathbf{u}|} \sum_t \mathrm{KL}(p_M(\cdot \mid A, \mathbf{u}_{<t}) \,\|\, p_M(\cdot \mid B, \mathbf{u}_{<t})) \,.$$

The substitution loss at depth $i$ is $\mathcal{L}_{\mathrm{sub}}^{(i)} = D(C_i^{\mathrm{T}} \| C_i^{\mathrm{S}}; \mathbf{u}_i)$, self-distillation through one frozen model, with the reference steps supplying the behavior to reproduce and both distributions evaluated under teacher forcing on the shared continuation, whose positions are aligned across the two prefixes (Appendix C.1). We optimize depths $i = 1, \ldots, K$ in order, using the substitution loss only for examples with $S > i$, and take one encoder update at each depth. After every update, $\mathbf{v}_{1:i}$ is regenerated with the new parameters before constructing depth $i + 1$. Gradients pass through the preceding blocks and the fixed language model, but only the shared encoder parameters $\theta$ are updated.

**On-policy preservation.** Because $\mathcal{L}_{\mathrm{sub}}$ is teacher-forced, it cannot see failures that appear only when the model generates freely. We therefore sample a rollout under the complete prefix, $\hat{\mathbf{y}} \sim p_M(\cdot \mid \mathbf{x}, \mathbf{v}_{1:K})$, and penalize the intervention's effect along it with $\mathcal{L}_{\mathrm{pre}} = D(\mathbf{x} \| [\mathbf{x}, \mathbf{v}_{1:K}]; \hat{\mathbf{y}})$. The sampled continuation is held fixed, and the unsteered model supplies the target distribution, regularizing the latent-conditioned model on prefixes encountered during free generation. Preservation applies only at $i = K$, so the Stage 1 loss of Algorithm 2 is $\mathcal{L}_{\mathrm{S1}}^{(i)} = \mathcal{L}_{\mathrm{sub}}^{(i)} + \mathbf{1}[i = K]\, \beta \mathcal{L}_{\mathrm{pre}}$.

**Stage 2: latent-policy refinement.** Stage 1 learns from the behavior induced by reference traces. Stage 2 instead refines the encoder against answer correctness. The encoder and frozen model together define $\pi_\theta(\hat{\mathbf{y}} \mid \mathbf{x}) = p_M(\hat{\mathbf{y}} \mid \mathbf{x}, \mathbf{v}_{1:K})$ with the $K$ blocks $\mathbf{v}_{1:K}$ built sequentially by the encoder $f_\theta$, which alone is updated. For each question, we sample $G$ completions $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^G$ and assign the binary correctness reward $r^g = R(\hat{\mathbf{y}}^g, \mathbf{y})$, where $R$ indicates whether the extracted answer matches the reference under the answer-comparison protocol of Appendix C.4. We standardize rewards within the group to obtain $A^g = (r^g - \bar{r})/(\sigma_r + \epsilon)$ and minimize the per-sequence-normalized policy loss

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{G} \sum_{g=1}^{G} A^g \frac{1}{|\hat{\mathbf{y}}^g|} \sum_t \log p_M(\hat{y}_t^g \mid \mathbf{x}, \mathbf{v}_{1:K}, \hat{\mathbf{y}}_{<t}^g).$$

Groups in which every completion receives the same reward have zero advantage and make no update. The Stage 2 block of Algorithm 2 summarizes this update.

# 4 Experiments

## 4.1 Setup

**Datasets and backbones.** We report two settings, which train *separate* encoders and are never mixed. **Mathematics training on GSM8K-Aug** follows prior latent-reasoning work and uses GSM8K-Aug (Deng et al., 2023), 385K examples whose solutions are sequences of compact arithmetic steps; it is evaluated in domain on the GSM8K test set (Cobbe et al., 2021) and out of domain on GSM-Hard, SVAMP and MultiArith, over which we report an unweighted macro-average. **Multi-domain training on Llama-Nemotron** instead uses the Llama-Nemotron post-training corpus (Bercovich et al., 2025), approximately $64\%$ mathematics, $21\%$ science and $14\%$ code, and is evaluated on twelve held-out benchmarks. We use four backbones: LLAMA-3.2-1B-INSTRUCT, LLAMA-3.2-3B-INSTRUCT, LLAMA-3.1-8B-INSTRUCT (Grattafiori et al., 2024), and the base QWEN2.5-3B (Yang et al., 2024). An encoder reads one backbone's hidden states and is trained for it alone, so each setting yields one encoder *per backbone*. Appendix C.2 gives the corpus selection and the segmentation into steps, and Appendix E the twelve evaluation benchmarks.

**Algorithm 2** INTACT training. The language model $M$ is frozen throughout; only $\theta$ is learned.

**Require:** example $(\mathbf{x}, \mathbf{z}_{1:S}, \mathbf{y})$; frozen LM $M$; encoder $f_\theta$; $K{=}4$ blocks of $c{=}4$ vectors
$H(\cdot)$ denotes the final-layer states $M$ produces on an embedding-space input, as in Section 3
$D(A\|B;\mathbf{u}) = \frac{1}{|\mathbf{u}|}\sum_t \mathrm{KL}\big(p_M(\cdot|A,\mathbf{u}_{<t})\|p_M(\cdot|B,\mathbf{u}_{<t})\big)$
**Stage 1: distil the chain into latents**
**for** minibatch, then $i = 1, \ldots, K$ **do** ▷ $\mathcal{L}_{\mathrm{sub}}$ uses examples with $S > i$
  $\mathbf{v}_j \leftarrow f_\theta^{(j)}\big(H([E(\mathbf{x}), \mathbf{v}_{1:j-1}])\big), \;\; j = 1, \ldots, i$
  $\mathcal{L} \leftarrow D\big(\mathbf{x}, \mathbf{z}_{1:i} \,\|\, \mathbf{x}, \mathbf{v}_{1:i}\,;\, \mathbf{z}_{i+1:S}, \mathbf{y}\big)$ ▷ $\mathcal{L}_{\mathrm{sub}}$: $i$ latent blocks replace $i$ text steps
  **if** $i = K$ **then**
    $\hat{\mathbf{y}} \sim p_M(\cdot \mid \mathbf{x}, \mathbf{v}_{1:K})$; $\quad \mathcal{L} \leftarrow \mathcal{L} + \beta\, D\big(\mathbf{x}\|\mathbf{x}, \mathbf{v}_{1:K}; \hat{\mathbf{y}}\big)$ ▷ $\mathcal{L}_{\mathrm{pre}}$: match unsteered $M$
  **end if**
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ ▷ backprop through fixed $M$; update $\theta$ only
**end for**
**Stage 2: reinforce with answer reward**
**for** minibatch **do**
  $\mathbf{v}_j \leftarrow f_\theta^{(j)}\big(H([E(\mathbf{x}), \mathbf{v}_{1:j-1}])\big), \;\; j = 1, \ldots, K$; $\quad$ sample $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^G \sim p_M(\cdot \mid \mathbf{x}, \mathbf{v}_{1:K})$
  $r^g \leftarrow R(\hat{\mathbf{y}}^g, \mathbf{y}), \quad A^g \leftarrow (r^g - \bar{r})/(\sigma_r + \epsilon)$ ▷ group-relative advantage
  $\theta \leftarrow \theta + \eta \frac{1}{G} \sum_g \frac{A^g}{|\hat{\mathbf{y}}^g|} \sum_t \nabla_\theta \log p_M\big(\hat{y}_t^g \mid \mathbf{x}, \mathbf{v}_{1:K}, \hat{\mathbf{y}}_{<t}^g\big)$ ▷ per-sequence mean
**end for**

**Methods.** We report INTACT with $K = 4$ blocks of $c = 4$ vectors on every backbone, using the same encoder architecture and objectives throughout and keeping the language model frozen. We compare against the unchanged **Frozen LM**, and against three supervised baselines that update the language model and so measure what the same supervision achieves when changing the backbone is allowed. **No-CoT** trains on the answer alone. **Aug-SFT** trains on GSM8K-Aug, whose steps are compact symbolic traces, primarily calculator expressions. **Nemo-SFT** instead trains on the Llama-Nemotron mixture, whose steps are complete natural-language worked solutions. We further compare with **CODI** (Shen et al., 2025), **LOTUS** (Fan et al., 2026), and **MUX** (Suleymanzade et al., 2026), which replace written reasoning with continuous or recurrent computation but likewise update the language model. We refer to Appendix C for further experimental details.

### 4.2 Main Results

**Against the frozen backbone.** Table 1 reports accuracy at the 768-token cap, where both stages improve GSM8K and out-of-domain accuracy over the frozen backbone on all four models. Stage 2 further improves six of the eight Stage 1 scores; it is 0.3 points lower on 3B out-of-domain accuracy and 0.1 points lower on 8B GSM8K. The largest Stage 1 gains occur at 8B, where frozen-model generations often reach the cap. Appendix Table 7 reports the complete budget sweep and seed variation.

Table 1: Accuracy (%) at a 768-token generation cap across four backbones. Both INTACT stages are trained on GSM8K-Aug. *OOD* is the average on SVAMP, GSM-Hard and MultiArith.

| | Frozen | | INTACT Stage 1 | | INTACT Stage 2 | |
|---|---|---|---|---|---|---|
| | GSM8K↑ | OOD↑ | GSM8K↑ | OOD↑ | GSM8K↑ | OOD↑ |
| Llama-1B | 44.0 | 47.8 | 44.2 | 51.3 | **48.7** | **54.5** |
| Llama-3B | 75.8 | 66.5 | 77.9 | **69.1** | **79.3** | 68.8 |
| Llama-8B | 72.3 | 53.9 | **83.7** | 71.1 | 83.6 | **72.1** |
| Qwen-3B | 77.6 | 74.6 | 78.0 | 75.0 | **79.7** | **75.1** |

**Comparison with latent methods.** Table 2 reports accuracy on the four backbones. At Llama-3.2-1B, INTACT remains below Aug-SFT and the latent baselines, reaching 48.7 on GSM8K and 54.5 out of domain, compared with 61.6 and 60.5 for Aug-SFT. This ordering reverses at 3B, where INTACT reaches 79.3 and 68.8, outperforming the strongest published latent baseline by 9.3 and 4.7 points and Aug-SFT by 7.8 and 6.7 points. The advantage over Aug-SFT persists at 8B (83.6/72.1 versus 74.2/67.0) and on Qwen2.5-3B (79.7/75.1 versus 74.8/69.5), showing that it extends beyond the Llama family. Aug-SFT is strongest at 1B, whereas Nemo-SFT leads on GSM8K and the out-of-domain average at 3B and above, exceeding Stage 2 by 2.1–3.7 and 1.3–1.7 points, respectively. Thus, INTACT does not uniformly outperform supervised fine-tuning: it improves the frozen model and achieves the strongest GSM8K and average results among the evaluated latent methods at 3B and above, without modifying the language model.

Table 2: Accuracy (%) on four mathematics benchmarks. Our Frozen, SFT, and INTACT evaluations use a 768-token cap. CODI, MUX, and LOTUS use their source protocols, including reproductions. † and ‡ mark results quoted from MUX (Suleymanzade et al., 2026) and LOTUS (Fan et al., 2026), respectively. *Average* is the mean of SVAMP, GSM-Hard, and MultiArith. Shading marks INTACT, with ± denoting standard deviation over three encoder seeds. Best and second-best latent methods are **bold** and underlined. A dash marks a configuration for which no comparable trained checkpoint is available at that scale.

| Backbone | Method | GSM8K ↑ | SVAMP ↑ | GSM-Hard ↑ | MultiArith ↑ | Average ↑ excl. GSM8K |
|---|---|---|---|---|---|---|
| LLAMA 3.2-1B | No-CoT† | 30.9 | 44.1 | 7.1 | 70.9 | 40.7 |
| | Aug-SFT† | 61.6 | 66.7 | 15.6 | 99.3 | 60.5 |
| | Nemo-SFT | 53.1 | 59.3 | 15.0 | 92.8 | 55.7 |
| | LOTUS‡ | **57.3** | 60.9 | 12.7 | 98.3 | 57.3 |
| | MUX† | 56.7 | **63.6** | 13.0 | **98.5** | **58.4** |
| | CODI† | 55.6 | 61.1 | 12.8 | 96.1 | 56.7 |
| | INTACT Stage 1 | 44.2 ± 1.0 | 55.3 ± 2.3 | 13.1 ± 0.6 | 85.6 ± 0.1 | 51.3 ± 1.0 |
| | INTACT Stage 2 | 48.7 ± 1.3 | 59.8 ± 1.3 | **13.7** ± 0.3 | 89.9 ± 0.2 | 54.5 ± 0.4 |
| LLAMA 3.2-3B | No-CoT† | 38.3 | 52.9 | 9.5 | 88.7 | 50.4 |
| | Aug-SFT† | 71.5 | 71.0 | 17.0 | 98.3 | 62.1 |
| | Nemo-SFT | 82.9 | 83.5 | 30.3 | 97.8 | 70.5 |
| | LOTUS‡ | 70.0 | 75.7 | 16.0 | 99.9 | 63.9 |
| | MUX† | 65.0 | 77.1 | 15.2 | **100.0** | 64.1 |
| | CODI† | 60.8 | 73.3 | 14.3 | 98.7 | 62.1 |
| | INTACT Stage 1 | 77.9 ± 0.3 | 81.1 ± 0.6 | **27.9** ± 0.2 | 98.3 ± 0.5 | **69.1** ± 0.3 |
| | INTACT Stage 2 | **79.3** ± 0.4 | **81.8** ± 0.9 | 27.0 ± 0.7 | 97.7 ± 0.3 | 68.8 ± 0.2 |
| LLAMA 3.1-8B | No-CoT | 26.5 | 57.2 | 7.5 | 79.8 | 48.2 |
| | Aug-SFT | 74.2 | 77.3 | 26.5 | 97.3 | 67.0 |
| | Nemo-SFT | 87.3 | 86.3 | 35.9 | 98.0 | 73.4 |
| | LOTUS | – | – | – | – | – |
| | MUX | 65.4 | 77.8 | 16.1 | **100.0** | 64.6 |
| | CODI | – | – | – | – | – |
| | INTACT Stage 1 | **83.7** ± 0.6 | 82.8 ± 1.6 | 33.2 ± 0.7 | 97.4 ± 1.1 | 71.1 ± 1.1 |
| | INTACT Stage 2 | 83.6 ± 0.7 | **84.6** ± 1.1 | **33.4** ± 1.3 | 98.4 ± 0.6 | **72.1** ± 0.7 |
| QWEN 2.5-3B | No-CoT | 27.6 | 52.5 | 8.7 | 63.7 | 41.6 |
| | Aug-SFT | 74.8 | 76.2 | 34.6 | 97.7 | 69.5 |
| | Nemo-SFT | 81.8 | 86.8 | 45.3 | 97.5 | 76.5 |
| | LOTUS | 66.7 | 71.6 | 17.9 | **100.0** | 63.2 |
| | MUX | – | – | – | – | – |
| | CODI | – | – | – | – | – |
| | INTACT Stage 1 | 78.0 ± 2.0 | **87.1** ± 0.4 | 40.8 ± 2.2 | 97.0 ± 0.6 | 75.0 ± 1.0 |
| | INTACT Stage 2 | **79.7** ± 0.5 | 85.5 ± 0.1 | **41.4** ± 1.1 | 98.5 ± 0.6 | **75.1** ± 0.5 |

**Token efficiency.** Fixed-budget accuracy does not show how much generation a system uses before stopping. Figure 3 shows the accuracy–efficiency trade-off on LLAMA-3.2-3B. Aug-SFT reaches 71.5 accuracy at 34 total positions, whereas INTACT reaches 79.3 at 211: 195 generated tokens plus its 16 latent positions. Trained on longer natural-language traces, Nemo-SFT reaches 82.9 at 234, illustrating how the SFT operating point depends on its training corpus. Across generation budgets (Appendix D.2), Aug-SFT improves earlier but saturates sooner, while INTACT reaches a stronger endpoint and Stage 2 produces a clearer leftward shift. Thus, INTACT improves the accuracy–efficiency trade-off without targeting the most aggressive compression regime. On GSM8K, Stage 2 also reduces end-to-end latency by 19% relative to the frozen backbone at batch 1 on one H100 NVL, including latent-prefix construction (Appendix D.3).

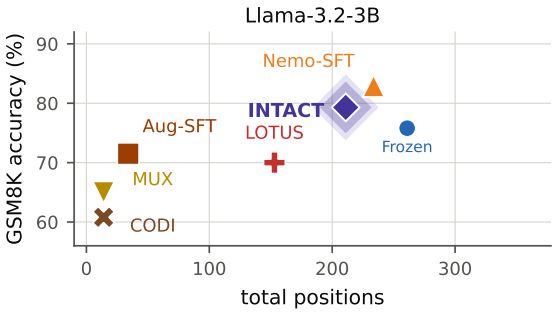


Figure 3: GSM8K accuracy versus total positions on LLAMA-3.2-3B. Frozen, INTACT, and Nemo-SFT use a 768-token cap. CODI, MUX, and LOTUS follow their source protocols. Counts include latent positions where applicable.

### 4.3 ABLATIONS

Each ablation isolates one design decision and holds the rest of the recipe fixed. The preservation and capacity ablations use QWEN2.5-3B, while the substitution-depth sweep uses LLAMA-3.1-8B. All are scored under the protocol of Section 4.1 with a 768-token cap.

| | GSM8K↑ | OOD↑ | Tok.↓ |
|---|---|---|---|
| Frozen backbone | 77.6 | 74.6 | 273 |
| *Preservation anchor $u$* | | | |
| None ($\beta = 0$) | 42.9 | 54.5 | 80 |
| Gold chain | 74.0 | 69.1 | 240 |
| Frozen rollout | 76.2 | 76.2 | 309 |
| Student rollout | 78.8 | 75.5 | 246 |
| *Block width $c$ (at $K$=4)* | | | |
| $c = 1$ (4 pos.) | 73.7 | 72.7 | 207 |
| $c = 2$ (8 pos.) | 78.0 | 76.4 | 285 |
| $c = 4$ (16 pos.) | 78.8 | 75.5 | 246 |
| $c = 8$ (32 pos.) | 73.7 | 72.2 | 316 |

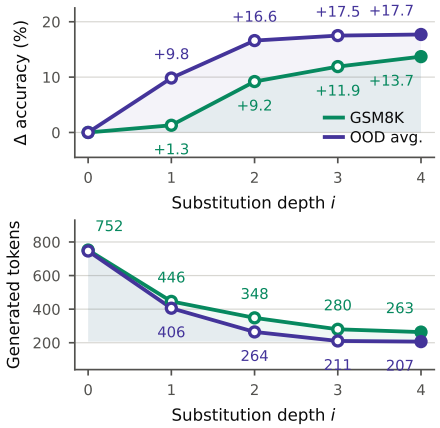


Figure 4: **Left:** ablations on QWEN2.5-3B; each block varies one design decision, shaded rows indicate settings used by INTACT. **Right:** substitution depth sweep on LLAMA-3.1-8B; at depth $i$ only the first $i$ of $K$ blocks are supplied, so $i = 0$ is the frozen backbone and $i = K$ the full Stage 1 prefix, for one encoder seed. *OOD* is the macro-average over the three held-out benchmarks.

**Preservation anchor.** Because $\mathcal{L}_{\text{sub}}$ is scored on a reference continuation, it does not constrain the model on its own free generations. The variants below differ only in the continuation $\mathbf{u}$ on which $\mathcal{L}_{\text{pre}}(\mathbf{u})$ is scored, and we compare omitting the term with three choices of $\mathbf{u}$.

- **Gold chain.** $\mathbf{u} = [\mathbf{z}_{1:S}, \mathbf{y}]$, the written trace itself, so $\mathbf{v}_{1:K}$ is scored while the model is given the very steps it replaces.
- **Frozen rollout.** $\mathbf{u} \sim p_M(\cdot \mid \mathbf{x})$, free generation from the backbone without the latent prefix.
- **Student rollout.** $\mathbf{u} \sim p_M(\cdot \mid \mathbf{x}, \mathbf{v}_{1:K})$, free generation from the latent-conditioned model, the continuations it actually produces.

In Figure 4 (left), omitting preservation collapses the output to 80 tokens, so its apparent efficiency is degraded generation rather than compression. The gold chain stabilizes training but stays below the backbone on both accuracy measures, as its teacher-forced context would predict. A frozen rollout removes that teacher forcing and gives the best out-of-domain accuracy of the four anchors, but remains off-policy for the latent-conditioned model and lengthens generation to 309 tokens. Only the student rollout improves on the backbone in accuracy and in length at once.

**Substitution depth.** Stage 1 supervises every available depth, so each block is learned in the context created by its predecessors. The sweep holds the encoder fixed and varies only the number of blocks supplied at inference. On LLAMA-3.1-8B (Figure 4, right), increasing the prefix from $i = 0$ to $i = K$ raises GSM8K accuracy by 13.7 points and the out-of-domain average by 17.7, while reducing generation almost threefold. Accuracy therefore depends on the depth supplied, not only on the presence of a latent prefix.

**Latent capacity.** The available substitution depth is $\min(K, S-1)$, and only 8% of GSM8K-Aug chains support five steps. We therefore fix $K = 4$ and vary the block width $c$, giving $4c$ latent positions. Performance is non-monotonic: $c = 1$ provides insufficient capacity, while $c = 8$ reduces accuracy and lengthens generation. Among the narrower variants, $c = 1$ saves tokens at the cost of accuracy, whereas $c = 2$ improves accuracy but generates more tokens than the backbone. Only $c = 4$ improves on the frozen backbone while also generating fewer tokens, providing the best accuracy–efficiency balance.

### 4.4 One Encoder Across Diverse Reasoning Domains

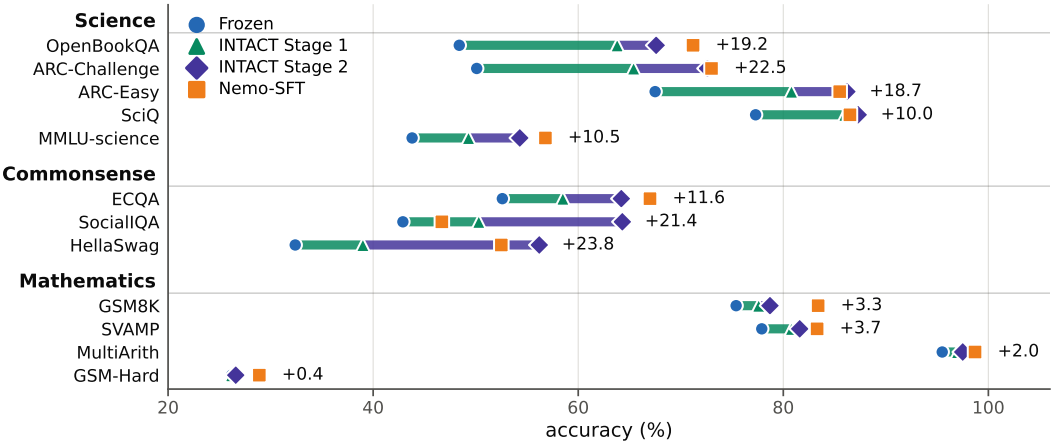


Figure 5: **One encoder across twelve benchmarks.** Accuracy on LLAMA-3.2-3B across mathematics, science, and commonsense. Each benchmark line connects the frozen backbone to Stage 2, with Stage 1 marking the intermediate result. Both Nemo-SFT and INTACT are trained on the Llama-Nemotron mixture. The same INTACT encoder is used across all domains/benchmarks.

**Setup.** We next test whether a single latent encoder can improve reasoning across domains beyond GSM8K-Aug. We train on the mathematics, science, and code solutions in Llama-Nemotron and evaluate the same encoder, without benchmark-specific adaptation, on twelve mathematics, science, and commonsense benchmarks. Commonsense evaluation tests transfer beyond the domains explicitly represented in training. Every system in this setting is scored at the 2,048-token budget of Appendix C.4, which the long natural-language solutions of this mixture require.

**Results.** Figure 5 shows that one encoder improves the frozen backbone on all twelve benchmarks. Gains are largest in science and commonsense, reaching 22.5 points on ARC-Challenge and 23.8 on HellaSwag, while mathematics gains range from 0.4 to 3.7 points. The commonsense result is particularly notable because that domain is absent from training. The improvements are therefore largest where the frozen backbone is weakest, with the exception of GSM-Hard, on which no system improves substantially. Compared with Nemo-SFT, INTACT is nearly tied on science (1.0 point lower) and trails on mathematics (2.5 points), but leads on commonsense (6.2 points) in domain-average accuracy, while leaving the backbone frozen.

## 5 Conclusion

We studied whether latent reasoning can improve a capable language model without updating its backbone. INTACT trains a small encoder that reads the frozen model's own hidden states to reproduce the effect of early written reasoning steps, first through distillation and then through answer-reward refinement, while the language model remains frozen. Across four backbones, it improves accuracy and reduces generation relative to the frozen model, and it outperforms the evaluated latent-reasoning baselines on larger backbones. One encoder trained on mathematics, science, and code also improves the same frozen backbone on held-out mathematics, science, and commonsense benchmarks without task-specific adaptation. Together, these results show that latent reasoning can complement a capable model's existing reasoning, rather than depend on rewriting its weights or maximizing compression.

**Limitations.** One limitation is that the latent blocks cannot be directly interpreted as the written steps they replace. Another is that the backbone still decodes autoregressively, so shorter generations do not necessarily produce proportional wall-clock savings (Appendix D.3). INTACT also still retains language-space reasoning, so it remains more expensive than methods that compress reasoning into a handful of tokens.

## REPRODUCIBILITY STATEMENT

The encoder and both training stages are specified in Section 3, with inference in Algorithm 1 and training in Algorithm 2. Appendix C.1 gives the Stage 1 and Stage 2 settings, including the position alignment between teacher and student prefixes, the checkpoint-selection rule, and the per-backbone hyperparameters of Table 6; Appendix C.2 does the same for the encoder trained on the Llama-Nemotron mixture, including how its solutions are segmented into steps. Appendix C.4 fixes the evaluation protocol shared by every number we report: the prompt format, the decoding parameters, the generation budget for each training setting, and the answer-extraction and correctness rules. Appendix C.3 records how each baseline was trained or, where a result is quoted, which table it is taken from. Appendix E lists the training corpora and the exact evaluation splits and sizes. Table 7 reports the mean and sample standard deviation over three encoder seeds at every budget, so the run-to-run spread of our main results can be read directly. Source code for the encoder, both training stages, and the evaluation harness is included in the supplementary materials.

## AI USE STATEMENT

In this work, we used generative AI tools to implement methods, to propose and refine hypotheses, to design and give feedback on experiments and ablations, to clean and reformat training data, and to interpret results. We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide ingredients for or assist in the writing of proofs, or to assist with translation. Support for qualitative and thematic data analysis and the formulation of survey or interview questions are not applicable to this work.

Additionally, we used generative AI tools for brainstorming, for searching for and summarizing existing literature, for identifying relevant related work, for writing and editing the code that trains and evaluates the models, for creating and revising the figures, and for editing this paper to improve its readability and structure.

We have reviewed all AI-assisted work. Every number reported here was produced by code the authors read and ran, and each figure and table was checked against the evaluation logs it is generated from. AI-suggested related work was verified by reading the cited papers rather than accepting the summaries. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

## REFERENCES

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for CommonsenseQA: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3050–3065, 2021.

Akhiad Bercovich, Itay Levy, Izik Golan, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*, 2025.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+3=? On the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 10764–10799, 2023.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

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. *arXiv preprint arXiv:2403.07974*, 2024.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yi-An Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. In *International Conference on Learning Representations*, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

Anna Kuzina, Maciej Pióro, and Babak Ehteshami Bejnordi. KaVa: Latent reasoning via compressed KV-cache distillation. In *International Conference on Learning Representations*, 2026.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2381–2391, 2018.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 2080–2094, 2021.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

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 *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

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. *arXiv preprint arXiv:2502.03275*, 2025.

Ayhan Suleymanzade, Halil Alperen Gozeten, Michael Bronstein, İsmail İlkan Ceylan, and Jinwoo Kim. MUX: Continuous reasoning via multiplexed tokens. *arXiv preprint arXiv:2607.18264*, 2026.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. LLM pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, 2026.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Workshop on Noisy User-generated Text*, 2017.

Chünhung Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. LLMs are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *arXiv preprint arXiv:2508.03440*, 2025.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025a.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025b.

# Appendix


**A Extended Related Work** **15**

A.1 From Written to Latent Reasoning . . . . . 15

A.2 Positioning of INTACT . . . . . 15

**B Notation** **17**

**C Experimental Details** **17**

C.1 INTACT Training . . . . . 17

C.2 Diverse-Domain Training and Evaluation . . . . . 19

C.3 Baselines . . . . . 19

C.4 Evaluation Protocol . . . . . 20

**D Extended Results** **20**

D.1 INTACT and the Frozen Backbone Across Budgets . . . . . 20

D.2 Accuracy–Efficiency Operating Points . . . . . 22

D.3 Efficiency . . . . . 24

D.4 What the Latent Prefix Carries . . . . . 25

D.5 Diverse-Domain Generalization . . . . . 26

**E Datasets and Evaluation Splits** **27**

E.1 Training Corpora . . . . . 27

E.2 Evaluation Benchmarks and Splits . . . . . 28

# A Extended Related Work

Section 2 introduces the work most directly connected to INTACT. Here we organize the broader literature by what happens to the written reasoning chain: it may remain in language, be removed, or be represented by continuous state. This distinction matters for INTACT, which replaces only an opening segment and leaves a frozen language model to produce the continuation.

## A.1 From Written to Latent Reasoning

Scratchpads and chain-of-thought make intermediate computation explicit (Nye et al., 2021; Wei et al., 2022; Kojima et al., 2022). More language-space computation can improve accuracy through sampled chains (Wang et al., 2023), test-time compute allocation (Snell et al., 2024; Muennighoff et al., 2025), process supervision (Lightman et al., 2024), or reinforcement learning on verifiable answers (DeepSeek-AI et al., 2025). Its cost is autoregressive decoding, and long-reasoning models can spend unnecessary computation on easy problems (Chen et al., 2024). Prompt compression reduces this cost while retaining a discrete, readable input (Jiang et al., 2023); after its latent prefix, INTACT likewise returns to language-space generation.

Other methods remove or replace the written chain. Implicit chain-of-thought distills a reasoning teacher into a student that answers directly (Deng et al., 2023), while stepwise internalization progressively deletes reasoning steps from the training target (Deng et al., 2024). Pause and filler tokens instead reserve computation without assigning those positions linguistic content (Goyal et al., 2024; Pfau et al., 2024). Compressed chain-of-thought predicts dense contemplation tokens as a surrogate for a trace (Cheng & Van Durme, 2024), and Token Assorted combines discrete codes with text (Su et al., 2025). These approaches reduce or eliminate the written intermediate sequence; INTACT is trained to re-encode its first $K$ steps and preserves a written continuation.

Continuous reasoning admits several constructions. Coconut feeds the model's last hidden state back as its next input embedding (Hao et al., 2024); CODI adds self-distillation between explicit and latent reasoning modes (Shen et al., 2025); and SIM-CoT supervises individual latent steps with an auxiliary decoder (Wei et al., 2026). Soft Thinking is training-free and replaces a sampled token with a probability-weighted mixture of embeddings (Zhang et al., 2025), although subsequent work questions whether this mixture provides genuinely parallel computation (Wu et al., 2025). Related architectures spend depth rather than generated tokens (Saunshi et al., 2025; Geiping et al., 2025). LOTUS jointly refines padded latent blocks through a looped backbone and supervises their token-level readout (Fan et al., 2026); MUX generates latent tokens sequentially but gives each a locally decodable target that multiplexes a span of the reference trace (Suleymanzade et al., 2026); and KaVa uses a compressed teacher KV-cache as its distillation target (Kuzina et al., 2026). LaDiR instead denoises blocks in a variational latent space whose frozen decoder reconstructs readable reasoning steps (Kang et al., 2026). Surveys by Zhu et al. (2025b) and Chen et al. (2025) provide broader taxonomies.

## A.2 Positioning of INTACT

INTACT is defined by two choices. The language model remains frozen, and the latents replace written steps rather than merely adding a task-specific prompt. Prompt tuning also keeps the model frozen, but learns one constant prefix per task (Lester et al., 2021). SoftCoT is the closest input-dependent alternative: a small frozen assistant reads the question and a learned projection maps its hidden states into soft tokens for a frozen backbone (Xu et al., 2025). It emits those tokens in one pass and uses them as additional context. INTACT instead derives each block from the backbone's own states, conditions it on the blocks already installed, and trains it to replace the effect of written steps.

Table 3 sets out the design choices that separate these methods. *Frozen* means the language model's parameters never change; *reads $M$'s states*, that the latents are computed from its hidden states rather than from its output distribution or from the question alone; *recurrent substitution*, that they are produced in sequence, each conditioned on those already placed, and stand in for written steps rather than supplementing them; *interpretable*, that the paper recovers token-level content from intermediate latents; and *multi-domain artifact*, that one trained artifact is evaluated across reasoning

domains without per-task adaptation. The last two columns record reported evidence, not an intrinsic limitation of methods without a checkmark.

Table 3: Design properties of representative latent-reasoning methods. Checkmarks record reported design choices or evidence, not a ranking of the methods.

| Method | Frozen | Reads $M$'s states | Recurrent subst. | Interpretable | Multi-domain artifact |
|---|---|---|---|---|---|
| Coconut (Hao et al., 2024) | × | ✓ | ✓ | × | × |
| CODI (Shen et al., 2025) | × | ✓ | ✓ | ✓ | × |
| SIM-CoT (Wei et al., 2026) | × | ✓ | ✓ | ✓ | × |
| LOTUS (Fan et al., 2026) | × | ✓ | × | ✓ | × |
| MUX (Suleymanzade et al., 2026) | × | ✓ | ✓ | ✓ | × |
| Soft Thinking (Zhang et al., 2025) | ✓ | × | ✓ | ✓ | × |
| SoftCoT (Xu et al., 2025) | ✓ | × | × | × | × |
| **INTACT** | ✓ | ✓ | ✓ | × | ✓ |

The methods that update the backbone answer a different experimental question. Coconut, CODI, SIM-CoT, LOTUS, and MUX jointly learn a latent mechanism and alter the language model, whereas INTACT isolates the interface by propagating gradients through fixed model weights. LOTUS also differs structurally: its blocks are refined jointly rather than emitted as recurrent substitutes. CODI, SIM-CoT, LOTUS, and MUX report token-level probes or readouts of their latents; INTACT does not make an interpretability claim.

Among the methods in Table 3, INTACT alone combines a frozen backbone, backbone-state feedback, and recurrent replacement of written steps. Soft Thinking is frozen and sequential, but constructs each soft token from an output distribution over the vocabulary rather than from hidden states. INTACT is also the only method in the table evaluated as one trained artifact across reasoning domains. These distinctions describe the intervention being evaluated; they do not imply that updating the backbone or making latents directly readable is undesirable.

**Against SoftCoT.** SoftCoT is the closest prior learned latent interface that also keeps the language model frozen. We use the authors' training script to train SoftCoT on LLAMA-3.1-8B-INSTRUCT. For a like-for-like comparison, both methods are evaluated under the SoftCoT protocol. The INTACT row is therefore a separate single-checkpoint evaluation and is not directly comparable to the three-seed results in Tables 1 and 2. Table 4 reports both accuracy and token use.

Table 4: Comparison with SoftCoT on LLAMA-3.1-8B-INSTRUCT under the SoftCoT evaluation protocol. Each entry reports accuracy (%)/mean total tokens, including latent and generated tokens. OOD is the macro-average over SVAMP, GSM-Hard, and MultiArith. Every entry is measured here: we train SoftCoT with the authors' script and evaluate both methods under their protocol. INTACT uses mathematics training (GSM8K-Aug).

| Method | GSM8K | SVAMP | GSM-Hard | MultiArith | OOD Avg. |
|---|---|---|---|---|---|
| SoftCoT | 81.0/250 | 81.0/187 | **30.6**/327 | 95.5/145 | 69.0/220 |
| INTACT | **82.3/192** | **83.5/143** | 30.2/**231** | **96.3/128** | **70.0/167** |

INTACT uses fewer total tokens on every benchmark and achieves a higher OOD average, although SoftCoT remains slightly more accurate on GSM-Hard. The separation is wider in cost than in accuracy: one point of out-of-domain accuracy against a 24% reduction in tokens, 167 on average rather than 220. That is the asymmetry the two designs predict, since SoftCoT's assistant reads the question alone and emits its tokens in a single pass, whereas each INTACT block is computed from the frozen model's state after the blocks already placed, and is trained to reproduce the effect of the written step it replaces rather than to supply additional context.

# B NOTATION

Table 5 collects the symbols used in the paper.

Table 5: Notation used in this paper.

| Symbol | Meaning |
|---|---|
| *Model and data* | |
| $M$, $p_M$ | frozen backbone and its next-token distribution |
| $\mathcal{V}$, $d_M$ | vocabulary and input-embedding dimension of $M$ |
| $\mathbf{x}$, $\mathbf{y}$ | token sequences representing the question and reference answer |
| $\mathbf{z}_{1:S}$ | reference reasoning trace of $S$ steps, $\mathbf{z}_i \in \mathcal{V}^{n_i}$ |
| $E(\mathbf{x})$ | token embeddings of the token string $\mathbf{x}$ |
| $\hat{\mathbf{y}}$, $\mathrm{Ans}(\cdot)$ | generated reasoning-and-answer continuation and answer extraction |
| *Latent interface* | |
| $K$, $c$ | number of latent blocks and vectors per block |
| $\mathbf{v}_{1:K}$ | latent prefix, block $\mathbf{v}_i \in \mathbb{R}^{c \times d_M}$ |
| $f_\theta$, $\theta$ | latent encoder and its parameters, the only trained weights |
| $H(\cdot)$ | final-layer states of $M$ on an embedding-space input $[E(\mathbf{x}), \mathbf{v}]$ |
| $q_j$ | learned queries specific to block $j$ |
| *Stage 1, progressive substitution* | |
| $i$ | substitution depth, running to $\min(K, S-1)$ |
| $C_i^{\mathrm{T}}$, $C_i^{\mathrm{S}}$ | teacher $[\mathbf{x}, \mathbf{z}_{1:i}]$ and student $[\mathbf{x}, \mathbf{v}_{1:i}]$ contexts |
| $\mathbf{u}_i$ | continuation $[\mathbf{z}_{i+1:S}, \mathbf{y}]$ shared by both contexts |
| $D(A\|B; \mathbf{u})$ | mean per-token KL between contexts $A$ and $B$ over $\mathbf{u}$ |
| $\mathcal{L}_{\mathrm{sub}}^{(i)}$ | substitution loss at depth $i$ |
| $\mathcal{L}_{\mathrm{pre}}$, $\beta$ | preservation loss on $\hat{\mathbf{y}}$ and its weight |
| *Stage 2, latent-policy refinement* | |
| $\pi_\theta$ | policy $p_M(\hat{\mathbf{y}} \mid \mathbf{x}, \mathbf{v}_{1:K})$, optimized through $\theta$ only |
| $G$, $\hat{\mathbf{y}}^g$ | group size and the $g$-th sampled continuation |
| $r^g$, $A^g$ | binary correctness reward and its group-standardized advantage |

# C EXPERIMENTAL DETAILS

## C.1 INTACT TRAINING

**Data and depth sampling.** For comparison against the latent-reasoning methods of Table 2, we train on the corpus they use, GSM8K-Aug, described in Appendix E. We parse its calculator annotations as ordered reasoning steps and retain the 322,712 examples containing at least two steps. For a trace of $S$ steps, Stage 1 supervises depths $i \leq \min(K, S-1)$, leaving at least one written step in the shared continuation. Only $55.5\%$, $23.2\%$, and $8.3\%$ of the retained examples support depths two, three, and four, respectively. We therefore divide the corpus into the four length buckets $S = 2$, $S = 3$, $S = 4$, and $S \geq 5$ and sample them uniformly. Figure 6 illustrates a complete four-block substitution.

**INTACT Stage 1.** All experiments use $K = 4$ blocks of $c = 4$ vectors. Each minibatch produces one encoder update at every depth supported by its examples. Preservation rollouts are sampled from the latent-conditioned model with temperature $0.7$ and top-$p = 0.95$, then held fixed while $\mathcal{L}_{\mathrm{pre}}$ is evaluated. The preservation weights are $\beta = 0.28$, $0.30$, $0.34$, and $0.28$ for Llama-1B, Llama-3B, Llama-8B, and Qwen-3B, respectively.

Teacher and student prefixes can occupy different numbers of positions. Let $\ell_z = |\mathbf{z}_{1:i}|$ and $\ell_v = ic$. We assign the latent span position IDs beginning at

$$\max(|\mathbf{x}|,\ |\mathbf{x}| + \ell_z - \ell_v)\,,$$

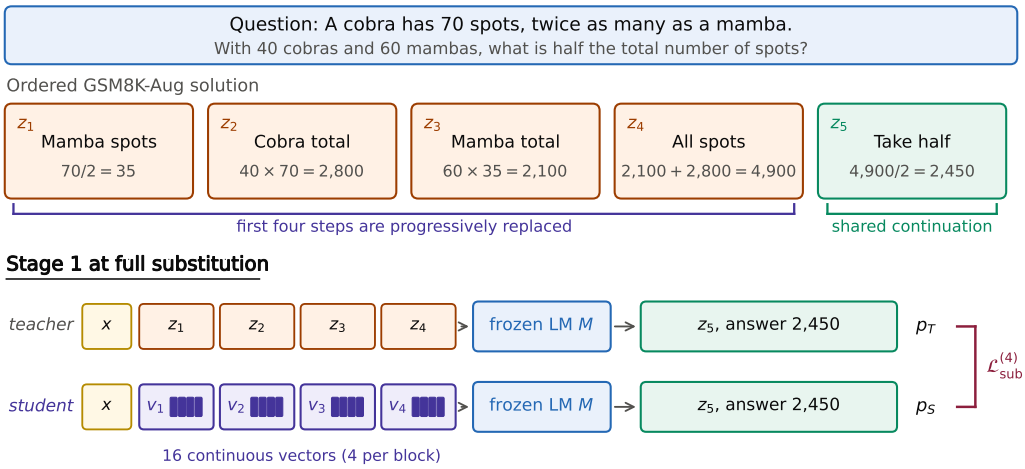


Figure 6: **Progressive substitution on a mathematics example.** GSM8K-Aug provides five ordered calculations; the question is shortened here for space. At full substitution, the student replaces the first four written steps $z_{1:4}$ with latent blocks $v_{1:4}$, while the final calculation $z_5$ and answer remain the shared continuation. The frozen language model evaluates both prefixes, and $\mathcal{L}_{\text{sub}}$ at depth $i = 4$ matches the resulting continuation distributions. Vertical alignment denotes which written step each latent block replaces, not a claim that the latent vectors decode to that step.

which aligns the shared continuation whenever $\ell_z \geq \ell_v$ and otherwise prevents the latent span from overlapping the question. This shift applies only to Stage 1 substitution, where teacher and student are evaluated on the same continuation. Preservation, Stage 2, and inference generate without a reference trace, so the question and latent prefix use contiguous position IDs.

On GSM8K-Aug the shift is systematic: the shortest step is 5 tokens against $c = 4$ latent slots, so $\ell_z > \ell_v$ at every depth and the clamp never applies. The displacement grows from 3 positions at $i$=1 to 14 at $i$=4, roughly a quarter of the prefix, so Stage 1 trains on a layout that inference does not reproduce.

Llama encoders are trained for 1,500 minibatch iterations, with checkpoints saved every 100 iterations. Qwen encoders are trained for 3,000 iterations and saved every 200. We select checkpoints by generation accuracy on 300 held-out GSM8K-Aug questions that do not overlap the GSM8K test set.

**INTACT Stage 2.** Each Stage 2 run begins from its selected Stage 1 checkpoint. We first evaluate four completions for each of the first 3,000 eligible training questions and retain only questions for which the completions include both a correct and an incorrect answer. This produces between 1,473 and 1,906 questions across the Stage 1 seed lineages. During training, every visit draws eight fresh completions; questions whose rewards all agree contribute no update.

An optimizer step contains 12 questions and eight completions per question, sampled at temperature 1.0. We optimize the per-sequence-normalized objective for 800 steps using AdamW with learning rate $10^{-5}$, zero weight decay, and gradient clipping at 1.0. The reported checkpoint is selected by generation accuracy on the held-out GSM8K-Aug split.

The frozen language model is kept in bfloat16 and the encoder in float32. Gradients pass through the language model, but only the encoder is updated. Each run uses one NVIDIA H100 NVL. Table 6 summarizes the optimization settings. Reported uncertainty is the sample standard deviation over the corresponding three encoder seed lineages.

Table 6: Hyperparameters for both stages. The optimiser is AdamW throughout.

| | LLAMA-3.2 1B-Instruct | LLAMA-3.2 3B-Instruct | LLAMA-3.1 8B-Instruct | QWEN2.5 3B (base) |
|---|---|---|---|---|
| **Stage 1: progressive substitution** | | | | |
| latent blocks $K$ | 4 | 4 | 4 | 4 |
| vectors per block $c$ | 4 | 4 | 4 | 4 |
| learning rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ |
| weight decay | 0.01 | 0.01 | 0.01 | 0.01 |
| minibatch size | 8 | 8 | 8 | 8 |
| gradient clipping | 1.0 | 1.0 | 1.0 | 1.0 |
| preservation weight $\beta$ | 0.28 | 0.30 | 0.34 | 0.28 |
| **Stage 2: latent-policy refinement** | | | | |
| learning rate | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ |
| weight decay | 0 | 0 | 0 | 0 |
| gradient clipping | 1.0 | 1.0 | 1.0 | 1.0 |
| rollouts per question | 8 | 8 | 8 | 8 |
| sampling temperature | 1.0 | 1.0 | 1.0 | 1.0 |

### C.2 DIVERSE-DOMAIN TRAINING AND EVALUATION

**Training corpus.** We construct a 12,000-example training stream from Llama-Nemotron. Checkpoints are selected on 200 held-out GSM8K-Aug validation questions that are disjoint from the GSM8K test set. The resulting mixture is 64.3% mathematics, 21.4% science, and 14.3% code. For mathematics, we use the reasoning-off collection of direct solutions from LLAMA-3.3-70B-INSTRUCT; the reasoning-on collection instead places the usable solution after a substantially longer exploratory trace. The science and code examples are shared by both collections. We convert responses to ordered steps using equation-bearing spans for mathematics, sentences for science, and lines of fenced programs for code.

**Training.** We train the same 107M encoder on LLAMA-3.2-3B-INSTRUCT, with $K = 4$ blocks of $c = 4$ vectors. Stage 1 runs for 1,500 steps with learning rate $10^{-4}$, batch size 8, a 200-step warmup, and preservation weight $\beta = 0.30$, using all three domains. Stage 2 starts from that checkpoint and runs for 800 group-relative policy-gradient steps with learning rate $10^{-5}$, batch size 12, and groups of eight rollouts sampled at temperature 1.0. Its rollout pool is 75% Nemotron mathematics and 25% Nemotron science as code is not used in Stage 2. Mathematics receives a numeric-correctness reward, and science receives an option-letter correctness reward. Rollouts are capped at 256 tokens, and no KL penalty to a reference policy is used. All other settings follow Appendix C.1 and the language model remains frozen throughout.

**Evaluation.** We evaluate under the sampling protocol of Appendix C.4 with a 2,048-token budget. The loose budget limits the censoring caused by unfinished frozen-model generations, which at this budget still run to the cap on 4.2–10.5% of the mathematics examples against 0.3–4.4% for Stage 2. Reported INTACT costs include both the generated tokens and the 16 latent positions.

### C.3 BASELINES

**Result provenance.** At LLAMA-3.2-1B and LLAMA-3.2-3B, we quote No-CoT, Aug-SFT, MUX, and CODI from Suleymanzade et al. (2026), and LOTUS from Fan et al. (2026). Measured entries at LLAMA-3.1-8B and QWEN2.5-3B are ours. No directly comparable latent-reasoning checkpoint has been released for these backbones. Published 8B MUX and CODI results use base models, whereas our 8B backbone is instruction-tuned.

**No-CoT and Aug-SFT.** Our reproductions use the same script and differ only in the supervision target. We apply rank-128 LoRA to the `q,k,v,o,gate,up,down` projections of every layer, with $\alpha = 2r$, dropout 0.05, and no bias. Training uses AdamW with learning rate $10^{-4}$, zero

weight decay, 100 warmup steps followed by cosine decay, gradient clipping at 1.0, sequence length 512, and effective batch size 16 in bfloat16. Both train on all 385,530 GSM8K-Aug examples, where INTACT uses the 322,712 of them that carry at least two steps (Appendix C.1). The adapters contain 194.5M trainable parameters at 3B and 239.5M on Qwen.

**Nemo-SFT.** We train this baseline on all four backbones using the same script and LoRA configuration as Aug-SFT, but replace GSM8K-Aug with the Llama-Nemotron mixture from Appendix C.2. Thus the two baselines differ only in their supervision: terse calculator expressions averaging roughly 31 tokens for Aug-SFT, and natural-language solutions averaging roughly 750 tokens for Nemo-SFT. We select the step-200 checkpoint on every backbone.

**Latent-reasoning baselines.** For LOTUS on QWEN2.5-3B, we run the authors' code with $c_{\text{thought}} = 25$, six latent loops, constant AdamW learning rate $5 \times 10^{-5}$, effective batch size 128, and seven epochs. This covers the curriculum through `max_latent_stage=6`; we report the final epoch using the authors' evaluation script. The quoted MUX results use the shared LoRA configuration reported in its Table 8: rank 128, $\alpha = 32$, and dropout 0.1. The quoted CODI results come from the same codebase and configuration, with CODI's additional hidden-size projector included in its trainable-parameter count. Figure 14 additionally includes SIM-CoT at 3B. We take its GSM8K and out-of-domain accuracies from Table 2 of Suleymanzade et al. (2026) and estimate its trainable parameters as the shared rank-128 LoRA adapter plus a full auxiliary decoder with $P_{\text{dec}} = P$, following their Appendix B.4.

### C.4 Evaluation Protocol

**Prompting and decoding.** For numerical benchmarks, the model receives the bare question followed by a newline. Multiple-choice prompts instead use `Question:` followed by the question and lettered options. Neither format includes few-shot demonstrations or a model-specific chat template. Following the CODI/MUX protocol, we sample with temperature 0.1, top-$k$ 40, and top-$p$ 0.95. Quoted results retain the protocol of their source.

**Generation budget.** Results trained on GSM8K-Aug use a 768-token cap, and results trained on the Llama-Nemotron mixture use 2,048. The larger budget reflects that mixture's longer targets: its steps are complete natural-language solutions rather than compact arithmetic, and at a 512-token cap the frozen backbone is truncated on up to 13.9% of mathematics problems, suppressing the reference point the gains are measured against. Appendix D.1 sweeps the cap from 128 to 768 tokens; over that last interval most systems move by tenths of a point, QWEN2.5-3B still gains 1.3, and the frozen LLAMA-3.1-8B loses accuracy because it rarely terminates (Appendix D.3). Table 4 instead adopts the 1,024-token cap of the SoftCoT protocol for both of its rows. Quoted results retain the budget of their source.

**Answer extraction.** We discard text following a newly generated `Question:`, which base models sometimes emit after completing a solution. For numerical benchmarks, we extract, in order, a value following `####` or `###`, an explicit "answer is" phrase, a boxed value, or the final number in the remaining completion. A prediction is correct when the extracted value matches the gold answer within $10^{-4}$. For multiple-choice benchmarks, we extract an explicit answer letter first, then the last matching option text, and finally the last standalone letter. Correctness is exact match with the gold option letter.

## D Extended Results

This appendix first expands the GSM8K-Aug results across generation budgets, then examines output length and computational cost. It closes with the per-benchmark results for the encoder trained on the Llama-Nemotron mixture.

### D.1 INTACT and the Frozen Backbone Across Budgets

Table 7 compares the frozen backbone with both INTACT stages at every backbone and every generation budget, of which Table 1 shows one cap. All budget values are scored from prefixes of the

Table 7: Accuracy (%) as a function of the generation budget, with the encoder trained on GSM8K-Aug. *GSM8K* is the matching test set and *OOD* averages SVAMP, GSM-Hard and MultiArith. Stage 1 is chain-to-latent distillation alone and Stage 2 adds the latent-policy stage. Columns with ± show mean ± sample standard deviation over three encoder seeds.

| | | **Frozen LM** | | INTACT **Stage 1** | | INTACT **Stage 2** | |
|---|---|---|---|---|---|---|---|
| **Backbone** | **Budget** | **GSM8K ↑** | **OOD ↑** | **GSM8K ↑** | **OOD ↑** | **GSM8K ↑** | **OOD ↑** |
| LLAMA 3.2-1B | 128 | 13.2 | 33.4 | 14.0 ± 0.6 | 35.8 ± 0.2 | 17.3 ± 2.6 | 42.1 ± 3.0 |
| | 256 | 37.8 | 47.3 | 40.5 ± 0.9 | 50.8 ± 0.9 | 45.5 ± 1.2 | 54.1 ± 0.4 |
| | 384 | 43.7 | 47.8 | 43.9 ± 1.0 | 51.3 ± 1.0 | 48.5 ± 1.4 | 54.4 ± 0.5 |
| | 512 | 44.0 | 47.8 | 44.2 ± 1.0 | 51.3 ± 0.9 | 48.8 ± 1.3 | 54.5 ± 0.4 |
| | 768 | 44.0 | 47.8 | 44.2 ± 1.0 | 51.3 ± 1.0 | 48.7 ± 1.3 | 54.5 ± 0.4 |
| LLAMA 3.2-3B | 128 | 15.3 | 40.0 | 15.2 ± 0.7 | 42.6 ± 1.0 | 24.7 ± 1.7 | 52.6 ± 0.9 |
| | 256 | 61.6 | 64.1 | 65.0 ± 0.7 | 66.9 ± 0.4 | 72.2 ± 0.9 | 67.5 ± 0.1 |
| | 384 | 74.1 | 66.2 | 76.4 ± 0.5 | 68.8 ± 0.3 | 78.4 ± 0.3 | 68.7 ± 0.3 |
| | 512 | 75.8 | 66.5 | 77.6 ± 0.3 | 69.0 ± 0.3 | 79.2 ± 0.3 | 68.8 ± 0.2 |
| | 768 | 75.8 | 66.5 | 77.9 ± 0.3 | 69.1 ± 0.3 | 79.3 ± 0.4 | 68.8 ± 0.2 |
| LLAMA 3.1-8B | 128 | 14.9 | 38.9 | 15.1 ± 0.7 | 41.0 ± 1.2 | 30.7 ± 2.8 | 55.8 ± 1.6 |
| | 256 | 62.8 | 63.7 | 66.6 ± 0.7 | 68.2 ± 0.7 | 78.9 ± 2.1 | 71.3 ± 0.4 |
| | 384 | 76.3 | 62.1 | 81.8 ± 0.8 | 70.9 ± 0.9 | 83.3 ± 0.6 | 72.1 ± 0.7 |
| | 512 | 75.7 | 57.0 | 83.4 ± 0.6 | 71.1 ± 1.0 | 83.6 ± 0.6 | 72.1 ± 0.7 |
| | 768 | 72.3 | 53.9 | 83.7 ± 0.6 | 71.1 ± 1.1 | 83.6 ± 0.7 | 72.1 ± 0.7 |
| QWEN 2.5-3B | 128 | 18.5 | 41.9 | 28.2 ± 3.1 | 47.3 ± 2.5 | 52.3 ± 0.8 | 68.4 ± 0.3 |
| | 256 | 51.6 | 68.1 | 57.3 ± 0.8 | 69.3 ± 0.2 | 78.7 ± 0.5 | 74.8 ± 0.5 |
| | 384 | 70.3 | 73.1 | 72.9 ± 2.0 | 73.6 ± 1.0 | 79.7 ± 0.4 | 75.1 ± 0.5 |
| | 512 | 76.3 | 74.3 | 77.2 ± 2.1 | 74.7 ± 1.0 | 79.7 ± 0.5 | 75.1 ± 0.5 |
| | 768 | 77.6 | 74.6 | 78.0 ± 2.0 | 75.0 ± 1.0 | 79.7 ± 0.5 | 75.1 ± 0.5 |

same generation pass. The frozen LLAMA-3.1-8B row falls as the budget grows because that model rarely terminates (Appendix D.3): a larger cap lets more of its generations run past the answer rather than finishing.

**Prompt format.** That shortfall is a property of the prompt as much as of the model. Our protocol supplies the bare question, with no chat template or answer-format instruction (Appendix C.4). Re-evaluating the three instruction-tuned Llama backbones with their own chat templates, changing nothing else (Figure 7), removes the termination failure at 8B: capped GSM8K generations fall from 97.1% to 0.9% and accuracy rises from 72.3 to 78.4. The template is not uniformly better, as GSM8K accuracy rises at 1B and falls at 3B, and INTACT Stage 2, still scored under the bare prompt, remains more accurate than the chat-prompted frozen model on GSM8K and out of domain at all three scales (Table 8). Its token cost is closer: INTACT generates fewer tokens at 8B (182 against 186) but more at 1B and 3B.

**Bare question**

```
John runs 60 miles in 3 days.
How many miles per day?\n
```

**Llama-3.1-8B chat template**

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>
Cutting Knowledge Date:  December 2023
Today Date:  26 Jul 2024
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
John runs 60 miles in 3 days.
How many miles per day?<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Figure 7: Inputs compared for the frozen LLAMA-3.1-8B-INSTRUCT. The question is illustrative; the chat template adds its default system metadata and turn markers.

Table 8: Prompt sensitivity of the frozen Llama backbones at a 768-token cap. The two frozen rows differ only by prompt format. The INTACT rows are Stage 2, evaluated with the bare prompt for reference. OOD is the out-of-domain macro-average, and *Tok.* is mean generated tokens on GSM8K.

| Backbone | Condition | GSM8K↑ | OOD↑ | Tok.↓ |
|---|---|---|---|---|
| LLAMA-3.2-1B | Frozen (bare) | 44.0 | 47.8 | 252 |
| | Frozen (chat) | 47.3 | 52.2 | **165** |
| | INTACT (bare) | **48.7** | **54.5** | 215 |
| LLAMA-3.2-3B | Frozen (bare) | 75.8 | 66.5 | 261 |
| | Frozen (chat) | 71.6 | 66.1 | **172** |
| | INTACT (bare) | **79.3** | **68.8** | 195 |
| LLAMA-3.1-8B | Frozen (bare) | 72.3 | 53.9 | 754 |
| | Frozen (chat) | 78.4 | 69.6 | 186 |
| | INTACT (bare) | **83.6** | **72.1** | **182** |

### D.2 ACCURACY–EFFICIENCY OPERATING POINTS

Figures 8–10 compare accuracy with total positions; upper left is better, and INTACT counts include all 16 latent positions. Figures 11–13 instead sweep the generation budget, with dashed curves updating the language model and solid curves keeping it frozen. LLAMA-3.2-1B is omitted because its logs do not contain the finer budget grid these panels require. At 8B, no comparable CODI or LOTUS checkpoint is available and the MUX point is our reproduction (Appendix C.3).

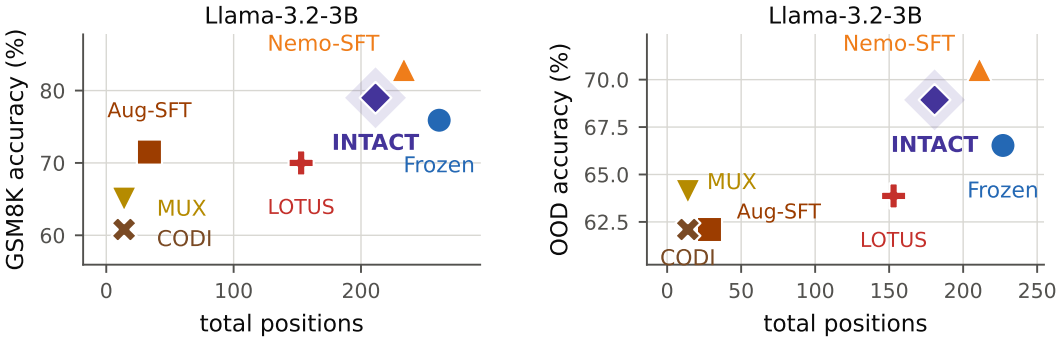


Figure 8: LLAMA-3.2-3B: accuracy against total positions on GSM8K (left) and the out-of-domain macro-average (right); upper left is better. Frozen and Nemo-SFT are evaluated at the 768-token cap, and every other system at the operating point it reports.

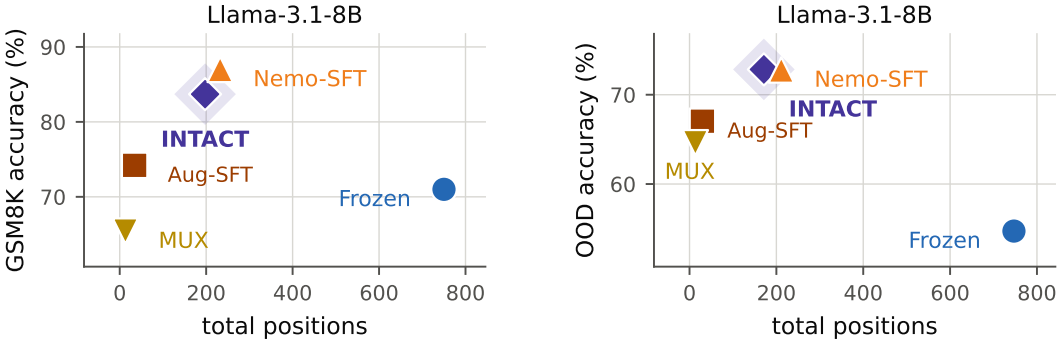


Figure 9: LLAMA-3.1-8B: accuracy against total positions on GSM8K (left) and the out-of-domain macro-average (right); upper left is better. Frozen and Nemo-SFT are evaluated at the 768-token cap, and every other system at the operating point it reports.

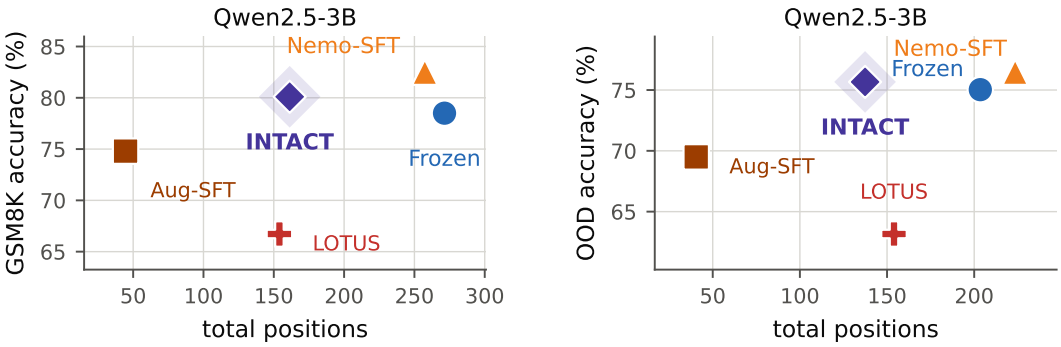


Figure 10: QWEN2.5-3B: accuracy against total positions on GSM8K (left) and the out-of-domain macro-average (right); upper left is better. Frozen and Nemo-SFT are evaluated at the 768-token cap, and every other system at the operating point it reports.

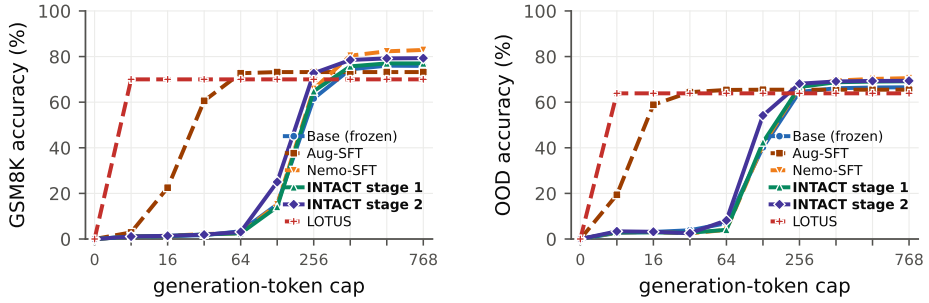


Figure 11: LLAMA-3.2-3B: accuracy against the generation budget on GSM8K (left) and the out-of-domain macro-average (right). Solid curves keep the language model frozen and dashed curves update it. Every budget is scored from prefixes of one generation pass, so a curve shows what that same run would have returned under a tighter cap.

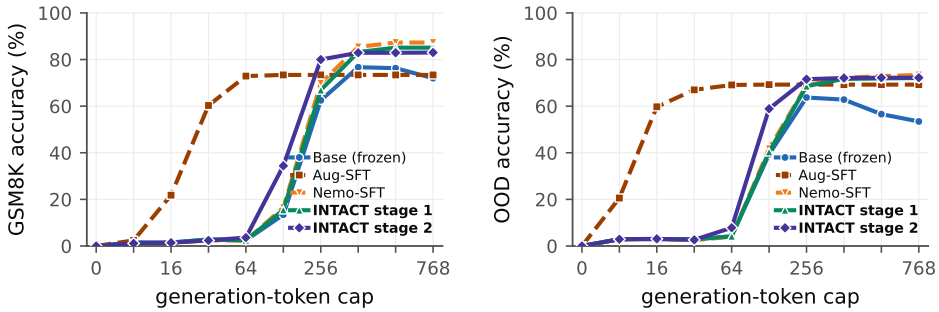


Figure 12: LLAMA-3.1-8B: accuracy against the generation budget on GSM8K (left) and the out-of-domain macro-average (right). Solid curves keep the language model frozen and dashed curves update it. Every budget is scored from prefixes of one generation pass, so a curve shows what that same run would have returned under a tighter cap.

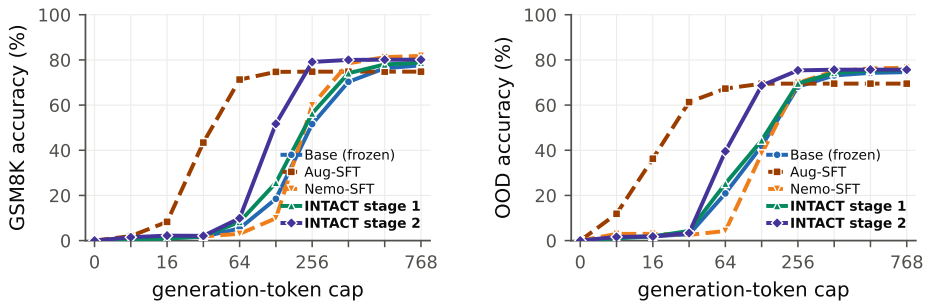


Figure 13: QWEN2.5-3B: accuracy against the generation budget on GSM8K (left) and the out-of-domain macro-average (right). Solid curves keep the language model frozen and dashed curves update it. Every budget is scored from prefixes of one generation pass, so a curve shows what that same run would have returned under a tighter cap.

### D.3 EFFICIENCY

Three quantities are reported, and they are not interchangeable.

- **Generated tokens.** The text the backbone emits, excluding the latent prefix. Table 9 and the ablations report this.
- **Total positions.** Generated tokens plus INTACT's $Kc = 16$ latent slots, excluding the question, so it is the cost of an INTACT answer against a baseline that has no prefix to pay for. The operating-point figures and Table 12 use this.
- **End-to-end latency.** Wall-clock milliseconds per problem, and the only quantity that charges for *building* the prefix rather than merely for its length.

The *generation budget* is not one of these: it is the cap supplied to the decoder (Appendix C.4), a setting rather than a measurement. A system that stops early is unaffected by it, whereas one that never stops reports the budget back as its cost.

Table 9: Mean generated tokens at a 768-token cap, for encoders trained on GSM8K-Aug. OOD is the macro-average over the three held-out datasets; lower is better.

| | **Frozen** | | **Stage 1** | | **Stage 2** | |
|---|---|---|---|---|---|---|
| | **GSM8K↓** | **OOD↓** | **GSM8K↓** | **OOD↓** | **GSM8K↓** | **OOD↓** |
| 1B | 252 | 210 | 241 | 202 | **215** | **171** |
| 3B | 261 | 224 | 225 | 190 | **195** | **165** |
| 8B | 754 | 745 | 269 | 207 | **182** | **155** |
| Qwen | 273 | 203 | 246 | 196 | **145** | **121** |

**Generation length and termination.** Table 9 reports mean generated tokens per problem for the frozen backbone and both INTACT stages, from the same 768-token pass as Table 7. Both stages shorten generation on every backbone, and Stage 2 is shorter than Stage 1 in every row. QWEN2.5-3B is the clearest example, answering in 145 tokens against the frozen backbone's 273 in domain and 121 against 203 out of domain.

The LLAMA-3.1-8B row differs because the frozen model rarely terminates: 97.1% of its GSM8K generations reach the cap, compared with 8.6% for Stage 1 and none for Stage 2. Its shorter learned-stage generations are therefore completed answers rather than earlier truncations.

**Trainable parameters and training cost.** Figure 14 places the LLAMA-3.2-3B systems of Table 2 against the number of parameters each trains, together with SIM-CoT. INTACT trains a 107.1M encoder and never updates the backbone, and is the most accurate of the methods that leave it frozen;

Nemo-SFT is more accurate still at 194.5M trainable parameters, which it uses to fine-tune the backbone. The LoRA baselines train 194.5M at $r$=128, which is Aug-SFT's and MUX's shared adapter configuration, and CODI adds a projector on top of that for 213.4M. LOTUS fine-tunes the backbone outright at 3.21B, and SIM-CoT trains a LoRA adapter plus a full auxiliary decoder with $P_{\text{dec}} = P$ parameters (MUX Appendix B.4), so at 3.40B it updates more than a full fine-tune does. No paper in this comparison reports its own trainable count, so each is derived from the method's description; Appendix C.3 gives the corresponding sources and parameter-count derivations. INTACT is the leftmost point in both panels and the only one that leaves the backbone frozen. Nemo-SFT places higher on both, and does so by updating 194.5M backbone parameters.

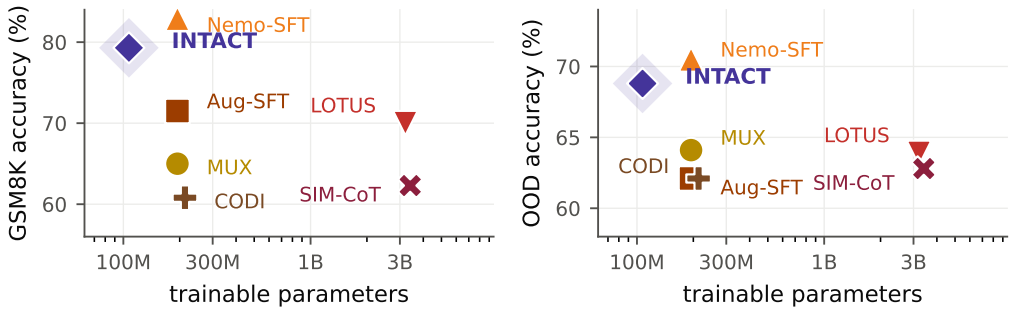


Figure 14: **Accuracy against trainable parameters on LLAMA-3.2-3B**, where up and to the left is better. *OOD* averages SVAMP, GSM-Hard and MultiArith. INTACT trains a 107M-parameter encoder and never updates the backbone, while the baselines train between 195M and 3.40B. No method here reports a trainable count, so each is derived (Appendix C.3). Mathematics training (GSM8K-Aug), except Nemo-SFT (Llama-Nemotron).

Freezing the backbone also removes its parameter gradients and optimizer state, although its activations remain necessary for gradients to reach the encoder. On one H100 NVL, Stage 1 takes 6.2 hours and Stage 2 a further 10.3 hours, placing the complete two-stage recipe below 17 GPU-hours.

**Inference latency.** The following breakdown reports average milliseconds per GSM8K example at batch 1 on one H100 NVL with the LLAMA-3.2-3B backbone, timed over the full GSM8K test split with the Stage 2 encoder. *Tokens* counts generated tokens only and reproduces the corresponding column of Table 9. *Latent* is the time spent constructing the latent prefix before generation, and *generation* is the subsequent autoregressive decoding time.

| System | Latent (ms) | Generation (ms) | Tokens | ms/token | Total (ms) |
|---|---|---|---|---|---|
| Frozen | 0.0 | 4,179.6 | 260.8 | 16.03 | 4,179.7 |
| INTACT | 127.5 | 3,258.1 | 195.5 | 16.67 | 3,385.6 |

INTACT finishes 794ms earlier, a 19% end-to-end reduction: generation is 921.5ms faster and constructing the latent prefix costs 127.5ms. The generation saving is the net of two opposing terms. Emitting 65.3 fewer tokens saves about 1,046ms at the frozen model's rate of 16.03ms per token, while the 16 additional positions raise INTACT's rate to 16.67ms per token and cost about 125ms across the 195.5 tokens it does emit. The saving therefore comes from earlier termination, and slower decoding reduces it by roughly an eighth rather than contributing to it.

### D.4 WHAT THE LATENT PREFIX CARRIES

Two controls ask whether the prefix is a question-specific, ordered object rather than a generic cue that merely shortens generation. Both are reported for Stage 1, where the prefix is trained to substitute for written steps; the Stage 2 objective rewards answers rather than step correspondence, so the same controls there test a different claim and are outside the scope of this section.

Table 10: Two controls on the latent prefix. **Top:** permuting a trained Stage 1 prefix at inference on LLAMA-3.2-3B (full splits, 768-token cap; Table 1's first Stage 1 seed, own sampling pass). *Within* permutes the four blocks of the same question; *across* replaces them with another question's. **Bottom:** removing the encoder's recurrence on QWEN2.5-3B (full splits, 768-token cap), so every block is computed from the question alone. *Comp.* is the fraction of generations that terminate.

| | GSM8K ↑ | OOD ↑ | Tok. ↓ | Comp. ↑ |
|---|---|---|---|---|
| *Prefix permutation, Stage 1 (*LLAMA-3.2-3B*)* | | | | |
| Trained prefix | **78.8** | **69.1** | **225** | **0.99** |
| Shuffled within question | 65.2 | 61.2 | 330 | 0.80 |
| Shuffled across questions | 69.8 | 62.0 | 304 | 0.85 |
| *Encoder recurrence (*QWEN2.5-3B*)* | | | | |
| Frozen backbone | 77.6 | 74.6 | 273 | — |
| Question-only blocks | 76.9 | **75.6** | 303 | — |
| Recurrent blocks | **78.8** | 75.5 | **246** | — |

**Order and identity.** Permuting a trained prefix at inference degrades it. Shuffling the four blocks *within* a question costs 13.6 points on GSM8K and 7.9 out of domain, and the paired comparison is decisive: 220 problems are solved only with the correct order against 40 only with the shuffled one (McNemar exact, $p < 10^{-30}$). Substituting another question's prefix costs 9.0 and 7.1 points (161 against 42, $p < 10^{-16}$). Every one of the eight paired comparisons across the four benchmarks favours the trained prefix at $p < 10^{-4}$. Both permutations also suppress termination, from 99% of GSM8K generations completing to 80% and 85%, while generation grows by 79–105 tokens. The prefix is therefore read as an ordered sequence, and its contents are not interchangeable across questions.

**Recurrence.** Computing every block from the question alone, rather than conditioning each block on the model's state after the preceding ones, removes the in-domain gain: the question-only encoder reaches 76.9 against the frozen backbone's 77.6, while the recurrent encoder reaches 78.8. It is also less efficient, generating 303 tokens against 246. Out-of-domain accuracy is unaffected (75.6 against 75.5), so recurrence buys in-domain accuracy and token efficiency rather than transfer. This arm is a single seed, and its checkpoint is selected independently of the default's.

### D.5 DIVERSE-DOMAIN GENERALIZATION

Table 11: Accuracy (%) of one LLAMA-3.2-3B encoder across twelve benchmarks at a 2,048-token budget, trained on the Llama-Nemotron mixture rather than GSM8K-Aug. Each system is a single checkpoint used in every domain, so a column is one encoder rather than a per-domain best. Δ is Stage 2 against the frozen backbone. Every split is scored in full. Bold is the best of the three trained systems.

| Domain | Benchmark | n | Frozen | Stage 1 | Stage 2 | Nemo-SFT | Δ |
|---|---|---|---|---|---|---|---|
| Science | OpenBookQA | 500 | 48.4 | 63.8 | 67.6 | **71.2** | +19.2 |
| | ARC-Challenge | 1172 | 50.1 | 65.4 | 72.6 | **73.0** | +22.5 |
| | ARC-Easy | 2376 | 67.5 | 80.8 | **86.2** | 85.5 | +18.7 |
| | SciQ | 1000 | 77.3 | 86.0 | **87.3** | 86.5 | +10.0 |
| | MMLU-science | 1397 | 43.8 | 49.3 | 54.3 | **56.8** | +10.5 |
| Commonsense | ECQA | 2162 | 52.6 | 58.5 | 64.2 | **67.0** | +11.6 |
| | SocialIQA | 1954 | 42.9 | 50.3 | **64.3** | 46.7 | +21.4 |
| | HellaSwag | 10042 | 32.4 | 39.0 | **56.2** | 52.5 | +23.8 |
| Mathematics | GSM8K | 1319 | 75.4 | 77.6 | 78.7 | **83.4** | +3.3 |
| | SVAMP | 1000 | 77.9 | 80.8 | 81.6 | **83.3** | +3.7 |
| | MultiArith | 600 | 95.5 | 97.0 | 97.5 | **98.7** | +2.0 |
| | GSM-Hard | 1319 | 26.2 | 26.2 | 26.6 | **28.9** | +0.4 |

**Accuracy by stage.** Table 11 gives the accuracies behind Figure 5. The two stages divide the work differently by domain. In science, Stage 1 contributes 15.4 of the final 19.2-point gain on OpenBookQA and 15.3 of 22.5 points on ARC-Challenge. Stage 2 has its largest effect in commonsense: it adds 14.0 points on SocialIQA beyond the 7.4-point Stage 1 gain. Its contribution on mathematics is smaller but consistently positive, ranging from 0.4 to 1.1 points. Across the three commonsense benchmarks, Stage 2 reaches 61.6% compared with 55.4% for Nemo-SFT.

Table 12: Mean total positions per problem for the encoder trained on the Llama-Nemotron mixture, at a 2,048-token budget. INTACT columns include the 16 latent prefix positions as well as the tokens generated, so a number here is comparable across rows and to Figure 1. Lower is better, and bold is the cheapest system on a row.

| Domain | Benchmark | n | Frozen | Stage 1 | Stage 2 | Nemo-SFT |
|---|---|---|---|---|---|---|
| Science | OpenBookQA | 500 | 277.0 | 244.0 | **80.0** | 147.0 |
| | ARC-Challenge | 1172 | 234.0 | 240.0 | **82.0** | 166.0 |
| | ARC-Easy | 2376 | 262.0 | 237.0 | **98.0** | 153.0 |
| | SciQ | 1000 | 338.0 | 191.0 | **123.0** | 145.0 |
| | MMLU-science | 1397 | 376.0 | 278.0 | **148.0** | 246.0 |
| | Mean | | 297.4 | 238.0 | **106.2** | 171.4 |
| Commonsense | ECQA | 2162 | 214.0 | 146.0 | **61.0** | 130.0 |
| | SocialIQA | 1954 | 177.0 | 256.0 | 143.0 | **141.0** |
| | HellaSwag | 10042 | 776.0 | 756.0 | **60.0** | 138.0 |
| | Mean | | 389.0 | 386.0 | **88.0** | 136.3 |
| Mathematics | GSM8K | 1319 | 358.0 | 259.3 | 254.8 | **241.9** |
| | SVAMP | 1000 | 289.2 | 201.7 | 197.8 | **197.4** |
| | MultiArith | 600 | 215.4 | 176.5 | 154.7 | **133.2** |
| | GSM-Hard | 1319 | 448.1 | 388.7 | **352.1** | 357.0 |
| | Mean | | 327.7 | 256.6 | 239.8 | **232.4** |

**Token cost and termination.** Table 12 gives what each system spends to reach the accuracies of Table 11. Stage 2 uses fewer total positions than the frozen backbone on all twelve benchmarks while improving accuracy on each. Stage 1 is cheaper on ten of twelve but is more expensive on ARC-Challenge and SocialIQA. The shortening is therefore not imposed mechanically by the latent prefix; it emerges most consistently after latent-policy refinement.

The 2,048-token budget reduces but does not eliminate truncation of frozen-model generations. Their reported means consequently remain more sensitive to the evaluation cap than those of INTACT and Nemo-SFT, which terminate more consistently.

# E Datasets and Evaluation Splits

## E.1 Training Corpora

The experiments use two training corpora, each for a separate encoder.

**GSM8K-Aug.** The main mathematics experiments of Section 4.1 use **GSM8K-Aug** (Deng et al., 2023), a 385K-example augmentation of GSM8K whose solutions consist of ordered, compact arithmetic steps. This structure directly supports progressive substitution: latent block $i$ replaces written step $i$ without additional segmentation.

**Llama-Nemotron.** The diverse-domain experiment of Section 4.4 uses **Llama-Nemotron post-training data** (Bercovich et al., 2025), which combines mathematics, science, and code under one collection process. We use its reasoning-off subset, whose responses are direct written solutions rather than search traces; Appendix C.2 describes the sampling and step extraction.

Table 13: Evaluation benchmarks and scored split sizes, grouped by domain. Every row uses the complete evaluation split after ECQA de-duplication. *Form* is either a numeric answer or an $m$-way letter choice.

| Domain | Benchmark | n | Form |
|---|---|---|---|
| Mathematics | GSM8K (Cobbe et al., 2021) | 1319 | number |
| | GSM-Hard (Gao et al., 2023) | 1319 | number |
| | SVAMP (Patel et al., 2021) | 1000 | number |
| | MultiArith (Roy & Roth, 2015) | 600 | number |
| Science | OpenBookQA (Mihaylov et al., 2018) | 500 | 4-way |
| | ARC-Easy (Clark et al., 2018) | 2376 | 4-way |
| | ARC-Challenge (Clark et al., 2018) | 1172 | 4-way |
| | SciQ (Welbl et al., 2017) | 1000 | 4-way |
| | MMLU-science (Hendrycks et al., 2021a) | 1397 | 4-way |
| Commonsense | ECQA (Aggarwal et al., 2021) | 2162 | 5-way |
| | SocialIQA (Sap et al., 2019) | 1954 | 3-way |
| | HellaSwag (Zellers et al., 2019) | 10,042 | 4-way |

### E.2 Evaluation Benchmarks and Splits

**Benchmark coverage.** Table 13 lists every benchmark scored in the paper. Each benchmark uses its complete evaluation split, subject only to the ECQA de-duplication described below. GSM8K is the matching test set for GSM8K-Aug. For the diverse-domain experiment, we check for overlap after case-folding the question text and removing punctuation and multiple-choice options. We find no exact question-text match between the examples used in either training stage and any of the twelve evaluation splits. No evaluation question is used for checkpoint selection. Selection uses the disjoint 200-question GSM8K-Aug validation split described in Appendix C.2. Multiple-choice benchmarks share the prompt format specified in Appendix C.4.

**Dataset-specific conventions.** *MMLU-science* is not a released split. It aggregates the eight MMLU subjects covered by Llama-Nemotron's science data: high-school and college physics, chemistry, and biology, together with conceptual physics and astronomy. ECQA contains 32 test questions that also occur in its training split, inherited from CommonsenseQA; we remove these questions before scoring, leaving 2,162 examples.