SOFT TEACHER FORCING

Anonymous authors
Paper under double-blind review

ABSTRACT

Soft tokens have so far been used to replace discrete chain-of-thought in latent reasoning methods. These approaches typically require multi-stage training and, in some cases, auxiliary modules, yet they still underperform supervised fine-tuning (SFT) on ⟨input, chain-of-thought, output⟩ triples. In this paper, we ask whether soft tokens can be used to improve SFT itself, instead of as an avenue for latent reasoning. We introduce Soft Teacher Forcing (Soft-TF), a simple training algorithm that combines teacher-forced embeddings with soft tokens through random convex combinations. The soft tokens are computed in a single parallel forward pass over the ground-truth sequence. Soft-TF adds no parameters, no external model and no curriculum, and leaves inference unchanged. Experiments show that Soft-TF improves over SFT on GSM8K, MATH, MathQA and AQuA, by +2.1 to +4.1 points on GSM8K across seven Gemma, Llama and Mistral models.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities on complex reasoning tasks through Chain-of-Thought (CoT) (Wei et al., 2022; Wang et al., 2023), in which the model autoregressively generates intermediate tokens that explicitly trace the steps toward a final answer. Building on this, supervised fine-tuning (SFT) on ⟨*input*, *CoT*, *output*⟩ triples has become a standard approach for training LLMs for reasoning (Mukherjee et al., 2023; Yue et al., 2024; Chung et al., 2024; Shao et al., 2024), before further Reinforcement Learning (RL) (Lightman et al., 2024; Liu et al., 2024; Team et al., 2025b). SFT uses teacher forcing (TF) (Jordan, 1986; Williams & Zipser, 1989), which conditions each prediction on the preceding ground-truth tokens rather than on the model's own outputs. This allows training to be fully parallelized and aligns naturally with the next-token prediction objective that underlies modern LLMs.

A separate line of work has explored soft tokens, which are continuous inputs such as convex combinations of vocabulary embeddings or last-layer hidden states. The motivation is that a continuous space may be more expressive than a discrete one (Hao et al., 2025; Zhang et al., 2025), and soft tokens have primarily been used to replace discrete CoT (Deng et al., 2024; Hao et al., 2025; Shen et al., 2025; Xu et al., 2025). However, these latent reasoning methods still underperform the SFT models they use as initialization on real-world mathematical reasoning benchmarks (Cobbe et al., 2021). Being recursive by nature, they deviate from TF and are also computationally expensive: generating n latent tokens requires $n + 1$ sequential forward passes, and backpropagating through them requires storing every intermediate activation in memory.

While these difficulties are often interpreted as evidence that soft tokens are not yet ready to replace discrete CoT, we view them as an opportunity to ask a different question: rather than attempting to *replace* discrete reasoning, can soft tokens be used to *improve* standard SFT? We answer this positively by introducing **Soft Teacher Forcing (Soft-TF)**, an algorithm that trains on random convex combinations of teacher-forced (*i.e.*, ground-truth) embeddings and soft tokens. Given a training sequence, a single forward pass with causal masking over the ground truth yields the model's own next-token distribution at every position. Mapping these distributions back through the embedding matrix produces a soft token per position, in parallel and without any unrolled computation graph. Each input embedding is then replaced by a convex combination of its ground-truth embedding and the corresponding soft token. Soft-TF introduces no additional parameters, requires no external models or complex training curricula, and leaves both the loss and the inference procedure unchanged.

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.

We observe consistent improvements across 4 mathematical reasoning benchmarks and 7 open-source models spanning the Gemma (Team et al., 2024a;b; 2025a), Llama (Grattafiori et al., 2024) and Mistral (Jiang et al., 2023) families. Moreover, we also outperform latent reasoning methods without the need for external models or multi-stage training. Our results establish soft tokens as a highly practical ingredient for supervised post-training, independent of their viability for latent reasoning.

2 METHOD

2.1 PRELIMINARIES

We first establish the notation used throughout this paper. Let V be a tokenized vocabulary of size $|V|$. Let $S = [q_{1:M}, r_{1:N}]$ be an input sequence of length $L = M + N$, comprising M question tokens q and N response tokens r. Response tokens include both intermediate CoT steps c and final answer tokens a. The subsequence $Y = [r_{1:N}]$ represents the full response. $\texttt{LM}_\theta$ is a language model parameterized with θ acting as a next-token predictor, i.e., $\texttt{LM}_\theta(s_i, s_{<i})$ models the conditional probability for s_i given the preceding prefix $s_{<i}$. $\texttt{LM}_\theta$ consists of an input embedding layer W_{enc}, a transformer decoder Φ and a projection layer W_{dec} followed by a softmax operation, meaning $\texttt{LM}_\theta(\cdot) = \texttt{Softmax}(W_{dec} \cdot \Phi(W_{enc}(\cdot)))$. We write δ_v for the one-hot vector of token v and $e_v = W_{enc}\delta_v$ for its embedding, such that a sequence $s_{1:L}$ has an embedding matrix $E = [e_{s_1}, ..., e_{s_L}]$. We index positions in the full sequence by $i \in \{1, \dots, L\}$, meaning response tokens occupy positions $M < i \leq L$ and $y_j = s_{M+j}$. We denote by $\Delta^{|V|-1} := \{q \in \mathbb{R}^{|V|}_{\geq 0} : \mathbf{1}^\top q = 1\}$ the probability simplex over the vocabulary, and by $\mathcal{E} := \{e_v\}_{v \in V} \subset \mathbb{R}^d$ the set of vocabulary embeddings (assumed pairwise distinct), where d is the embedding dimension. We write $\text{conv}(\mathcal{E})$ for the convex hull of $\mathcal{E}$, and note that it is exactly the image of the simplex under the embedding layer: $\text{conv}(\mathcal{E}) = \{W_{enc}q : q \in \Delta^{|V|-1}\}$.

Teacher Forcing (TF). In transformer-based LLMs, TF is the implicit standard. The model receives the entire *ground-truth* sequence at once, and causal masking ensures that the prediction at position t is conditioned solely on the ground-truth tokens from positions 1 to $t-1$. The next-token prediction objective minimizes the negative log-likelihood of each token in the response sequence Y, conditioned on the *ground-truth* prefix rather than the model's own predictions:

$$\mathcal{L}(\theta) = -\,\mathbb{E}_\chi \left[\sum_{i=1}^{N} \log p_\theta(y_i \mid q_{1:M}, y_{<i}) \right] \tag{1}$$

where χ is the true distribution of sequences S. The same objective applies to each token during pretraining. Inference is autoregressive, meaning that generation is token-by-token. Under greedy decoding, this is expressed as:

$$\hat{y}_i = \arg\max_{y \in V} \; p_\theta(y \mid q_{1:M}, \hat{y}_{<i}) \tag{2}$$

Next, we start by explaining the deviation from TF in current methods that use soft tokens.

2.2 LATENT REASONING DEVIATES FROM TEACHER FORCING

In this section, we present the general framework for training-based continuous latent reasoning methods (Hao et al., 2025; Shen et al., 2025). In CoCoNut-style latent reasoning (Hao et al., 2025), the last-layer hidden states are used as continuous tokens. The following formulation is generic and applies to output-based latent reasoning methods. Let $S^{(0)} = [q_{1:M}]$ denote the question prefix, and $E^{(0)} = W_{enc}(S^{(0)})$ the corresponding embeddings. Latent tokens are generated iteratively by extending the embedding sequence and re-running the transformer. For $k \geq 0$, define:

$$h_M^{(k)} = \left[\Phi(E^{(k)})\right]_{M+k}, \quad E^{(k+1)} = [E^{(k)}, h_M^{(k)}].$$

Assuming the total number of latent tokens is P, the latent token sequence is:

$$r_{1:P}^{\text{latent}} = [h_M^{(0)}, h_M^{(1)}, \dots, h_M^{(P-1)}].$$

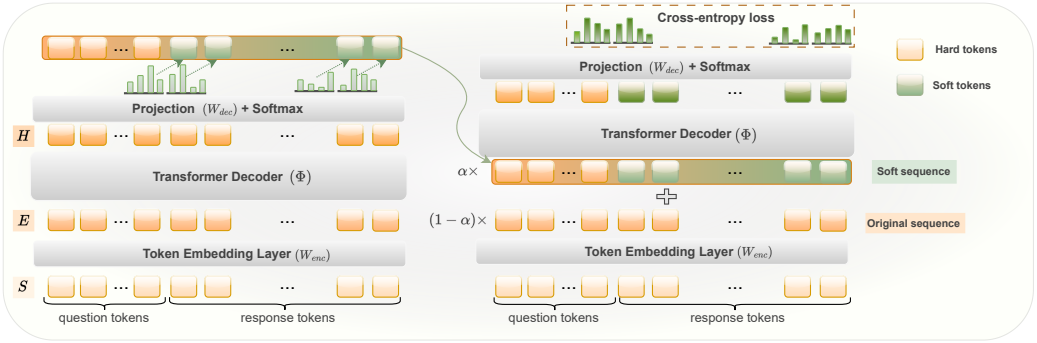


Figure 1: (*Left*) **Teacher-forced soft sequence computation.** Given an initial input sequence, we forward it through the LLM to obtain output probability vectors. These are used as weights for convex combinations of vocabulary tokens, forming soft tokens. (*Right*) **Training with Soft-TF.** We train the LLM with the next-token prediction objective on convex combinations of the original sequence embeddings and soft sequence embeddings. We call this framework Soft Teacher Forcing (Soft-TF).

Each latent token requires a full forward pass over the extended sequence, leading to a computational cost that grows with both sequence length and P. Finally, the full latent sequence is:

$$E_{\text{latent}} = [E^{(0)}, r^{\text{latent}}_{1:P}, W_{enc}(r_{P+1:N})],$$

where $r_{P+1:N}$ denotes the remaining discrete response tokens.

The loss function in this case is:

$$\mathcal{L}(\theta) = -\,\mathbb{E}_{\chi'}\left[\sum_{i=P+1}^{N} \log p_\theta(y_i \mid E^{(0)}, r^{\text{latent}}_{1:P}, E_{r_{P+1:i-1}})\right] \tag{3}$$

where χ' is the distribution of latent sequences, which has a continuous support. We can see from Equation (3), unlike Equation (1), that only the response part after the latent phase is supervised, and that each token is conditioned on the previous output-based embeddings instead of ground-truth data, which induces a growing deviation from TF. The drift increases with the latent sequence length P, which places the LLM in an out-of-distribution (OOD) regime since it was never trained on recursively generated hidden-state trajectories.

Compression: a goal or a necessity? The previous arguments suggest that compression is not merely desirable in latent reasoning models but necessary to mitigate the divergence induced by recursive latent generation. It is also empirically shown that increasing the number of latent tokens significantly harms the performance and leads to training instability (Hao et al., 2025; Wei et al., 2026). This is coupled with computational challenges, since gradients propagate through all latent steps, requiring backpropagation through a computation graph unrolled over P iterations.

2.3 Soft Teacher Forcing (Soft-TF)

We argue that, unlike latent reasoning methods, the effective use of soft tokens can be unlocked by maintaining the proximity to TF during fine-tuning, otherwise the LLM will be in an out-of-distribution regime. To achieve this, we train LLMs on random convex combinations of teacher-forced embeddings and soft tokens. Soft-TF consists of two main steps: *i) Teacher-forced soft sequence computation*, and *ii) training with Soft-TF*.

Teacher-forced soft sequence computation. Given training sequences of the form $S = [q_{1:M}, r_{1:N}]$, we perform a single parallel forward pass through the LLM to obtain the predicted distribution at every position, $p = [p_1, \ldots, p_L] = \texttt{LM}_\theta(S)$, where $p_i = p_\theta(\cdot \mid s_{\leq i})$ is the distribution over the token at position $i+1$. We use these to construct soft tokens at the response positions: for $M < i \leq L$,

$$e_i^{\texttt{soft}} = W_{enc}\, p_{i-1} = W_{enc}\, p_\theta(\cdot \mid s_{<i}), \tag{4}$$

while $e_i^{\texttt{soft}} = E_i$ at question positions $i \leq M$. We also consider a top-k variant, in which p_{i-1} is restricted to its k most probable tokens and renormalized before projection. Teacher-forced soft sequence computation is illustrated in Figure 1 (*Left*). Two key properties distinguish this from latent reasoning methods (Hao et al., 2025; Shen et al., 2025), where soft tokens are generated recursively (each feeding back as input for the next). First, our construction is *teacher-forced*: causal masking ensures that the soft token at position i is conditioned on the ground-truth prefix $s_{<i}$. The recursive construction loses this immediately, as discussed in Section 2.2. Second, our construction is *parallel*: all soft tokens are obtained using a single forward pass, requiring only one additional pass regardless of sequence length. Conversely, the recursive alternative demands one pass per soft token and forces all intermediate activations to be retained in memory for backpropagation. We ablate the number of forward passes K in Section 3.3 and find that performance degrades as K grows beyond a small number of passes, so the single pass is not only the cheaper option but also more effective. We conjecture that this reflects the deviation from the teacher-forcing algorithm used during pretraining, and that remaining close to it matters when fine-tuning.

Training with Soft-TF. Once the sequence with soft tokens is computed, we train on convex combinations of ground-truth embeddings and soft tokens. This introduces another level of proximity to teacher forcing on top of the teacher-forced soft sequence computation. Specifically, the final sequence for training is computed as:

$$E^{\texttt{final}} = \alpha \underbrace{E^{\texttt{soft}}}_{\text{Soft term}} + (1-\alpha) \underbrace{E}_{\text{TF term}}, \quad (5)$$

where $\alpha \sim \text{Beta}(\beta, \beta)$ is sampled at each iteration, with hyperparameter $\beta \in (0, \infty)$. This sequence ($E^{\texttt{final}}$) is then forwarded through Φ, and the final probabilities for supervision are computed by projecting the hidden states via W_{dec} followed by `Softmax` (*cf.* Figure 1 (*Right*)). Algorithm 1 summarizes the Soft-TF process.

Algorithm 1 Soft Teacher Forcing (Soft-TF)

Input: original input sequence $S \in \mathbb{N}^{B \times L}$, loss mask `Mask` (`True` for response tokens, `False` else), language model $\texttt{LM}_\theta$.

$E = W_{enc}(S)$
$p = \texttt{stopgrad}\big(\texttt{Softmax}(W_{dec} \cdot \Phi(E))\big)$
$E^{soft} = (1 - \texttt{Mask}) * E + \texttt{Mask} * W_{enc} p_{\texttt{shift}}$

$\alpha \sim \texttt{Beta}(\beta, \beta)$
$E^{\texttt{final}} = \alpha E^{\texttt{soft}} + (1-\alpha) E$
$\hat{p} = \texttt{Softmax}(W_{dec} \cdot \Phi(E^{\texttt{final}}))$
$\mathcal{L} = \texttt{Mask} * \texttt{cross-entropy}(\hat{p}, p_{\texttt{GT}})$

We highlight that training with $\alpha = 0$ reduces Soft-TF to TF. Moreover, we investigate if the second level of proximity to TF is necessary, or if training with $\alpha = 1$ suffices. Experimentally, we show that it is better to train on convex combinations of E and $E^{\texttt{soft}}$, and that randomizing α during training is better than fixing its value.

While Equation (5) is written as a mixture of two embeddings, W_{enc} is linear, meaning the operation is better understood one layer earlier, i.e., on the simplex. Indeed, for every response position $M < i \leq L$ and every $\alpha \in [0, 1]$,

$$E_i^{\texttt{final}} = W_{enc}\, q_i, \quad (6)$$

where $q_i := \alpha\, p_{i-1} + (1-\alpha)\, \delta_{s_i} \in \Delta^{|V|-1}$. Consequently $E_i^{\texttt{final}} \in \text{conv}(\mathcal{E})$.

Thus, Soft-TF does not arbitrarily corrupt an embedding; rather, it feeds the model the embedding of a perturbed distribution. It retains mass $1 - \alpha + \alpha\, p_{i-1}(s_i)$ on the ground-truth token and redistributes the remainder according to the model's own beliefs. Training with $\alpha = 0$ (*i.e.* $q_i = \delta_{s_i}$) reduces Soft-TF to TF. Importantly, Soft-TF ensures that the transformer does not encounter a point outside the convex hull of its vocabulary embeddings. This property is broken when last-layer hidden states are used as soft tokens, since they are outputs of Φ rather than images of $\Delta^{|V|-1}$ under W_{enc} (i.e., they are not of the form $W_{enc}q$ for any distribution q over the vocabulary).

2.4 Inference

At inference, given an input sequence $Q = [q_1, q_2, ..., q_M]$, there are multiple ways to perform generation through autoregression. The first is the (i) *standard* discrete inference, exemplified by greedy decoding where, at each step, the most probable token is used as the next input. The second follows Zhang et al. (2025), where soft tokens are generated until the maximum prediction probability corresponds to an `<end-of-thought>` token. We call this setting (ii) *soft inference*. More

generally, we can perform inference using both approaches:

$$\hat{e}_i = \gamma \hat{e}_i^{\texttt{soft}} + (1-\gamma)\hat{e}_i^{\texttt{greedy}}, \tag{7}$$

where $\gamma \in [0, 1]$ is a constant, e.g., 0.5. We call this setting (iii) *mixed* inference. A fourth type of inference we consider is the (iv) *hybrid* inference, where we perform *mixed* sampling ($\gamma = 0.5$) for K steps; K being the number of forward passes used for soft sequence computation—$K = 1$ by default—and *standard* greedy decoding after. These strategies are illustrated in Appendix C.

Next, we present experiments with Soft-TF. Across different inference strategies, we find that standard discrete inference (i.e., $\gamma = 0$) works best in most cases (cf. Section 3.3), so we adopt as a default. This aligns with recent findings by Butt et al. (2026), where soft tokens with added noise are used to train instruction-tuned models with Reinforcement Learning (RL), yet inference is discrete.

3 Experiments

3.1 Experimental setup

Datasets. We evaluate our method mainly on mathematical datasets with reasoning traces, including GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MathQA (Amini et al., 2019) and AQuA (Ling et al., 2017). We train on the training set of each dataset and report the accuracy of the test sets. GSM8K, MATH, MathQA, AQuA are composed of 7 473, 12 000, 29 837 and 97 467 training samples respectively. The test sets contain 1 319, 500, 2 985 and 254 samples respectively.

Implementation details. We report main results using the Gemma2-2B (Team et al., 2024b) model, but also corroborate our approach with Gemma3-1B, Gemma3-4B, Gemma2-9B, Llama3.2-1B, Llama3.2-3B, and Mistral-7B-v0.3. Since our work is centered on SFT, we use pretrained checkpoints of these models. We run all the experiments with 5 seeds and report the mean and standard deviation. More implementation details and paired significance test are provided in Appendix A.

Prompt formatting. For all datasets, the question is followed directly by the rationale, and the answer is separated using a `<start-of-answer>` token. For MathQA and AQuA, which are multiple choice questions, the options/choices are inserted between the question and the rationale.

Metrics. At inference, we consider as output all tokens generated after the `<start-of-answer>` token. Exact match accuracy (%) is used as metric for evaluation. We adopt greedy decoding, i.e., tokens with top-1 probability are considered during autoregressive generation.

3.2 Main Results

Baselines. We compare Soft-TF to the following baselines:

- TF (No-CoT), trained on <*question*, *answer*> pairs in a supervised manner using TF.
- TF (CoT), trained on <*question*, *CoT*, *answer*> triples in a supervised manner using TF.
- Label smoothing (Szegedy et al., 2016; Pereyra et al., 2017), a standard output-side regularizer which was first proposed in computer vision then used in neural machine translation (Vaswani et al., 2017). Specifically, we train with TF on <*question*, *CoT*, *answer*> triples, where the one-hot target δ_{y_i} is replaced by $(1-\lambda)\delta_{y_i} + \lambda u$ with u uniform over V. This is the output-side counterpart of Soft-TF, which softens the input instead.
- Scheduled sampling (Mihaylova & Martins, 2019) is a method proposed to mitigate exposure bias, i.e., the mismatch between conditioning on ground-truth previous tokens during training and on model-generated tokens during inference. During training, scheduled sampling randomly replaces ground-truth previous tokens with model-generated tokens according to a schedule that controls the probability of using generated tokens.
- Original CoCoNut (Hao et al., 2025), a latent reasoning approach that progressively removes the explicit CoT, and trains the model to perform the corresponding reasoning in latent space. We divide the tokenized CoT into 3 parts and remove one part per stage of training.
- A modified variant CoCoNut† without multi-stage training nor compression, where reasoning is trained using hidden states but supervised explicitly with CoT. This baseline isolates the effect of the compression procedure in latent reasoning. More details are in Appendix B.

Table 1: Performance (accuracy %) of Soft-TF compared against baselines on GSM8K, MATH, MathQA and AQuA datasets. All methods fine-tune Gemma2-2B pretrained checkpoint, except SoftThinking, which is an inference-only method using the TF (CoT) checkpoint. For CoCoNut†, we report the best decoding strategy per dataset (see Appendix B.2).

METHOD	GSM8K	MATH	MATHQA	AQUA
TF (NO COT)	14.00± 0.28	12.40± 0.32	49.92± 0.49	40.71± 1.94
TF (COT)	44.70± 0.89	14.44± 0.91	59.86± 0.80	38.19± 4.30
COCONUT (STAGE 1)	26.03± 0.69	9.32± 1.07	46.55± 1.01	32.13± 2.06
COCONUT (STAGE 2)	19.36± 0.78	6.80± 0.47	48.17± 0.88	31.18± 0.85
COCONUT (HAO ET AL., 2025)	13.59± 0.56	10.08± 1.01	45.16± 1.12	34.96± 1.98
COCONUT† (HAO ET AL., 2025)	21.00± 1.25	0.92± 0.77	12.14± 2.28	12.05± 2.58
SOFTTHINKING (ZHANG ET AL., 2025)	41.62± 1.21	11.44± 0.77	57.71± 0.43	36.69± 4.00
SOFTTHINKING† (ZHANG ET AL., 2025)	44.70± 1.50	13.12± 0.98	54.69± 0.79	37.64± 3.27
LABEL SMOOTHING	41.53±0.37	15.16±1.05	59.73±0.87	39.05±2.20
SCHEDULED SAMPLING (MIHAYLOVA & MARTINS, 2019)	46.38±1.02	15.04±1.76	59.78±1.01	39.45±2.32
SOFT-TF (OURS)	**49.28**± 0.59	**16.20**± 1.33	**61.49**± 0.96	**42.06**± 1.95

- SoftThinking which uses soft tokens only during inference, similar to (Zhang et al., 2025). Here we use the TF (CoT) model.
- SoftThinking†, where we extend SoftThinking by applying it during training.

Illustrations of CoCoNut† and SoftThinking† are provided in Appendix C. Results in Table 1 show that Soft-TF is the only method that consistently outperforms TF (CoT) across all datasets.

We observe that CoCoNut not only underperforms TF (CoT) but also TF (No-CoT), despite using TF (CoT) weights as initialization. These results are in line with Hao et al. (2025) showing limitations of CoCoNut for model sizes beyond GPT-2 (Radford et al., 2019) ($\sim$ 117M parameters). Additionally, by projecting the hidden states and decoding them, we found that the corresponding top-1 predictions are not interpretable and exhibit repetitive patterns. More details are provided in Appendix B.1.

CoCoNut† removes the compression and multi-stage training, obtaining soft tokens from a single forward pass but using hidden states instead of convex combinations of vocabulary embeddings (with $\alpha = 1$ in Equation (5)). It improves over CoCoNut on GSM8K but is much worse elsewhere, which supports prior findings that latent reasoning requires compression (Hao et al., 2025; Wei et al., 2026). Results with different sampling strategies are in Appendix B.2.

SoftThinking (Zhang et al., 2025) is applied during inference using the TF (CoT) model, providing additional evidence to Wu et al. (2026) that vanilla convex reasoning does not lead to improvements. We also evaluate SoftThinking†, a variant that applies SoftThinking during training: Equation (4) remains unchanged, while $\alpha = 1$ in Equation (5). This variant shows improvement over SoftThinking, hinting at the potential of using soft tokens for training. However, it underperforms Soft-TF, highlighting the importance of training with a convex combination of soft tokens and teacher-forced embeddings (i.e., random α in Equation (5)). This variant therefore serves as a direct ablation of the mixing step: it matches the TF (CoT) baseline on GSM8K (44.70), and the $+4.58$ gain appears only once the soft sequence is mixed with the teacher-forced one.

Softening the target instead of the input. As discussed in Section 2.3, Soft-TF replaces the one-hot input δ_{s_i} by a mixture of the ground-truth token and the model's own prediction, which is the input-side analogue of label smoothing on the output. Label smoothing improves over TF (CoT) on MATH (15.16 against 14.44) and AQuA (39.05 against 38.19) but not on GSM8K (41.53 against 44.70) or MathQA (59.73 against 59.86), whereas Soft-TF improves on all four datasets. Softening the target is therefore not sufficient, and the gain does not come from smoothing per se.

Comparison to scheduled sampling. Empirically, scheduled sampling (Bengio et al., 2015; Mihaylova & Martins, 2019) has been shown to improve only marginally and occasionally upon teacher forcing. In our experiments, scheduled sampling performs generally better than label smoothing, but still lags behind Soft-TF on all four datasets. Since both methods use the same top-5 soft tokens (hyperparameters in Appendix B), this comparison isolates the effect of interpolating rather than replacing ground-truth embeddings, and whether a remplacement schedule is required. The Bernoulli

Table 2: Accuracy (%) on the GSM8K dataset using different models.

METHOD	TF (NO COT)	TF (COT)	SOFT-TF (OURS)
GEMMA3-1B	6.82±0.42	16.77±0.50	**19.38**±0.60 ↑ +2.61
GEMMA2-2B	14.00±0.28	44.70±0.89	**48.52**±1.24 ↑ +3.82
GEMMA3-4B	18.54±0.48	54.86±0.50	**57.95**±1.23 ↑ +3.09
GEMMA2-9B	30.37±0.44	73.89±0.77	**77.07**±1.19 ↑ +3.18
LLAMA3.2-1B	5.90±0.26	28.23±0.88	**30.33**±0.59 ↑ +2.10
LLAMA3.2-3B	13.18±0.26	48.58±1.20	**52.66**±1.12 ↑ +4.08
MISTRAL-7B-V0.3	13.91±0.95	57.57±0.80	**60.82**±0.66 ↑ +3.25

ablation in Table 4b, which replaces each embedding by its soft token with a fixed probability of 0.5, points the same way: both replacement schemes fall below Soft-TF.

Soft-TF with other model sizes. We train Soft-TF using Gemma3-1B, Gemma3-4B and Gemma2-9B, Llama3.2-1B, Llama3.2-3B, and Mistral-7B-v0.3 models. Table 2 shows that Soft-TF consistently leads to improvements across different LLMs.

3.3 DESIGN CHOICES & ABLATIONS

We ablate Soft-TF design choices (α, soft token type, and computation) using Gemma2-2B on GSM8K unless noted otherwise.

Which soft tokens are better? Prior work builds soft tokens either as convex combinations of vocabulary embeddings (Zhang et al., 2025; Wu et al., 2026; Butt et al., 2026) or from last-layer hidden states (Hao et al., 2025; Shen et al., 2025). We test the latter within our framework, which means using the last-layer hidden states at position $i-1$ in place of $W_{enc}p_{i-1}$. Table 3 shows that hidden states underperform convex soft tokens and do not improve over TF (CoT) on MATH. We also compare against argmax (top-1) tokens, which replace p_{i-1} with $\delta_{\hat{y}}$ in Equation (4). This variant remains a convex combination of two vocabulary embeddings; only the direction of the perturbation differs, pointing towards the single most likely token rather than towards the full predictive distribution. On GSM8K the two are close (48.35 against 49.28), while on MATH the argmax variant falls below the TF (CoT) baseline (13.76 against 14.44) and soft tokens reach 16.20.

Table 3: Ablation study on soft tokens in Soft-TF.

Method	GSM8K	MATH
TF (CoT)	44.70± 0.89	14.44± 0.91
hidden	46.81± 1.11	14.40± 0.94
argmax	48.35± 1.06	13.76± 1.11
convex-full	48.52± 1.24	15.48± 1.50
convex-top-5	**49.28**± 0.59	**16.20**± 1.33

Teacher-forced *vs* sequential soft tokens. Soft sequence computation is teacher-forced, i.e., performed in a single forward pass through the model. As discussed in Section 2.3, this keeps every soft token conditioned on the ground-truth prefix and avoids multiple forward passes. To test this choice, we instead compute soft tokens sequentially, feeding each one back as input to produce the next:

$$e_{r_k} = W_{enc}\,\texttt{Softmax}\big(W_{dec}\,\Phi(E^{(0)}, e_{r_1}, \ldots, e_{r_{k-1}})\big), \quad (8)$$

where $e_{r_1} = W_{enc}\,p_\theta(\cdot \mid q_{1:M})$. $E^{(0)} = W_{enc}(q_{1:M})$ and Φ attends over the question and all previous soft tokens, its output being taken at the last position. This resembles CoCoNut (Hao et al., 2025) except that soft tokens are convex combinations of vocabulary embeddings rather than last-layer hidden states. The number of forward passes K sets the length of the sequential dependency, and $K = 1$ recovers teacher-forced soft sequence computation. Increasing K removes the teacher-forcing signal from a growing part of the input. Figure 2 shows that performance degrades as K grows beyond a small number of passes. We illustrate sequential computation in Appendix C.

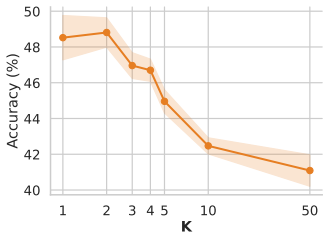
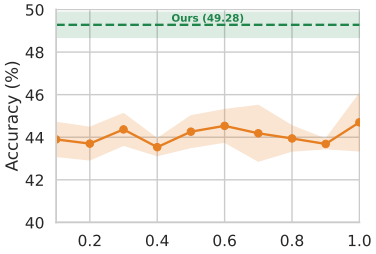


Figure 2: Ablation on the number of forward passes K for soft sequence computation. The x-axis is on a log scale.

Random vs. constant α. In this experiment, we investigate whether it is necessary to use a randomly sampled α during training. Thus, we apply Soft-TF training using constant values of α ranging from

Table 4: **Choice of α.** (a) Constant *vs.* random α: Soft-TF with constant $\alpha \in]0, 1]$. The dashed horizontal line shows accuracy (%) with $\alpha \sim \text{Beta}(0.1, 0.1)$ as a reference. Shaded regions show the standard deviation over 5 runs. (b) Sampling distribution of α: $\text{Beta}(0.1, 0.1)$ vs. $\text{Uniform}[0, 1]$ vs. $\text{Bernoulli}(0.5)$ with $\alpha \in \{0, 1\}$, with the TF (CoT) baseline as reference.



(a) Constant *vs.* random α

	Method	GSM8K	MATH
	TF (CoT)	44.70± 0.89	14.44± 0.91
Full	Beta	**48.52**± 1.24	**15.48**± 1.50
	Uniform	47.61± 1.04	15.12± 0.94
	Bernoulli	47.92± 1.01	13.52± 0.97
top-5	Beta	**49.28**± 0.59	**16.20**± 1.33
	Uniform	47.52±0.66	13.60± 2.02
	Bernoulli	48.63± 0.60	13.44± 1.40

(b) Sampling distribution of α

0 to 1 in increments of 0.1. Table 4a shows that performance significantly lags behind that obtained using continuously sampled random α values during training.

α sampling distribution. By default, α (Equation (5)) is sampled from $\text{Beta}(0.1, 0.1)$, a U-shaped distribution concentrated near 0 and 1. We compare it to $\mathcal{U}[0, 1]$ and to $\text{Bernoulli}(0.5)$, which $\text{Beta}(0.1, 0.1)$ approximates continuously, where each embedding is either ground-truth or soft token. As shown in Table 4b, Beta outperforms the uniform law with both full and top-5 soft tokens, and also outperforms Bernoulli, more significantly on MATH than GSM8K. Performance is thus non-monotone in the interior mass on $[0, 1]$, which we report as an empirical finding.

Sampling strategies. As previously mentioned in Equation (7), inference can be performed using discrete tokens, soft tokens, or a combination of the two. Here, we present four sampling strategies and show that there is no clear advantage to using soft tokens during inference. The inference strategies are as follows: (*i*) Discrete greedy decoding, which we call *Standard*. This corresponds to $\gamma = 0$ in Equation (7) and is the default strategy used throughout the paper. (*ii*) *Soft* sampling, which corresponds to $\gamma = 1$. (*iii*) *Mixed* sampling, which corresponds to $\gamma = 0.5$ and consists of mixing greedy decoding embeddings with soft embeddings. (*iv*) *Hybrid* sampling, which uses *Mixed* sampling ($\gamma = 0.5$) for K steps and *Standard* greedy decoding thereafter. This strategy is motivated by the fact that, to mimic training, mixed sampling should be performed for the same number of forward steps used for soft sequence computation during training, and greedy decoding for the rest. Results in Table 5 demonstrate that *standard* sampling works well within our framework, with hybrid sampling yielding a slight improvement at the cost of higher standard deviation.

Table 5: Accuracy (%) on the GSM8K dataset for different sampling strategies.

METHOD	TF (NO COT)	TF (COT)	SOFT-TF (OURS)
STANDARD ($\gamma = 0$)	14.00± 0.28	44.70± 0.89	**48.52**± 1.24
SOFT ($\gamma = 1$)	13.62± 0.41	41.62± 1.21	**43.31**± 0.84
MIXED ($\gamma = 0.5$)	14.10± 0.26	43.99± 1.05	**46.50**± 1.00
HYBRID	14.06± 0.27	44.49± 1.29	**48.72**± 1.46

Comparison to Noisy TF. Does Soft-TF only have a regularization effect from adding noise to TF embeddings? To examine this, we compare Soft-TF to Noisy TF (CoT), a baseline inspired by (Jain et al., 2024), where we add noise to token embeddings during training. Specifically, given an input embedding e_i, we augment it with Gaussian noise $e_i^{\text{noisy}} := e_i + \sigma\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and σ is a scalar controlling the noise level. Results in Table 6 show that, despite slightly improving accuracy for some values of σ on GSM8K, Noisy TF (CoT) lags well behind Soft-TF at every noise level, indicating that the effect of using soft tokens cannot be replaced by noise.

Table 6: Comparison to Noisy TF. Accuracy (%) is reported across different noise levels σ.

METHOD		GSM8K	MATH
TF (COT)		44.70± 0.89	14.44± 0.91
	NOISE LEVEL (σ)		
NOISY TF (COT)	0.1	44.50± 1.13	14.04± 1.02
	0.3	45.23± 0.30	14.44± 0.41
	0.5	45.44± 0.65	14.12± 0.84
	0.7	45.55± 1.03	13.72± 1.79
	0.9	45.00± 1.73	14.16± 1.05
	2.0	43.00± 0.69	12.96± 1.28
	5.0	19.97± 0.16	3.56± 0.61
SOFT-TF (OURS)		48.52± 1.24	15.48± 1.50
SOFT-TF (OURS) (TOP-5)		**49.28**± 0.59	**16.20**± 1.33

4 Related Works

Soft tokens for latent reasoning. Soft tokens have been used to replace discrete chain-of-thought with reasoning in a continuous space. iCoT (Deng et al., 2024) starts from a model fine-tuned on explicit CoT and gradually removes CoT tokens through a curriculum, internalizing the reasoning into the weights. CoCoNut (Hao et al., 2025) follows the same recipe but replaces each removed step with last-layer hidden states fed back as input, and CODI (Shen et al., 2025) self-distills an explicit-CoT teacher into such a latent student by aligning hidden activations. SoftCoT (Xu et al., 2025) generates instance-conditioned soft tokens with an auxiliary language model and a trainable projection, while Su et al. (2025) map CoT tokens to discrete latent codes with a VQ-VAE. SoftThinking (Zhang et al., 2025) instead applies convex combinations of vocabulary embeddings at inference only; Wu et al. (2026) show that this largely collapses to greedy decoding and that stochasticity is needed to recover a benefit. Two limitations recur across this line of work. First, the methods rely on multi-stage curricula, distillation or auxiliary modules, and still underperform the SFT baseline they build on for real-world mathematical reasoning (Cobbe et al., 2021). Second, latent reasoning requires compression: adding latent tokens degrades accuracy and destabilizes training, with latents collapsing to a narrow set of decoded tokens (Hao et al., 2025; Wei et al., 2026) (which we also observe, see Appendix B.1). Closest to our setting, Butt et al. (2026) train instruction-tuned models with RL on noisy soft tokens that replace the discrete CoT in model-generated rollouts, which requires sequential generation, and decode discretely at inference. In contrast, Soft-TF operates in the supervised setting: soft tokens are computed in a single parallel pass over the ground-truth CoT, are mixed with the ground-truth embeddings rather than replacing them, and are conditioned on the ground-truth prefix. The two approaches are complementary: Soft-TF could serve as the SFT stage before such RL training.

Soft tokens to improve SFT. We use soft tokens not to replace the CoT but to improve supervised fine-tuning on it. As in SoftThinking (Zhang et al., 2025), our soft tokens are convex combinations of vocabulary embeddings, but they are used only during training, computed in a single teacher-forced forward pass, and mixed with rather than replacing ground-truth embeddings. Soft-TF thus requires no recursive generation, compression, auxiliary module, distillation or curriculum, and inference remains standard discrete decoding. It also differs from mixup (Zhang et al., 2018; Sun et al., 2020), which interpolates two examples and their labels for data augmentation: Soft-TF mixes each token's ground-truth embedding with the model's own prediction and leaves the target unchanged.

Scheduled sampling. Our work is reminiscent of scheduled sampling (Bengio et al., 2015), which reduces the mismatch between teacher-forced training and autoregressive inference by randomly feeding the model either ground-truth tokens or its own predictions. Closest to ours is Mihaylova & Martins (2019), who adapt it to transformers using two decoder passes and feed the second pass either argmax predictions or mixtures of embeddings weighted by the predicted distribution, following a schedule that decreases the probability of choosing the ground-truth tokens across training. Soft-TF uses the same soft tokens but, rather than replacing ground-truth embeddings according to a schedule, interpolates between the two with a coefficient drawn from a fixed distribution, so the ground-truth token (i.e., the teacher forcing part) remains partly present at nearly every position.

5 Conclusion

We present Soft Teacher Forcing (Soft-TF), a training algorithm that leverages soft tokens to improve SFT of LLMs. Soft-TF consists of two components. The first is teacher-forced soft sequence computation, which performs a single forward pass through the model to obtain output probabilities used to compute soft tokens. The second is training on convex combinations of teacher-forced and soft tokens. Extensive experiments demonstrate the effectiveness of our framework. We hope Soft-TF will help rethink training LLMs with soft tokens. Future work may explore applying Soft-TF to other stages of language model training, for instance pretraining and RL post-training.

Limitations. Soft-TF requires one additional forward pass per training step, costing $+42\%$ wall-clock time and $+14\%$ peak memory (Table 8), a single extra pass regardless of sequence length, so the relative overhead does not grow with the number of soft tokens, while inference cost is unchanged. Our evaluation is confined to mathematical reasoning datasets, and whether Soft-TF improves SFT in other domains, such as instruction following or code generation, remains untested. All models are fine-tuned with LoRA and have at most 9B parameters.

AI USE STATEMENT

In this work, we used a large language model to provide feedback on a draft of the manuscript, including comments on how our results were presented, and what were the unclear points in the paper ; this use falls under required disclosure. Generative AI did not contribute to developing the method, formulating hypotheses or mathematical claims, implementing the method, running the experiments, or generating or processing data, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for readability (grammar and phrasing), to draft revisions of parts of the text. We have reviewed all AI-assisted work: all revised and reformulated text was checked by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2357–2367, 2019.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9JjKTp8Jmy.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, 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 E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. Neftune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *https://arxiv.org/abs/2310.06825*, 2023.

Michael I Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 8, 1986.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison 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.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167, 2017.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Tsvetomila Mihaylova and André FT Martins. Scheduled sampling for transformers. *arXiv preprint arXiv:1906.07651*, 2019.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604. PMLR, 2018.

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*, pp. 677–693, 2025.

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. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hYfOPXrbUr.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. Mixup-transformer: Dynamic data augmentation for nlp tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3436–3440, 2020.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024a.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025a.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Junhong 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. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *ACL (1)*, 2025. URL https://aclanthology.org/2025.acl-long.1137/.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=yLClGs770I.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ByQdHPGKgU.

Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6xHJ37MVxxp.

A ADDITIONAL EXPERIMENTAL DETAILS

Implementation details. All models are fine-tuned with low-rank adaptation (LoRA) (Hu et al., 2022) of rank 64, a batch size of 8, and a cosine learning-rate schedule with linear warmup over the first 10% of training iterations. We use Adafactor (Shazeer & Stern, 2018) for Gemma models and AdamW (Loshchilov & Hutter, 2019) for Llama and Mistral models. The peak learning rate is 10^{-2} for all Gemma models, 2×10^{-4} for Llama3.2-1B, 10^{-4} for Llama3.2-3B, and 5×10^{-5} for Mistral-7B-v0.3. These values are tuned for the TF (CoT) baseline and reused unchanged for Soft-TF, so any benefit from learning-rate tuning goes to the baseline rather than to our method.

Training budget and model selection. The purpose of our comparison is to determine whether Soft-TF yields a real improvement over standard SFT, rather than an improvement that results from a particular training configuration. To this end, we aim to get the most out of every method, including the baselines, and compare each method at its best. Indeed, methods with different optimization dynamics reach their best performance after different numbers of iterations, and we want to avoid an under-tuned baseline. We thus treat the training budget as a hyperparameter to be tuned for each method, and tune it in exactly the same way for all methods.

Concretely, we sweep the total number of iterations for each method, using the same candidate sets for all methods. Each candidate is a separate run with its own cosine learning-rate schedule, whose warmup and decay are scaled to the total number of iterations. Each run is evaluated at its final iteration; we use no early stopping and no intermediate checkpoint selection. For each method, we report the candidate budget with the highest mean accuracy over 5 seeds. The results in the main paper are therefore best-versus-best comparisons: each method is evaluated at its own optimal budget, within the same search space and under the same selection rule.

Interestingly, we found that all baselines achieve their best results for the same total number of iterations in most cases. We use 5,000 iterations for GSM8K and MATH, 30,000 for MathQA, 50,000 for AQuA, and 10,000 iterations on Llama3.2-1B/GSM8K. They are identical for TF (CoT) and Soft-TF in the main experiments except in two cases: (i) on Gemma3-1B/GSM8K, TF (CoT) selects 5,000 iterations whereas Soft-TF selects 10,000; and (iii) on Gemma2-2B/MATH with top-5 soft tokens, TF (CoT) selects 5,000 iterations whereas Soft-TF selects 10,000.

These cases illustrate why the best-versus-best comparison is the appropriate one: fixing a common budget would favor whichever method prefers that budget. On Gemma3-1B/GSM8K, fixing the budget at the value selected by Soft-TF (10,000 iterations) would give a gain of $+3.75$ points, fixing it at 15,000 would give $+0.89$, and fixing it at the value selected by TF (CoT) (5,000) would give -0.53. The best-versus-best gain we report, $+2.61$, compares each method at its own optimum. Table 7 shows that TF (CoT) cannot achieve Soft-TF performance (19.38%) at any training budget.

Table 7: Gemma3-1B on GSM8K: accuracy (%) at different total iteration budgets (5 seeds). Each method is trained separately at each budget using its corresponding learning-rate schedule and evaluated at the final iteration. Bold marks the budget selected for each method.

ITERATIONS	TF (NO COT)	TF (COT)	SOFT-TF
5K	**6.82**±0.42	**16.77**±0.50	16.24±0.70
10K	6.07±0.46	15.63±0.74	**19.38**±0.60
15K	5.70±0.66	16.41±0.83	17.30±1.00

Soft-TF hyperparameters. The mixing coefficient α (Equation (5)) is sampled from $\mathrm{Beta}(\beta, \beta)$ with $\beta = 0.1$ (Zhou et al., 2021). It is sampled independently per token on GSM8K and AQuA, and once per sequence on MATH and MathQA. All reported results are means and standard deviations over 5 seeds, measured at the final training iteration.

CoCoNut. CoCoNut is trained for 5 epochs per stage, and is initialized from TF (CoT) weights. The CoT is tokenized and divided into three equal chunks, each representing a CoT "step". At each stage, one step is removed following Hao et al. (2025), and replaced with 2 latent tokens. Each stage is trained with 5 random seeds, and the best-performing checkpoint is used to initialize the next stage.

Prompt formatting. We provide details on prompt formatting for Gemma by showing examples from each training dataset:

GSM8K:

```
<start_of_turn>user
Question:
Natalia sold clips to 48 of her friends in April, and then she sold
half as many clips in May. How many clips did Natalia sell altogether
in April and May?
Answer:
<end_of_turn>
<start_of_turn>model
Natalia sold 48/2 = <<48/2=24>>24 clips in May.
Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.
#### 72<end_of_turn>
```

MATH:

```
<start_of_turn>user
How many vertical asymptotes does the graph of $y=\frac{2}{x^2+x-6}$
have?
<end_of_turn>
<start_of_turn>model
The denominator of the rational function factors into $x^2+x-6=(x-2)
(x+3)$. Since the numerator is always nonzero, there is a vertical
asymptote whenever the denominator is $0$, which occurs for $x = 2$
and $x=-3$. Therefore, the graph has $\boxed{2}$ vertical asymptotes.
#### 2<end_of_turn>
```

MATHQA:

```
<start_of_turn>user
there are 10 girls and 20 boys in a classroom . what is the ratio of
girls to boys ?
a ) 1 / 2 , b ) 1 / 3 , c ) 1 / 5 , d ) 10 / 30 , e ) 2 / 5
<end_of_turn>
<start_of_turn>model
if girls is 10 and boys is 20 , then 10 / 20 . so ratio of girls
to boys is = 10 / 20 = 1 / 2 answer : a
#### a<end_of_turn>
```

AQuA:

```
<start_of_turn>user
Question: Two friends plan to walk along a 43-km trail, starting at
opposite ends of the trail at the same time. If Friend P's rate is 15%
faster than Friend Q's, how many kilometers will Friend P have walked
when they pass each other?
Choices:
A)21
B)21.5
C)22
D)22.5
E)23
Answer:<end_of_turn>
<start_of_turn>model
If Q complete x kilometers, then P completes 1.15x kilometers.
x + 1.15x = 43
2.15x=43
x = 43/2.15 = 20
Then P will have have walked 1.15*20=23 km.
The answer is E.
#### E<end_of_turn>
```

In all datasets, #### is used as the <start_of_answer> token.

Training cost. Soft-TF adds one forward pass per training step. Since gradients do not backpropagate through the first forward pass, its activations need not be stored, incurring no additional backward cost. Table 8 reports measured wall-clock and peak memory under identical conditions (Gemma2-2B, GSM8K, 4×V100, 300 steps, batch size 8). The overhead is +42% in time and +14% in memory, for an accuracy gain up to 4.58 percentage points on GSM8K. This contrasts with continuous-CoT methods (Hao et al., 2025), whose cost grows with the number of latent tokens P (P+1 forward passes with backpropagation through an unrolled graph). Inference cost is identical to SFT, since Soft-TF decodes discretely.

Table 8: Measured training cost of Soft-TF against TF (CoT) on Gemma2-2B, GSM8K.

METHOD	SEC/STEP	PEAK MEM/DEVICE	THROUGHPUT
TF (COT)	0.882	21.42 GIB	9.07 SAMP/S
SOFT-TF	1.255	24.47 GIB	6.38 SAMP/S
OVERHEAD	+42%	+14%	0.70×

Paired significance tests. All comparisons in Table 1 use 5 seeds. Table 9 reports two-sided paired t-tests, pairing by seed, on the per-seed difference between Soft-TF and TF (CoT). The improvement is significant on GSM8K, MATH and AQuA. On MathQA, Soft-TF is higher on all five seeds (differences +3.35, +1.47, +0.57, +2.68, +0.07) but the test is borderline.

Table 9: Two-sided paired t-tests over 5 seeds for Soft-TF against TF (CoT) on Gemma2-2B.

DATASET	TF (COT)	SOFT-TF	Δ	p	95% CI
GSM8K	44.70±0.89	49.28±0.59	+4.58	0.0002	[+3.56, +5.60]
MATH	14.44±0.91	16.20±1.33	+1.76	0.0053	[+0.88, +2.64]
MATHQA	59.86±0.80	61.49±0.96	+1.63	0.058	[−0.09, +3.34]
AQUA	38.19±4.30	42.06±1.95	+3.87	0.0405	[+0.27, +7.47]

These tests should be read with caution. With $n = 5$ the t-test has four degrees of freedom, so the estimates of the standard error, and hence the p-values and confidence intervals, are themselves noisy, and the normality assumption cannot be checked at this sample size. This problem applies to all four rows, including those with small p-values, and not only to the borderline MathQA comparison. We report the tests because the per-seed differences are consistently positive across datasets, not as a precise quantification of the effect size.

B DETAILS ABOUT BASELINES

Baseline hyperparameters. For label smoothing, we use $\lambda = 0.1$. For scheduled sampling, the second decoder pass receives mixtures of the top-5 predicted token embeddings weighted by their renormalized probabilities, and the probability of feeding the ground-truth token decays linearly from 1 to 0.5 over training. These settings performed best in Mihaylova & Martins (2019).

B.1 DECODING THE LATENT COT TOKENS

As demonstrated in Table 1, CoCoNut collapses and underperforms the TF (No CoT) baseline. We present some examples of CoCoNut inference here. By projecting the hidden states used for latent reasoning at inference time and decoding them, we find that the corresponding top-1 predictions are not interpretable and exhibit repetitive patterns. This aligns with recent findings in Wei et al. (2026), which show that failed latent reasoning models generate only a narrow range of decoded tokens, a phenomenon they call "semantic homogenization". We observe the same pattern. Below are examples from the GSM8K test set:

```
Prompt: <start_of_turn>user
Sally received the following scores on her math quizzes: 50, 80, 80,
60, 40, 90, 100, 70, and 60. Find her mean score.<end_of_turn>
<start_of_turn>model
Text from latent:  The #### #### #### #### ####
Prediction: #### 72<end_of_turn>
Ground truth: #### 70<end_of_turn>
```

```
--------------------------------------------------------------------
Prompt: <start_of_turn>user
Tim enters a competition and has to try and guess the number of red
jelly beans in a jar. The jar has a square base and is 6 inches by 6
inches and 15 inches tall. He knows that a jelly bean is .15 cubic
inches. He also knows that they get about 80% efficiency in packing.
He estimates  30% of the jelly beans are red. The true number of red
jelly beans in the container is 900. How far off was his guess?
<end_of_turn>
<start_of_turn>model
Text from latent:  #### #### #### #### #### ####
Prediction: #### 30<end_of_turn>
Ground truth: #### 36<end_of_turn>
--------------------------------------------------------------------
Prompt: <start_of_turn>user
To heat during the winter, Ali ordered 850 kilos of coal. The coal is
delivered in 50 kg bags, costing $18 per unit. How much does his order
cost? <end_of_turn>
<start_of_turn>model
Text from latent:  #### #### #### #### #### ####
Prediction: #### 252<end_of_turn>
Ground truth: #### 306<end_of_turn>
```

Examples from the MATH test set:

```
Prompt: <start_of_turn>user
Let $z = 1+i$ and $w = \dfrac{3z+1}{5z+7}$. Find $|w|$.
<end_of_turn>
<start_of_turn>model
Text from latent:  We $ #### #### #### ####
Prediction: #### \dfrac{\sqrt{10}}{5}<end_of_turn>
\textbf{Ground truth}: #### \frac{5}{13}<end_of_turn>
--------------------------------------------------------------------
Prompt: <start_of_turn>user
Let $n$ be a positive integer.  Simplify the expression
\[\frac{(2^4 + \frac{1}{4})(4^4 + \frac{1}{4}) \dotsm [(2n)^4 +
\frac{1}{4}]}{(1^4 + \frac{1}{4})(3^4 + \frac{1}{4}) \dotsm [(2n - 1)
^4 + \frac{1}{4}]}.\]
<end_of_turn>
<start_of_turn>model
Text from latent:  The all #### #### #### ####
Prediction: #### 2n + 1<end_of_turn>
Ground truth: #### 8n^2 + 4n + 1<end_of_turn>
```

B.2 CoCoNut† results with different sampling strategies.

In Table 1, we include the results of CoCoNut†, a variant of CoCoNut without multi-stage training or compression. Here, we report in Table 10 the results of this variant for each sampling strategy described in Section 2.4. This experiment supports the idea that compression is necessary for latent reasoning methods and highlights a fundamental problem in using hidden states *alone* for reasoning (*i.e.*, $\alpha = 1$ in Equation (5)).

C Additional illustrations

To clarify the variants we test, we provide additional illustrations.

Sequential soft sequence computation. In Section 3.3, we present an ablation study contrasting teacher-forced soft-sequence computation ($K = 1$) with sequential computation. In the latter case, K determines the number of latent tokens that exhibit sequential dependency. Figure 3 illustrates the case where $K = 3$.

Table 10: Performance (accuracy %) of CoCoNut† with different sampling strategies on GSM8K, MATH, MathQA, and AQuA datasets. Mixed and soft decoding yield identical results on all four datasets.

METHOD	GSM8K	MATH	MATHQA	AQUA
TF (NO COT)	14.00± 0.28	12.40± 0.32	49.92± 0.49	40.71± 1.94
TF (COT)	44.70± 0.89	14.44± 0.91	59.86± 0.80	38.19± 4.30
COCONUT† (HYBRID)	20.68± 1.71	0.52± 0.44	0.96± 0.51	11.89± 4.48
COCONUT† (STANDARD ($\gamma = 0$))	21.00±1.25	0.92± 0.77	0.94± 0.69	11.18± 3.81
COCONUT† (MIXED ($\gamma = 0.5$))	1.00±0.38	0.04± 0.09	12.14± 2.28	12.05± 2.58
COCONUT† (SOFT ($\gamma = 1$))	1.00±0.38	0.04± 0.09	12.14± 2.28	12.05± 2.58

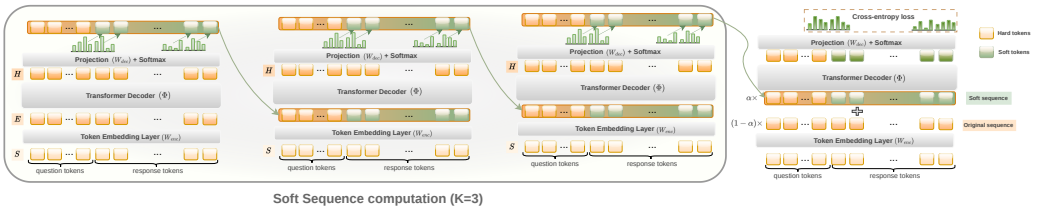
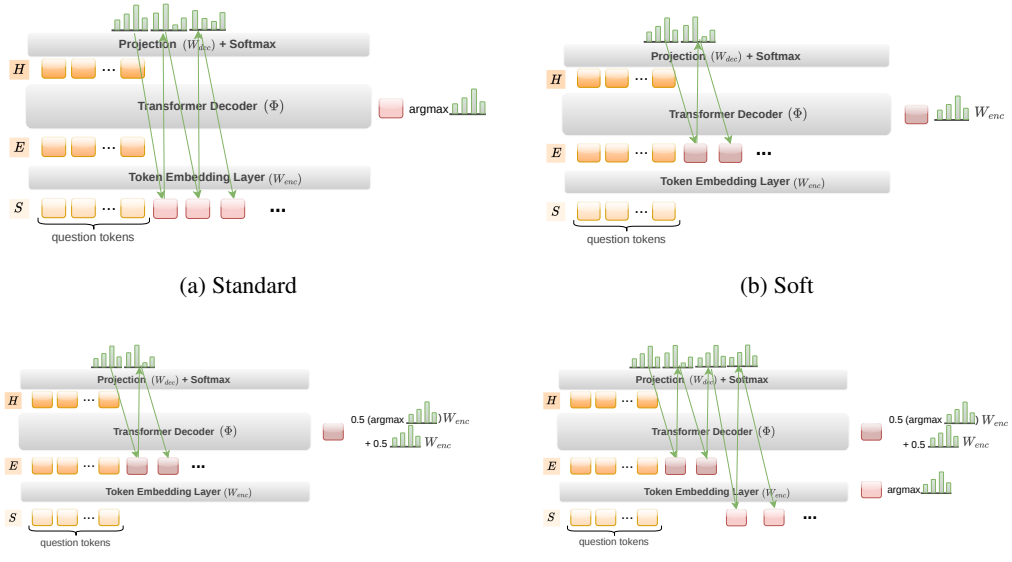


Figure 3: Sequential soft sequence computation with $K = 3$. In contrast to the teacher-forced soft sequence computation ($K = 1$, *cf.* Figure 1) adopted in this paper, sequential soft sequence computation requires multiple forward passes through the transformer. K reflects the number of soft tokens exhibiting sequential dependency.



(a) Standard (b) Soft

(c) Mixed (d) Hybrid ($K = 2$)

Figure 4: Inference strategies described in Section 2.4. (a) Standard discrete inference, (b) Soft inference, (c) Mixed inference, (d) Hybrid inference. Note that, for illustration only, we use $K = 2$ for hybrid inference, while the ablation in Table 5 uses $K = 1$.

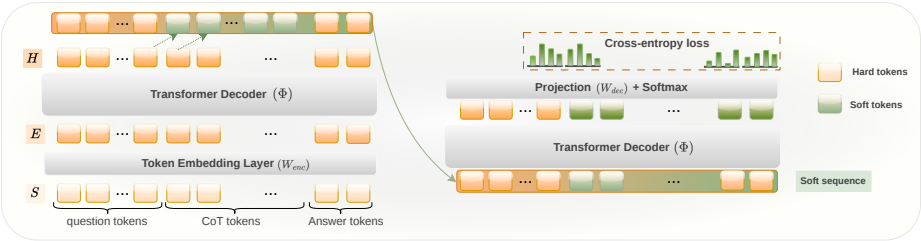


Figure 5: The CoCoNut† variant, which does not perform compression. Hidden states are used to construct the *teacher-forced* soft sequence, which is then used for training without resorting to the original sequence.

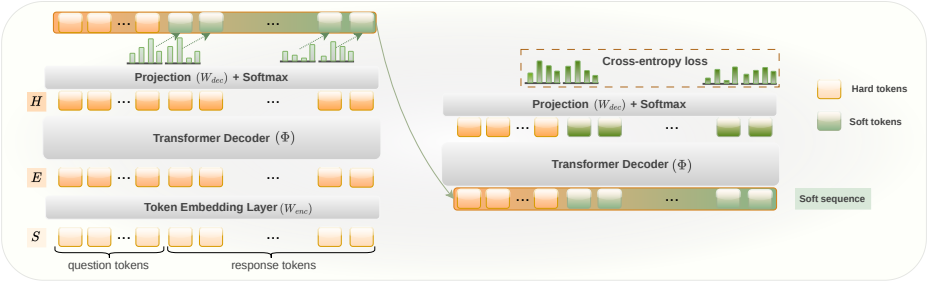


Figure 6: The SoftThinking† variant extends the original SoftThinking (Zhang et al., 2025) to training. The *teacher-forced* soft sequence is used for training without resorting to the original sequence.

Inference strategies. In Section 2.4, we present four inference strategies and ablate them in Table 5. We illustrate these strategies in Figure 4.

CoCoNut† and SoftThinking†. CoCoNut† and SoftThinking† are variants of CoCoNut (Hao et al., 2025) and SoftThinking (Zhang et al., 2025), respectively. The goal is to adapt these baselines to our setting, in which no compression or multi-stage training is used for CoCoNut, and SoftThinking is applied for both training and inference. CoCoNut† and SoftThinking† are illustrated in Figures 5 and 6, respectively. We note that for CoCoNut†, we use soft tokens only for the CoT part, as in CoCoNut. We also empirically find that CoCoNut† fails to generate an answer when soft sequence computation involves the answer part.