REAL-TIME STREAMING REASONING WITH TEMPORAL SELF-DISTILLATION POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time streaming reasoning requires language models to reason incrementally over progressively revealed inputs, producing segment-level reasoning under partial observations. However, GRPO assigns a uniform sequence-level advantage to all tokens, causing a temporal credit mismatch: redundant segments may be reinforced after correct answers, while useful local reasoning may be suppressed after incorrect ones. We propose **Temporal Self-Distilled Policy Optimization (TSDPO)**, a streaming-specific credit allocation method that treats the trajectory-level accuracy advantage as a fixed budget and redistributes it according to each segment's estimated answer support. TSDPO estimates segment-level answer support by comparing online and answer-conditioned likelihoods, then normalizes the support scores into budget-preserving temporal weights. Without external teachers or annotated reasoning trajectories, TSDPO assigns credit more faithfully across streaming segments. Experiments on Qwen3-1.7B and Qwen3-4B show that TSDPO improves accuracy over supervised streaming baselines and achieves both higher accuracy and shorter total generation than GRPO across four benchmarks.

1 INTRODUCTION

Large reasoning models (OpenAI, 2024; DeepSeek-AI, 2025) have achieved remarkable performance by scaling inference-time computation and generating long chain-of-thought trajectories (Wei et al., 2022). This paradigm emphasizes depth: given a complete problem, the model deliberates extensively before producing an answer. However, intelligence in real-world scenarios is not only reflected in solving difficult problems after full observation, but also in continuously interpreting and responding to evolving information in real time. Such capability is essential for interactive question answering (Choi et al., 2018), real-time video understanding (Chen et al., 2024), embodied navigation (Wei et al., 2025), and online monitoring (Yew et al., 2026), where waiting for the complete input may introduce unnecessary latency or even miss critical intervention windows.

Recent studies have begun to explore real-time streaming reasoning. StreamingThinker (Tong et al., 2025) first proposes a *think-while-reading* paradigm for LLMs, while TaYS (Zhang et al., 2026) extends stream-aligned CoT generation to video reasoning. These works demonstrate the feasibility of reasoning alongside input reception, but mainly focus on paradigm construction and often rely on constructed streaming CoT traces or teacher-guided supervision. How to further optimize streaming reasoning models without expensive trajectory annotations or strong online teachers remains underexplored.

A natural alternative is to optimize streaming reasoning with verifiable outcome rewards. GRPO (Shao et al., 2024) is attractive in this setting because it removes the need for explicit process annotations and estimates advantages from group-relative outcome scores. However, GRPO assigns a single sequence-level advantage to all tokens in a sampled response. This limitation is already coarse for ordinary reasoning, but becomes more problematic in streaming reasoning, where a trajectory is not a flat sequence generated after full observation. Instead, it consists of multiple reasoning segments, each conditioned on different partial observations. A correct final answer does not imply that every intermediate streaming segment was useful; conversely, an incorrect final answer may still contain locally useful intermediate reasoning.

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.

A correct final answer may reinforce redundant or even misleading intermediate segments, while an incorrect answer may suppress locally useful reasoning. This reveals a temporal credit mismatch: outcome rewards indicate whether the whole trajectory succeeds, but not how credit should be allocated across stream-aligned reasoning units.

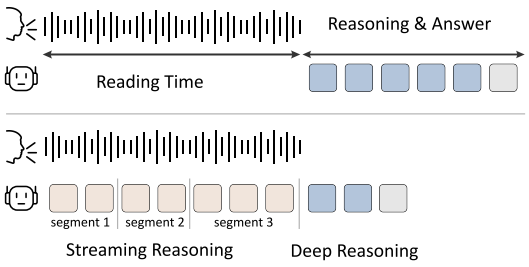

Figure 1: Overview of the two interaction reasoning paradigms. (Top) The read-then-think reasoning process. (Bottom) The real-time streaming reasoning process, which allows parallel processing of the reasoning and reading, thus significantly reducing the latency.

These observations raise a central question: *if GRPO can tell whether a streaming trajectory succeeds, can we further identify which local reasoning segments provide evidence for that success and redistribute advantages accordingly?* Inspired by recent self-distilled policy optimization methods (Hübotter et al., 2026; Yang et al., 2026) and a posterior-style view of privileged evidence (Vapnik & Vashist, 2009; Zhao et al., 2026), we propose Temporal Self-Distilled Policy Optimization (TSDPO). The key idea is to use privileged outcome information as posterior evidence to retrospectively evaluate the answer-evidential value of each local reasoning unit in a GRPO rollout.

Concretely, TSDPO decomposes each on-policy streaming rollout into sentence-aligned reasoning segments and estimates the answer-support value of each segment by comparing its likelihood under online visibility and answer-conditioned privileged visibility. The obtained support scores are normalized into temporal weights, which redistribute the original GRPO advantage across streaming segments. Thus, the outcome reward determines whether a trajectory should be reinforced or suppressed, while the temporal weights determine where the update should be applied. Importantly, TSDPO does not introduce extra process annotations or external teachers. The temporal weights are normalized within each trajectory, so the weighted segment-level advantages preserve the original trajectory-level advantage magnitude. This makes TSDPO a budget-preserving credit allocation method rather than an additional reward-shaping strategy.

Our contributions are summarized as follows:

- We identify a streaming-specific temporal credit assignment problem: in real-time reasoning, different segments are produced under progressively revealed contexts and should not receive identical credit from a single outcome reward.
- We propose TSDPO, a temporal self-distillation framework that leverages the intrinsic segmentation of streaming rollouts to estimate the answer support of each local reasoning unit from retrospective outcome evidence.
- We develop a budget-preserving advantage allocation objective that redistributes GRPO advantages across streaming segments according to their estimated answer support, without introducing extra process supervision or amplifying the outcome signal.

2 PRELIMINARY

Real-time Streaming Reasoning. Conventional reasoning follows a *read-then-think* paradigm: the model observes the complete context before generating any reasoning tokens. Let Q denote a question and $C = \{C_1, \ldots, C_T\}$ a context decomposed into T sequential sentences. Under this paradigm, each reasoning segment R_t is conditioned on the full context $C_{\leq T}$. In contrast, *real-time streaming reasoning* (Tong et al., 2025; Zhang et al., 2026; Chiang et al., 2025) generates reasoning with input reception. As each context sentence C_t arrives, the model may produce a local reasoning segment R_t under partial observation, or emit a skip token `<skip>` when the incoming sentence is uninformative. After the full context is available, the model performs a final *deep reasoning* phase.

This process factorizes as:

$$\mathcal{P}_{\text{streaming}} = P(R_q \mid Q) \prod_{t=1}^{T} P(R_t \mid Q, C_{\leq t}, R_{\leq t-1}) \cdot P(R \mid Q, C_{\leq T}, R_{\leq T}). \tag{1}$$

The first factor corresponds to the *streaming phase*: each R_t is generated before future sentences are seen. The second factor is the *deep phase*, where the model integrates the full context $C_{\leq T}$ and all streaming outputs $R_{\leq T}$ into a final response R.

GRPO for Reasoning. Group Relative Policy Optimization (GRPO; Shao et al., 2024) eliminates the need for a learned value function by estimating advantages from group-relative outcome scores. For each prompt q, GRPO samples a group of G responses $\{o_i\}_{i=1}^{G}$ from the current policy $\pi_{\theta_{\text{old}}}$, evaluates each response with a verifiable reward r_i (e.g., answer correctness), and computes a group-normalized advantage:

$$J_{\text{GRPO}} = \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{k=1}^{|o_i|} \left[C\left(\frac{\pi_\theta(o_{i,k} \mid h_{i,k})}{\pi_{\theta_{\text{old}}}(o_{i,k} \mid h_{i,k})}, A_i \right) - \beta\, \text{KL}(\pi_\theta(\cdot \mid h_{i,k}) \,\|\, \pi_{\text{ref}}(\cdot \mid h_{i,k})) \right], \tag{2}$$

where $C(\rho, A) = \min(\rho A, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A)$. Crucially, every token in o_i receives the same advantage A_i, regardless of its position or functional role in the trajectory.

On-policy Self-distillation. On-policy self-distillation (Zhao et al., 2026; Hübotter et al., 2026; Yang et al., 2026) provides dense token-level supervision without relying on an external teacher model. The key idea is to let the same policy act as both teacher and student under asymmetric information conditions. Specifically, the teacher observes *privileged information*, such as the ground-truth answer or a complete chain-of-thought trajectory, whereas the student operates under the standard observation setting. For each token position k, the discrepancy between the teacher and student token distributions is measured by

$$\delta_k = \text{KL}\Big(\pi_\theta^{\text{teacher}}(\cdot \mid h_k, h_k^{\text{priv}}) \;\Big\|\; \pi_\theta(\cdot \mid h_k) \Big). \tag{3}$$

This KL signal serves as a dense evidence signal indicating how privileged information changes the policy distribution at each token position.

3 Temporal Self-Distilled Policy Optimization

We propose Temporal Self-Distilled Policy Optimization (TSDPO), a segment-level policy optimization framework for real-time streaming reasoning. TSDPO keeps the outcome-driven stability of GRPO while introducing temporal credit allocation over streaming reasoning segments.

Streaming-aware Policy Optimization Objective Given a prompt q, a streaming context $C = \{C_1, \ldots, C_T\}$, and a verified answer a, the old policy samples a group of G streaming rollouts. Each rollout is parsed into a sequence of streaming reasoning segments and a final answer: $o_i = [R_{i,1}, R_{i,2}, \ldots, R_{i,T_i}, \hat{a}_i]$, where $R_{i,t}$ denotes a reasoning unit, T_i counts the intermediate streaming segments together with the final deep-reasoning unit, and $\hat{a}_i$ is the extracted answer. For an intermediate segment, $C_{\leq t}$ denotes the partial context available at its generation boundary; for the final deep-reasoning unit, it denotes the complete context. Full TSDPO assigns a separate support score and credit weight to each of these units.

Standard GRPO assigns one trajectory-level advantage to the entire rollout. This is coarse for streaming reasoning, because a trajectory contains temporally ordered segments generated under different partial observations. TSDPO therefore uses a segment-aware objective while preserving the original trajectory-level learning signal.

TSDPO optimizes the policy under the original streaming visibility and final answer privilege. Let $h_{i,t,k}^{\text{on}} = (q, C_{\leq t}, R_{i,<t}, r_{i,t,<k})$ denote the online history available when token $r_{i,t,k}$ is generated. The policy ratio is $\rho_{i,t,k}(\theta) = \pi_\theta(r_{i,t,k} \mid h_{i,t,k}^{\text{on}}) / \pi_{\theta_{\text{old}}}(r_{i,t,k} \mid h_{i,t,k}^{\text{on}})$. The verified answer is not used in the policy ratio or in the policy input during rollout generation. It is only used to estimate temporal

credit during training. We first compute the final-answer reward and its group-normalized advantage:

$$R_i = \mathbf{1}[\hat{a}_i = a], A_i = \frac{R_i - \mu(\{R_j\}_{j=1}^G)}{\sigma(\{R_j\}_{j=1}^G) + \epsilon_{\text{norm}}}. \tag{4}$$

Here, $\epsilon_{\text{norm}} > 0$ is a numerical stabilizer used in reward and support-score normalization, distinct from the PPO clip range ϵ. The advantage A_i determines whether rollout i should be reinforced or suppressed. TSDPO does not replace this trajectory-level signal; instead, it redistributes it across the streaming segments of the same rollout. The complete TSDPO objective is:

$$J_{\text{TSDPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{T_i}\sum_{t=1}^{T_i}\frac{1}{|R_{i,t}|}\sum_{k=1}^{|R_{i,t}|}\mathcal{C}\Big(\rho_{i,t,k}(\theta), A_i W_{i,t}\Big) - \beta\,\mathrm{KL}_i(\pi_\theta\|\pi_{\text{ref}})\right)\right], \tag{5}$$

where $\mathcal{C} = \min\big(\rho A,\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\big)$, π_{ref} is the frozen reference policy, and $W_{i,t}$ is the temporal credit weight derived from the segment's answer-support score.

This objective makes the credit assignment structure explicit. The factor A_i provides the trajectory-level direction inherited from GRPO. The temporal weight $W_{i,t}$ reallocates this advantage across streaming segments. Since $W_{i,t}$ is shared by all tokens in $R_{i,t}$, tokens inside the same segment receive a uniform segment-level advantage, while different segments receive different update strengths.

Self-distilled Temporal Weighting The remaining question is how to obtain the temporal allocation weight $W_{i,t}$. TSDPO uses the verified answer as privileged outcome evidence to retrospectively evaluate whether an already sampled streaming segment is consistent with the desired final answer. Importantly, the answer-conditioned model does not generate new trajectories and is not used as a direct KL teacher. It only re-scores existing rollout tokens to provide a segment-level support signal for temporal credit allocation.

For each sampled segment, we compare two likelihood views. The online view scores the segment under the original streaming condition, while the answer-conditioned view scores the same segment with access to the verified answer. Let $h^{\text{ans}}_{i,t,k} = (q, C_{\le t}, a, R_{i,<t}, r_{i,t,<k})$ denote the answer-conditioned history. We define two segment-level scoring views, namely the online log-likelihood and the answer-conditioned log-likelihood:

$$\ell^{\text{on}}_{i,t} = \frac{1}{|R_{i,t}|}\sum_{k=1}^{|R_{i,t}|}\log \pi_{\theta_{\text{old}}}\left(r_{i,t,k} \mid h^{\text{on}}_{i,t,k}\right), \quad \ell^{\text{ans}}_{i,t} = \frac{1}{|R_{i,t}|}\sum_{k=1}^{|R_{i,t}|}\log \pi^{\text{sg}}_{\theta_{\text{old}}}\left(r_{i,t,k} \mid h^{\text{ans}}_{i,t,k}\right). \tag{6}$$

Here, $S_{i,t} = \ell^{\text{ans}}_{i,t} - \ell^{\text{on}}_{i,t}$ denotes answer-support score. A larger $S_{i,t}$ means that the segment becomes more plausible when the verified answer is provided, indicating stronger consistency with the desired outcome. This likelihood gap is a support proxy, not a validated causal contribution estimate; Appendix C illustrates its interpretation and limitations.

Since segments at different streaming positions observe different amounts of context, their likelihood gaps may have different scales. We therefore calibrate support scores among rollouts that contain the same segment position:

$$\bar{S}_{i,t} = \text{clip}\left(\frac{S_{i,t} - \mu(\{S_{j,t} : j \in \mathcal{G}_t\})}{\sigma(\{S_{j,t} : j \in \mathcal{G}_t\}) + \epsilon_{\text{norm}}}, -c, c\right), \tag{7}$$

where $\mathcal{G}_t = \{j \mid t \le T_j\}$ denotes the set of rollouts that contain a segment at position t, and c is the support clipping bound.

The calibrated scores are then converted into temporal allocation weights:

$$\begin{aligned} p_{i,t} &= \text{softmax}_t\left(\frac{\text{sign}(A_i)\bar{S}_{i,t}}{\tau}\right), \\ W_{i,t} &= (1-\lambda) + \lambda T_i p_{i,t}, \quad \lambda \in [0,1]. \end{aligned} \tag{8}$$

The sign-aware softmax aligns temporal allocation with the trajectory-level learning direction. For successful trajectories, high-support segments receive larger positive credit. For failed trajectories, low-support segments receive stronger negative credit. The temperature τ controls how concentrated the allocation is, while λ controls the deviation from uniform GRPO-style assignment. When $\lambda = 0$, we have $W_{i,t} = 1$ for all segments, and TSDPO reduces to uniform trajectory-level advantage assignment.

Algorithm 1 TSDPO: Temporal Self-Distilled Policy Optimization for Streaming Reasoning

Require: Policy π_θ, frozen reference policy π_{ref}, dataset $\mathcal{D} = \{(q, C, a)\}$, group size G, support mixing coefficient $\lambda \in [0, 1]$, temperature $\tau > 0$, support clip bound c, normalization stabilizer $\epsilon_{\text{norm}} > 0$, PPO clip range ϵ, KL coefficient β

Ensure: Updated policy π_θ

for each training iteration **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$ and freeze a snapshot $\theta_{\text{old}} \leftarrow \theta$
  for each $(q, C, a) \in \mathcal{B}$ **do**
    Generate G streaming rollouts $\{o_i\}_{i=1}^{G}$ with $\pi_{\theta_{\text{old}}}$ under the original streaming visibility
    Parse each rollout into intermediate streaming units and a final deep unit, $o_i = [R_{i,1}, \ldots, R_{i,T_i}, \hat{a}_i]$; cache token boundaries and old log-probabilities
    Compute final-answer rewards $R_i = \mathbf{1}[\hat{a}_i = a]$
    Normalize rewards within the current prompt's group to obtain A_i using Eq. 4
    for each rollout i and reasoning unit $t = 1, \ldots, T_i$ (full context for the final deep unit) **do**
      Compute mean token log-likelihoods $\ell_{i,t}^{\text{on}}$ and $\ell_{i,t}^{\text{ans}}$ of the same sampled segment under the frozen old policy, using histories $h_{i,t,k}^{\text{on}}$ and $h_{i,t,k}^{\text{ans}}$, respectively (Eq. 6)
      Set the support score $S_{i,t} = \ell_{i,t}^{\text{ans}} - \ell_{i,t}^{\text{on}}$
    end for
    For each position t, standardize support scores within $\mathcal{G}_t = \{j \in \{1, \ldots, G\} : t \le T_j\}$ and clip to $[-c, c]$ to obtain $\bar{S}_{i,t}$ using Eq. 7
    for each rollout i **do**
      Compute $p_{i,t} = \text{softmax}_t\big(\text{sign}(A_i)\bar{S}_{i,t}/\tau\big)$ over $t = 1, \ldots, T_i$
      Set $W_{i,t} = (1 - \lambda) + \lambda T_i p_{i,t}$ and assign advantage $A_i W_{i,t}$ to all tokens in $R_{i,t}$
    end for
  end for
  Holding A_i, $W_{i,t}$, $\pi_{\theta_{\text{old}}}$, and π_{ref} fixed, update θ by maximizing Eq. 5 under the original online histories, with PPO clipping and KL regularization
end for

Advantage-preserving Credit Redistribution The temporal weights are designed to redistribute, rather than rescale, the original GRPO advantage. Since $\sum_{t=1}^{T_i} p_{i,t} = 1$, we have:

$$\sum_{t=1}^{T_i} W_{i,t} = T_i, \frac{1}{T_i} \sum_{t=1}^{T_i} A_i W_{i,t} = A_i. \tag{9}$$

Therefore, TSDPO preserves the mean segment-level advantage while reallocating credit across temporally ordered reasoning units. This scalar budget constraint does not imply preservation of the gradient norm, since the score-function gradients differ across segments.

Training and Inference Algorithm 1 summarizes TSDPO training. For each prompt, the frozen old policy samples G rollouts under the original streaming visibility. Each rollout is parsed into intermediate reasoning segments, a final deep-reasoning unit, and an answer, from which we compute the group-normalized accuracy advantage A_i.

Next, the frozen old policy re-scores each sampled segment under two views: the online view and the answer-conditioned view. Their likelihood gap gives the answer-support score, which is calibrated by streaming position and converted into temporal allocation weights $W_{i,t}$. Finally, the policy is updated using the segment-aware objective in Eq. 5.

The verified answer is used only during training to estimate temporal credit. Policy optimization and inference retain the original streaming visibility: the model conditions each segment on the currently observed context and previously generated segments. Figure 2 in Appendix E illustrates the training and inference workflow.

4 Experiments

4.1 Experimental Setup

Models and datasets. We evaluate TSDPO on Qwen3-1.7B and Qwen3-4B (Yang et al., 2025). Training uses a mixture of mathematical reasoning and context-based QA datasets: GSM-Symbolic

Table 1: Main results. Acc is final-answer accuracy; S.Len, D.Len, and T.Len are streaming, deep, and total token counts. Shading highlights TSDPO; bold marks the best value per dataset and model. Checkpoint results are reported here; three-seed statistics appear in Table 9.

	Qwen3-1.7B				Qwen3-4B			
Method	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**
GSM-symbolic P1								
Read-then-think	0.634	–	1598.7	1598.7	0.636	–	1717.2	1717.2
StreamingThinker (SFT)	0.834	165.7	**118.0**	**283.7**	0.934	167.8	119.2	287.0
GRPO	0.850	172.994	133.798	306.792	0.946	157.305	**111.145**	268.450
TSDPO (Ours)	**0.871**	**160.032**	131.972	292.004	**0.950**	**150.688**	116.559	**267.247**
GSM-symbolic P2								
Read-then-think	0.424	–	1866.5	1866.5	0.462	–	1937.4	1937.4
StreamingThinker (SFT)	0.642	229.3	**132.6**	**361.9**	0.816	227.2	130.4	357.6
GRPO	0.758	235.784	148.704	384.488	0.874	209.866	**126.268**	336.134
TSDPO (Ours)	**0.788**	**210.046**	160.210	370.256	**0.884**	**196.390**	131.162	**327.552**
MetaMathQA								
Read-then-think	**0.848**	–	1281.8	1281.8	0.828	–	1379.1	1379.1
StreamingThinker (SFT)	0.688	**149.2**	**121.5**	**270.7**	0.860	152.2	122.0	274.2
GRPO	0.823	158.712	159.285	317.997	0.909	140.010	**113.545**	253.555
TSDPO (Ours)	0.826	173.246	129.356	302.602	**0.924**	**121.708**	115.650	**237.358**
PubMedQA								
Read-then-think	0.588	–	434.6	**434.6**	0.615	–	331.8	**331.8**
StreamingThinker (SFT)	0.749	414.7	**120.5**	535.3	0.756	402.9	116.1	519.0
GRPO	0.776	433.573	143.314	576.887	0.792	378.214	**111.842**	490.056
TSDPO (Ours)	**0.793**	**414.206**	136.642	550.848	**0.807**	**356.728**	114.512	471.240

P1/P2 (Mirzadeh et al., 2025), MetaMathQA (Yu et al., 2024), and PubMedQA (Jin et al., 2019). We additionally evaluate Qwen3-1.7B on GSM-Infinite to examine whether efficient streaming reasoning is retained under distribution shift. This single-dataset evaluation is a targeted check of streaming behavior, rather than a comprehensive assessment of out-of-distribution (OOD) generalization.

Baselines. Our main comparisons include *Read-then-think*, which receives the full context before reasoning; *StreamingThinker (SFT)* (Tong et al., 2025), trained on constructed streaming CoT traces; and *GRPO* (Shao et al., 2024), which applies uniform trajectory-level advantages to streaming rollouts. Appendix B.1 additionally compares TSDPO with a privileged self-distillation baseline under matched answer privilege during training.

Metrics and reporting. We report final-answer accuracy (Acc), streaming reasoning length (S.Len), final deep-reasoning length (D.Len), and total generated length (T.Len). Lengths are measured in tokens; lower length indicates a smaller generation budget and is interpreted jointly with accuracy. Accuracy is reported as a fraction unless explicitly marked as a percentage. The full-TSDPO entries in the ablations use the same checkpoint as the corresponding main results. Appendix F separately reports Qwen3-4B means and standard deviations over three random seeds; these aggregate statistics are not the individual checkpoint values in the main tables.

Implementation details. We follow the streaming setup of StreamingThinker (Tong et al., 2025): the input is decomposed into sentences, and intermediate reasoning segments are interleaved with incoming context, followed by a final deep-reasoning phase. We use the same TSDPO training protocol across datasets. Training details and hyperparameters, including the KL coefficient $\beta = 0.01$, are provided in Appendix E.

4.2 Main Results

Table 1 reports results across four benchmarks and two model sizes. TSDPO improves accuracy over GRPO in all eight model–benchmark settings. On Qwen3-1.7B, the gains are 2.1 and 3.0 percentage

Table 2: Credit granularity (Qwen3-1.7B). TSDPO-streaming uses two units: all streaming segments and final deep reasoning; full TSDPO uses individual segments. Reported numeric runs are stable. TSDPO-token collapses on GSM P1/P2 and is unreported elsewhere. Shading marks full TSDPO; bold marks each dataset's best values.

	GSM-symbolic P1				GSM-symbolic P2			
Method	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**
GRPO	0.850	172.994	133.798	306.792	0.758	235.784	**148.704**	384.488
TSDPO-token	*Collapsed*				*Collapsed*			
TSDPO-streaming	0.862	168.540	138.964	307.504	0.774	229.612	161.348	390.960
TSDPO (full)	**0.871**	**160.032**	**131.972**	**292.004**	**0.788**	**210.046**	160.210	**370.256**

	MetaMathQA				PubMedQA			
Method	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**
GRPO	0.823	158.712	159.285	317.997	0.776	433.573	143.314	576.887
TSDPO-streaming	0.824	**153.684**	164.911	318.595	0.785	422.681	146.874	569.555
TSDPO (full)	**0.826**	173.246	**129.356**	**302.602**	**0.793**	**414.206**	**136.642**	**550.848**

points on GSM-symbolic P1 and P2, respectively, and 0.3 and 1.7 points on MetaMathQA and PubMedQA. On Qwen3-4B, the corresponding gains are 0.4, 1.0, 1.5, and 1.5 points. These results support the usefulness of segment-level credit allocation at both evaluated model sizes.

TSDPO also reduces total length relative to GRPO in all eight settings. For example, total length decreases from 306.792 to 292.004 on Qwen3-1.7B GSM-symbolic P1 and from 253.555 to 237.358 on Qwen3-4B MetaMathQA. Streaming length decreases in seven settings. The exception is Qwen3-1.7B MetaMathQA, where streaming reasoning becomes longer but deep reasoning becomes shorter, yielding a net reduction in total length. Thus, the observed benefit is a better overall accuracy–length trade-off, rather than a uniform reduction in every reasoning phase.

TSDPO achieves higher accuracy than StreamingThinker (SFT) in every setting. SFT uses fewer total tokens on all four Qwen3-1.7B benchmarks, whereas TSDPO is both more accurate and shorter on Qwen3-4B. Compared with read-then-think, TSDPO achieves higher accuracy in seven of eight settings; the exception is Qwen3-1.7B MetaMathQA (0.826 versus 0.848). These comparisons distinguish gains over uniform GRPO from the broader trade-offs between supervised streaming reasoning and reasoning after full input arrival.

Comparison with privileged self-distillation. Under matched answer privilege, TSDPO improves accuracy and reduces total generation relative to privileged self-distillation on all four Qwen3-1.7B benchmarks. This supports the combined TSDPO design, although the comparison does not isolate the contributions of segment granularity and advantage preservation. Full results and discussion appear in Table 6, Appendix B.1.

4.3 Influence of Credit Granularity

Stage-level versus segment-level allocation. Table 2 compares four credit-assignment granularities on Qwen3-1.7B. GRPO assigns the same advantage throughout a trajectory, while TSDPO-token assigns credit at the token level. *TSDPO-streaming* is a coarser, stage-level variant: it treats the entire streaming-reasoning phase, containing multiple intermediate segments, as one unit, and the final deep-reasoning phase as a second unit. All intermediate streaming segments therefore share the same temporal credit. In contrast, *full TSDPO* computes a separate answer-support score and credit weight for each individual reasoning segment, including the final deep-reasoning unit. This comparison tests whether distinguishing the two phases is sufficient, or whether allocating credit among the intermediate streaming segments provides an additional benefit.

Accuracy and length. Stage-level allocation improves accuracy over GRPO on all four benchmarks, but does not consistently shorten the whole trajectory: total length increases slightly on both GSM-symbolic splits and MetaMathQA. Full TSDPO further reduces total length relative to TSDPO-streaming by 5.0%, 5.3%, 5.0%, and 3.3% on P1, P2, MetaMathQA, and PubMedQA, respectively. It

Table 3: Position-wise support normalization (Eq. 7). Shading marks the normalized variant; bold marks the better value within each model–dataset pair.

	GSM-symbolic P1				GSM-symbolic P2			
Normalization	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**	**Acc↑**	S.Len↓	D.Len↓	**T.Len↓**
Qwen3-1.7B								
w/o	**0.875**	182.384	143.219	325.603	0.762	244.912	164.706	409.618
w/	0.871	**160.032**	**131.972**	**292.004**	**0.788**	**210.046**	**160.210**	**370.256**
Qwen3-4B								
w/o	0.941	156.863	121.830	278.693	0.875	202.536	136.743	339.279
w/	**0.950**	**150.688**	**116.559**	**267.247**	**0.884**	**196.390**	**131.162**	**327.552**

Table 4: Streaming reasoning under distribution shift on GSM-Infinite with Qwen3-1.7B. This targeted evaluation compares accuracy and generation length; it is not a comprehensive OOD generalization assessment.

Method	Acc↑	S.Len↓	D.Len↓	T.Len↓
StreamingThinker (SFT)	0.464	**164.152**	**124.656**	**288.809**
GRPO	**0.5060**	204.015	247.92	451.935
TSDPO	0.4917	173.57	179.10	352.68

reaches accuracies of 0.871, 0.788, 0.826, and 0.793, with gains over stage-level allocation of 1.4, 0.2, and 0.8 percentage points on P2, MetaMathQA, and PubMedQA. The MetaMathQA result illustrates that this benefit need not come from shorter intermediate reasoning: full TSDPO increases S.Len from 153.684 to 173.246 while reducing D.Len from 164.911 to 129.356. These results support allocating credit within the streaming phase, beyond merely separating streaming and deep reasoning.

Training stability. GRPO, TSDPO-streaming, and full TSDPO remain stable in the reported runs. Under the same hyperparameters, TSDPO-token collapses on both GSM-symbolic splits: streaming responses drift from the required output format, token length increases sharply, and verifier failures accompany a rapid reward decline. We therefore report these runs as collapsed rather than presenting final-checkpoint metrics. This observation is limited to the tested configuration and does not establish that token-level credit allocation is unstable in general. Token-level results are not available for MetaMathQA or PubMedQA.

4.4 Influence of Normalization

Table 3 evaluates position-wise support normalization (Eq. 7) on both Qwen3-1.7B and Qwen3-4B. Different streaming positions have different amounts of observed context, so raw answer-support scores may have different scales. Position-wise normalization compares rollouts at the same position before allocating credit across a trajectory, reducing the influence of these scale differences.

Normalization reduces streaming, deep-reasoning, and total lengths in all four model–split settings. On Qwen3-1.7B, total length falls from 409.618 to 370.256 on P2 and from 325.603 to 292.004 on P1, reductions of 9.6% and 10.3%, respectively. Accuracy improves from 0.762 to 0.788 on P2, but decreases slightly from 0.875 to 0.871 on P1. On Qwen3-4B, normalization improves accuracy from 0.875 to 0.884 on P2 and from 0.941 to 0.950 on P1, while reducing total length from 339.279 to 327.552 and from 278.693 to 267.247. The evidence therefore supports a consistent reduction in generation length and accuracy gains in three of four settings, rather than an unconditional accuracy improvement.

4.5 Streaming Reasoning under Distribution Shift

We evaluate Qwen3-1.7B on GSM-Infinite to check whether TSDPO retains effective streaming reasoning on a distribution-shifted dataset. Table 4 shows accuracy of 0.4917 for TSDPO, compared with 0.464 for StreamingThinker (SFT) and 0.5060 for GRPO. Relative to GRPO, TSDPO has a 1.43-percentage-point accuracy gap while reducing total length from 451.935 to 352.68 (22.0%).

Table 5: Mean serving performance across all four datasets with Qwen3-1.7B under vLLM. Throughput and estimated post-input completion latency are averaged equally across datasets. Speedup is the ratio of mean Read-then-think latency to mean method latency, computed before rounding. Per-dataset results are reported in Appendix G (Table 10); bold marks the best value.

Method	Throughput↑ (tokens/s)	Post-input latency↓ (s)	Speedup↑
Read-then-think	**141.2**	9.17	1.00×
StreamingThinker (SFT)	134.8	**0.92**	**10.02**×
GRPO	136.6	1.07	8.57×
TSDPO	137.1	1.02	9.01×

Both streaming and deep-reasoning lengths decrease, from 204.015 to 173.57 and from 247.92 to 179.10, respectively.

This result indicates that TSDPO can retain compact streaming reasoning while maintaining accuracy close to GRPO on this dataset. The length measurements characterize the generation budget; they do not directly establish wall-clock latency under distribution shift. Serving latency is evaluated separately below.

4.6 Real Latency Analysis

Table 5 summarizes Qwen3-1.7B serving performance under vLLM on an H100 GPU, using equal-weight means across GSM-symbolic P1/P2, MetaMathQA, and PubMedQA. Following StreamingThinker (Tong et al., 2025), we simulate input arrival at 150 words per minute. Latency estimates the remaining decoding time after input completion: total generated length divided by throughput for read-then-think, and final deep-reasoning length divided by throughput for streaming methods. Read-then-think has the highest mean throughput, 141.2 tokens per second, but a mean post-input latency of 9.17 seconds.

TSDPO achieves a mean throughput of 137.1 tokens per second and a mean post-input latency of 1.02 seconds. The ratio of the unrounded mean latencies gives a $9.01\times$ speedup over read-then-think, equivalent to an 88.9% latency reduction. StreamingThinker (SFT) and GRPO achieve $10.02\times$ and $8.57\times$ speedups, respectively. TSDPO reduces the average remaining wait relative to GRPO while improving accuracy on all four datasets; SFT retains the lowest mean latency.

Appendix G, Table 10, reports the corresponding per-dataset results over three evaluation runs, together with timing definitions and overlap assumptions. TSDPO's estimated post-input latency is 0.94–1.17 seconds and is lower than GRPO on three of four datasets; GSM-symbolic P2 is an exception because its final deep-reasoning phase is longer. These estimates exclude input reception time and do not measure full end-to-end serving latency or establish robustness across input rates.

5 Conclusion

We presented TSDPO for temporal credit assignment in real-time streaming reasoning. TSDPO compares online and answer-conditioned likelihoods under the frozen old policy to estimate support for segments generated under partial observations. It redistributes GRPO credit while preserving the mean segment-level advantage, without process annotations or an external teacher.

Across four benchmarks with Qwen3-1.7B and Qwen3-4B, TSDPO improves accuracy and reduces total generation relative to GRPO in all eight settings. Ablations support segment-level allocation beyond a two-stage streaming/deep split, while token-level allocation collapses in the tested GSM-symbolic runs. Position-wise normalization reduces generation length in all four evaluated settings and improves accuracy in three. On GSM-Infinite, TSDPO retains compact streaming reasoning and accuracy close to GRPO, supporting streaming behavior under the tested shift without establishing broader OOD generalization.

AI USE STATEMENT

We used generative AI tools to assist with method implementation and to polish the manuscript. The authors manually reviewed the AI-assisted code to verify implementation correctness and checked the manuscript text for accuracy. The authors take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we specify the TSDPO objective and temporal credit allocation in Section 3, provide training pseudocode in Algorithm 1, and derive the policy gradient under detached credit weights in Appendix D. Section 4.1 describes the model backbones, datasets, baselines, streaming protocol, and evaluation metrics. Appendix E and Table 8 report the training framework, hardware, optimization settings, and rollout-sampling hyperparameters. Appendix F reports means and standard deviations over three random seeds for Qwen3-4B, separately from the individual-checkpoint results. For latency evaluation, Appendix G specifies the serving setup, simulated input rate, throughput-based latency estimates, overlap assumptions, and results over three evaluation runs.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18407–18418, 2024.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. SHANKS: Simultaneous hearing and thinking for spoken language models. *arXiv preprint arXiv:2510.06917*, 2025.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2174–2184, 2018.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

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 *Proceedings of the 12th International Conference on Learning Representations*, 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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2567–2577, 2019.

Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Anthony Liang, Jonathan Berant, Adam Fisch, Abhimanyu Goyal, Kalpesh Krishna, and Jacob Eisenstein. Plantain: Plan-answer interleaved reasoning. *arXiv preprint arXiv:2512.03176*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

OpenAI. Learning to reason with llms. *OpenAI Blog*, 2024. URL `https://openai.com/index/learning-to-reason-with-llms/`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Li, Mingchuan Zhang, Y.K. Zhang, Yu Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi-Jen Shih, Desh Raj, Chunyang Wu, Wei Zhou, SK Bong, Yashesh Gaur, Jay Mahadeokar, Ozlem Kalinli, and Mike Seltzer. Can speech LLMs think while listening? *arXiv preprint arXiv:2510.07497*, 2025.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Junlong Tong, Yingqi Fan, Anhao Zhao, Yunpu Ma, and Xiaoyu Shen. Streamingthinker: Large language models can think while reading. *arXiv preprint arXiv:2510.17238*, 2025.

Jonathan Uesato, Nate Kushman, Ramesh Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Ceze, Tobias Glasmachers, and Laurent Sifre. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5–6):544–557, 2009.

Peiyi Wang, Lei Li, Zhihong Shao, Run Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

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. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025.

Zhifei Xie and Changqiao Wu. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Wei Chee Yew, Hailun Xu, Sanjay Saha, Xiaotian Fan, Hiok Hian Ong, David Yuchen Wang, Kanchan Sarkar, Zhenheng Yang, and Danhui Guan. Dynamic content moderation in livestreams: Combining supervised classification with mllm-boosted similarity matching. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2528–2538, 2026.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Jialiang Zhang, Junlong Tong, Junyan Lin, Hao Wu, Yirong Sun, Yunpu Ma, and Xiaoyu Shen. Think-as-you-see: Streaming chain-of-thought reasoning for large vision-language models. *arXiv preprint arXiv:2603.02872*, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026.

A Related Work

Real-time Streaming Reasoning. Recent reasoning models improve accuracy by increasing test-time computation after the full input is observed (Wei et al., 2022; Zelikman et al., 2022; OpenAI, 2024; DeepSeek-AI, 2025). One line of low-latency work interleaves reasoning with *output generation* under static inputs, such as Pause Tokens (Goyal et al., 2024), Quiet-STaR (Zelikman et al., 2024), Coconut (Hao et al., 2024), Interleaved Reasoning via RL (Xie et al., 2025), and Plantain (Liang et al., 2025). Another line studies true streaming inputs, including online video, speech, and navigation (Chen et al., 2024; Xie & Wu, 2024; Défossez et al., 2024; Wei et al., 2025). StreamingThinker (Tong et al., 2025) first formulates text-based *think-while-reading*; TaYS (Zhang et al., 2026), SHANKS (Chiang et al., 2025), and speech-LLM work (Shih et al., 2025) extend streaming reasoning to multimodal settings. These works study paradigms, inference, or supervised traces; we optimize streaming reasoning policies with verifiable outcomes and temporal credit.

RLVR for Reasoning. Language-model post-training has relied on human preference feedback and policy or preference objectives (Ouyang et al., 2022; Schulman et al., 2017; Rafailov et al., 2023). For checkable reasoning tasks, RLVR replaces learned preference rewards with automatic outcome verification. GRPO (Shao et al., 2024) estimates group-relative advantages without a learned value function and has been scaled in recent reasoning systems (DeepSeek-AI, 2025; Kimi Team, 2025). Its limitation is that outcome rewards are sparse and sequence-level: they verify the final answer but do not identify which intermediate steps caused success or failure. Process reward models (Uesato et al., 2022; Lightman et al., 2023) and Math-Shepherd (Wang et al., 2024) provide denser step-level signals, but often require annotations, verifier rollouts, or process-level supervision. TSDPO keeps outcome-only scalability while allocating credit across streaming segments.

Distillation and Privileged Information. Knowledge distillation (Hinton et al., 2015) transfers knowledge from a teacher to a student, while privileged-information learning (Vapnik & Vashist, 2009) studies training-only signals unavailable at test time. OPD reduces exposure bias by training on student-sampled trajectories with teacher feedback, from Generalized Knowledge Distillation (Agarwal et al., 2024) to black-box, context, and entropy-aware variants (Ye et al., 2025; 2026; Jin et al., 2026); recent analyses summarize its design space and failure modes (Song & Zheng, 2026; Li et al., 2026; Zhu et al., 2026). Related self-distillation and privileged-information methods use the same model with additional training-time context. Self-Distilled Reasoner and SDPO provide privileged distribution matching on student trajectories (Zhao et al., 2026; Hübotter et al., 2026); π-Distill jointly trains a shared-parameter teacher and student using teacher-generated trajectories (Penaloza et al., 2026); self-distilled RLVR reweights outcome-grounded token advantages (Yang et al., 2026). TSDPO builds on privileged credit weighting, aligning it with streaming *segments* and preserving the mean segment-level advantage. Appendix B details these relationships.

B Relationship to RLVR and Privileged-Information Methods

The additional structure introduced by streaming. Outcome-only rewards leave credit-assignment ambiguity in both standard chain-of-thought and streaming reasoning. In read-then-think reasoning, the full input is available before reasoning begins. In streaming reasoning, each intermediate segment is generated under a different partial observation, so its temporal position also determines the available evidence and uncertainty. A correct final answer can follow an early omission that is repaired later; an incorrect answer can follow useful local reasoning that fails during final integration. TSDPO uses this explicit temporal structure to allocate the verified outcome signal across reasoning units, without assuming that the credit-assignment problem is unique to streaming.

Verified answers as retrospective evidence. TSDPO draws on privileged-information learning (Vapnik & Vashist, 2009) and self-distilled policy optimization (Hübotter et al., 2026; Yang et al., 2026; Zhao et al., 2026). The frozen old policy scores each already sampled segment twice: under its original online history and under the same history augmented with the verified answer. Their likelihood gap supplies an answer-support proxy. This is a posterior-style use of evidence for retrospective assessment, not an explicitly estimated Bayesian posterior or a causal effect. The answer-conditioned policy does not generate a replacement trajectory. Privileged information is used

Table 6: Comparison with GRPO and privileged self-distillation on Qwen3-1.7B. TSDPO and privileged self-distillation use matched answer privilege during training. Acc is final-answer accuracy; T.Len is total generated length in tokens. Bold marks the best value per benchmark.

	GRPO		Privileged self-distillation		TSDPO	
Benchmark	Acc↑	T.Len↓	Acc↑	T.Len↓	Acc↑	T.Len↓
GSM-symbolic P1	0.850	306.792	0.839	320.042	**0.871**	**292.004**
GSM-symbolic P2	0.758	384.488	0.746	373.280	**0.788**	**370.256**
MetaMathQA	0.823	317.997	0.811	333.641	**0.826**	**302.602**
PubMedQA	0.776	576.887	0.764	565.602	**0.793**	**550.848**

only in training-time scoring; rollout generation, the policy likelihood ratio, and deployment retain the original streaming visibility.

Distribution matching versus credit weighting. Privileged self-distillation covers several training objectives. On-policy self-distillation can match a same-model teacher's next-token distributions, conditioned on an answer or reference reasoning, on student-generated trajectories (Zhao et al., 2026). SDPO obtains such supervision from richer feedback, including environment feedback and successful rollouts (Hübotter et al., 2026). Other privileged-information methods jointly train a shared-parameter teacher and student using teacher-generated trajectories (Penaloza et al., 2026). These approaches differ in supervision and sampling; they are not characterized by a general restriction on generating chain-of-thought.

Self-distilled RLVR already uses privileged likelihood ratios to reweight outcome-grounded token advantages without an auxiliary distillation loss (Yang et al., 2026). TSDPO shares this separation between the outcome-based update direction and privileged credit allocation. Its distinction is the combination of streaming-aligned segment units, support assessment under the corresponding partial observations, and a normalized credit budget. Thus, using privilege for credit weighting is an inherited principle, while the temporal allocation structure is the focus of our method.

Segment units and credit-magnitude preservation. Full TSDPO gives each intermediate streaming segment and the final deep-reasoning unit a separate support score and weight. Position-wise calibration compares rollouts at matched streaming positions before allocating credit within each trajectory. The weights obey

$$\frac{1}{T_i}\sum_{t=1}^{T_i} W_{i,t} = 1, \qquad \frac{1}{T_i}\sum_{t=1}^{T_i} A_i W_{i,t} = A_i. \tag{10}$$

Here, credit-magnitude preservation means preservation of the mean segment-level advantage, including its sign; it does not imply an unchanged gradient norm or parameter update. The usual KL regularizer to π_{ref} remains separate from answer-conditioned scoring. With $\lambda = 0$, TSDPO recovers uniform credit within its segment-averaged surrogate, as derived in Appendix D.

B.1 Comparison with Privileged Self-Distillation

Matched-privilege results. Table 6 reports Qwen3-1.7B accuracy and total generated length for GRPO, privileged self-distillation, and TSDPO. The GRPO and TSDPO entries are the same checkpoint results as in Table 1. Privileged self-distillation and TSDPO use matched answer privilege during training: the former provides token-level distribution-matching supervision, while the latter uses answer support to allocate segment-level credit. TSDPO improves accuracy over privileged self-distillation from 0.839 to 0.871 on GSM-symbolic P1, from 0.746 to 0.788 on P2, from 0.811 to 0.826 on MetaMathQA, and from 0.764 to 0.793 on PubMedQA. It also reduces total length on every benchmark.

What the comparisons establish. These results support the combined TSDPO design, but do not independently identify the effect of every component: the distillation comparison changes the optimization target, credit granularity, and preservation of the mean segment-level advantage. The

granularity study in Table 2 separately compares a two-stage streaming/deep split with individual reasoning units, and Table 3 examines position-wise calibration. Neither is a standalone ablation of mean-one budget preservation. Self-distilled RLVR is discussed above as a related approach to privileged credit weighting; its numerical results are unavailable, so it is not included in the empirical comparison. The observed token-level collapse in Table 2 pertains specifically to the tested TSDPO-token variant.

C Qualitative Analysis of Temporal Credit

Information availability is part of credit assignment. Outcome-only rewards create credit-assignment ambiguity in both standard and streaming reasoning. Streaming adds a temporal distinction: each segment is generated under a different partial observation. An intermediate conclusion can be valid when produced but require revision after new evidence arrives. Conversely, an incorrect final answer can follow useful local computations. Credit allocation should therefore consider both the content of a segment and the information available at its generation boundary.

Correct answer after an intermediate omission. Table 7 illustrates an inventory problem whose answer is $80 - 26 + 9 - 7 = 56$. After the purchases are revealed, $80 - 26 = 54$ is a valid provisional calculation; it should not be treated as an error for failing to anticipate the later cancellations. Once the cancellations are visible, however, skipping the update leaves a stale inventory. The subsequent subtraction produces 47, and the final deep-reasoning segment repairs the omission to recover 56. A <skip> action is not inherently wrong; the failure illustrated here is leaving the newly available cancellation information out of the subsequent inventory calculation. In a mixed-reward rollout group, this successful trajectory has $A_i > 0$, so uniform GRPO credit reinforces every segment, including the omission and stale conclusion. The intended allocation gives these segments less positive credit than the useful calculations and final correction.

Incorrect answer despite useful intermediate reasoning. For a paired illustrative failure, keep the same streaming segments but replace the deep-reasoning continuation with "The remaining inventory is $54 - 7 = 47$," followed by answer 47. This continuation fails to incorporate the cancellations, although the earlier extraction of 80 and provisional subtraction of 26 remain locally valid. With $A_i < 0$, GRPO assigns the same negative advantage throughout. TSDPO is intended to concentrate the negative signal on low-support segments while attenuating it on higher-support segments. These two continuations illustrate why final correctness alone cannot distinguish useful local reasoning from an omitted update or a successful repair.

How to interpret the weights. The arrows describe a desired relative allocation, not measured support scores or numerical weights. Actual allocation depends on the answer-conditioned likelihood gap in Eq. 6, calibration across rollouts at the same segment position, and the sign-aware softmax in Eq. 8. For $A_i > 0$, a larger calibrated support score yields a larger weight; for $A_i < 0$, the ordering reverses, giving low-support segments stronger negative credit. The nonnegative weights preserve the sign of the trajectory advantage and satisfy $\sum_t W_{i,t} = T_i$ (Eq. 9). Thus, downweighting a flawed segment in a successful trajectory reduces its reinforcement; it does not penalize that segment. Likewise, useful segments in a failed trajectory do not receive positive credit. If $A_i = 0$, temporal weighting creates no reward-surrogate update.

What the example does and does not establish. Both continuations are illustrative constructions, not recorded model rollouts. Local validity does not guarantee a large likelihood gap: a generic restatement, a provisional calculation, or a <skip> action may receive scores that differ from the intended ordering. The score measures answer-conditioned plausibility, which is a proxy for answer support rather than a causal contribution estimate. Validating causality would require interventions, such as deleting or replacing a segment under the same visible prefix and regenerating the continuation. We do not report such an intervention study here; these examples explain the allocation mechanism and its limitations.

Table 7: Illustrative temporal credit assignment for a successful inventory trajectory. Each input row adds to the previously visible context; deep reasoning sees the complete input. GRPO assigns the same positive advantage A_i to all six reasoning units in a mixed-reward group. Arrows indicate the desired relative weight ordering within this trajectory, not observed TSDPO scores, absolute thresholds, or causal effects. A dash leaves the desired ordering unspecified.

Unit	Newly visible input	Illustrative reasoning segment	Local assessment	Desired weight
R_1	How many items remain in the store at closing time?	"We need to calculate the remaining inventory."	Task restatement; usefulness is not determined by correctness alone.	—
R_2	The store initially has 80 items.	"The initial inventory is 80."	Valid extraction from the visible input.	↑
R_3	Customers purchase 26 items during the day.	"After the purchases, $80 - 26 = 54$ items remain."	Valid provisional calculation before cancellations are revealed.	↑
R_4	Before pickup, 9 purchases are canceled; those items remain in the store.	`<skip>`	Omits a newly available inventory update.	↓
R_5	At closing time, 7 damaged items are discarded.	"Discarding 7 items leaves $54 - 7 = 47$."	Uses a stale inventory after the cancellation update.	↓
R_6	Full input (deep reasoning).	"The 9 canceled purchases were never picked up. Including them and discarding 7 items leaves $54 + 9 - 7 = 56$."	Correctly integrates the missed update and repairs the conclusion.	↑
Final answer		56 (correct; the answer is not an additional reasoning unit).		

D Policy Gradient of TSDPO

We derive the policy gradient of TSDPO to clarify how temporal weighting changes GRPO. During a policy update, the trajectory-level advantage A_i, the temporal weights $W_{i,t}$, and the old-policy denominators in $\rho_{i,t,k}$ are treated as fixed quantities:

$$\nabla_\theta A_i = 0, \qquad \nabla_\theta W_{i,t} = 0, \qquad \nabla_\theta \pi_{\theta_{\text{old}}} = 0. \tag{11}$$

This stop-gradient convention is important: the verified answer affects the scalar credit assignment, but gradients are only taken through the policy being optimized.

Token-level clipped gradient. Let $A^{\text{seg}}_{i,t} = A_i W_{i,t}$ and abbreviate $\rho_{i,t,k}(\theta)$ as $\rho_{i,t,k}$. For the clipped surrogate

$$\mathcal{C}(\rho, A) = \min(\rho A,\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A)\,,$$

the subgradient with respect to θ is active only when the PPO clipping does not saturate the update direction:

$$m_{i,t,k} = \mathbf{1}\left[\begin{array}{l} A^{\text{seg}}_{i,t} \geq 0 \text{ and } \rho_{i,t,k} \leq 1+\epsilon, \\ \text{or} \\ A^{\text{seg}}_{i,t} < 0 \text{ and } \rho_{i,t,k} \geq 1-\epsilon. \end{array}\right] \tag{12}$$

Ignoring boundary points of the piecewise function, and using $\nabla_\theta \rho_{i,t,k} = \rho_{i,t,k} \nabla_\theta \log \pi_\theta(r_{i,t,k} \mid h^{\text{on}}_{i,t,k})$, we obtain

$$\nabla_\theta \mathcal{C}(\rho_{i,t,k}, A_i W_{i,t}) = m_{i,t,k}\, A_i W_{i,t}\, \rho_{i,t,k}\, \nabla_\theta \log \pi_\theta(r_{i,t,k} \mid h^{\text{on}}_{i,t,k}). \tag{13}$$

Trajectory-level gradient. Substituting Eq. 13 into Eq. 5 gives

$$\nabla_\theta J_{\mathrm{TSDPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\Gamma_i^{\mathrm{TSDPO}}(\theta)\right], \tag{14}$$

where the per-rollout gradient contribution is

$$\Gamma_i^{\mathrm{TSDPO}}(\theta) = \frac{A_i}{T_i}\sum_{t=1}^{T_i} W_{i,t}\,\psi_{i,t}(\theta) - \beta\nabla_\theta \mathrm{KL}_i(\pi_\theta\|\pi_{\mathrm{ref}}), \tag{15}$$

and the segment score-function term is

$$\psi_{i,t}(\theta) = \frac{1}{|R_{i,t}|}\sum_{k=1}^{|R_{i,t}|} m_{i,t,k}\rho_{i,t,k}\cdot\nabla_\theta \log \pi_\theta(r_{i,t,k} \mid h_{i,t,k}^{\mathrm{on}}). \tag{16}$$

Thus, TSDPO keeps the same trajectory-level sign A_i as GRPO, but multiplies the segment score-function term by $W_{i,t}$ before averaging over streaming segments.

Relationship to GRPO. If $\lambda = 0$, then $W_{i,t} = 1$ for all t, and Eq. 14 reduces to the segment-averaged GRPO gradient:

$$\nabla_\theta J_{\mathrm{GRPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\Gamma_i^{\mathrm{GRPO}}(\theta)\right]. \tag{17}$$

Here

$$\Gamma_i^{\mathrm{GRPO}}(\theta) = \frac{A_i}{T_i}\sum_{t=1}^{T_i}\psi_{i,t}(\theta) - \beta\nabla_\theta \mathrm{KL}_i(\pi_\theta\|\pi_{\mathrm{ref}}). \tag{18}$$

For general λ, the difference from GRPO is a zero-budget redistribution term:

$$\nabla_\theta J_{\mathrm{TSDPO}} - \nabla_\theta J_{\mathrm{GRPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\Delta_i(\theta)\right], \tag{19}$$

where

$$\Delta_i(\theta) = \frac{A_i}{T_i}\sum_{t=1}^{T_i}(W_{i,t}-1)\psi_{i,t}(\theta), \tag{20}$$

and the redistribution is budget-neutral because $\sum_{t=1}^{T_i}(W_{i,t}-1) = 0$. Equation 19 shows the role of TSDPO precisely: it does not introduce an additional reward term or a distillation loss. It changes the coefficients of existing on-policy score-function gradients. Higher-support segments receive stronger positive credit when $A_i > 0$, whereas lower-support segments receive stronger negative credit when $A_i < 0$, under the same mean segment-level advantage budget.

Advantage preservation. Because $\sum_t W_{i,t} = T_i$, the average segment-level advantage remains equal to the original GRPO advantage:

$$\frac{1}{T_i}\sum_{t=1}^{T_i} A_{i,t}^{\mathrm{seg}} = \frac{1}{T_i}\sum_{t=1}^{T_i} A_i W_{i,t} = A_i. \tag{21}$$

Therefore, TSDPO preserves the mean segment-level advantage and its trajectory-level sign. This identity concerns the scalar advantage budget, not the norm of the resulting gradient, which also depends on the segment score-function terms.

E Training Details

Training workflow. Figure 2 illustrates the training and inference workflow corresponding to Algorithm 1 in Section 3. Each iteration generates on-policy streaming rollouts, computes trajectory-level advantages, evaluates online and answer-conditioned segment support with the frozen old policy, calibrates support scores by position, allocates temporal credit, and updates the policy using the segment-aware objective. The advantages and temporal weights are held fixed during this update, while the policy likelihood ratio uses only the original online histories.

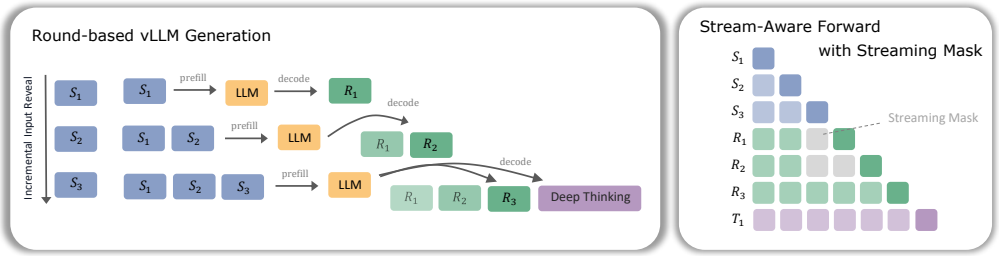

Figure 2: Training and inference workflow of TSDPO. During training, verified answers are used only for self-distilled segment support evaluation, while rollout generation, policy optimization, and inference follow the original streaming visibility.

Table 8: Hyperparameters used for streaming RL training.

Parameter	**Value**
Support mixing coefficient (λ)	0.5
Softmax temperature (τ)	1.0
Support clipping bound (c)	3.0
PPO clip range (ϵ)	0.2
Train batch size	128 prompts
PPO mini batch size	16
PPO micro batch size per GPU	2
Log-prob micro batch size per GPU	8
KL coefficient (β)	0.01
Max prompt length	8192 tokens
Max response length	2048 tokens
Max model length	8192 tokens
Sampling temperature	0.6
Top-p sampling	0.95
Top-k sampling	20
Number of samples per prompt (G)	8
Intermediate max tokens	512 tokens
Compression ratio	0.35
Minimum reasoning tokens	20 tokens
Maximum reasoning tokens	2048 tokens
Optimizer	AdamW
Learning rate	1×10^{-6}
Learning rate schedule	Cosine
Weight decay	0.01
Warmup ratio	0.05
Training epochs	3
Distillation steps	50

Hyperparameters. We train all models using the AdamW optimizer with a learning rate of 1×10^{-6} and a cosine learning rate schedule with a warmup ratio of 0.05. The batch size is set to 128 prompts per iteration, with a group size of $G = 8$ rollouts per prompt. For TSDPO-specific hyperparameters, we use a support mixing coefficient $\lambda = 0.5$, softmax temperature $\tau = 1.0$, support clipping threshold $c = 3.0$, PPO clip range $\epsilon = 0.2$, and KL coefficient $\beta = 0.01$. Training is conducted for 3 epochs on the combined training set. RL training is run on 4×A100 GPUs with the veRL framework; the per-dataset serving evaluation in Appendix G uses an H100 GPU. The detailed parameters can be found in Table 8.

F Results across Random Seeds

Table 9 reports the mean and standard deviation over three different random seeds for Qwen3-4B on all four benchmarks. These aggregate results supplement the checkpoint results in Table 1; their means and standard deviations describe the three-seed evaluation rather than the individual checkpoint

Table 9: Qwen3-4B results across three random seeds, reported as mean ± standard deviation. Accuracy is in percent; S.Len, D.Len, and T.Len denote streaming, deep, and total generated lengths in tokens. These three-seed aggregates are reported separately from the fixed-reference results in Table 1, so their means need not equal the corresponding entries in that table.

Method	Acc (%)↑	S.Len↓	D.Len↓	T.Len↓
GSM-symbolic P1				
Read-then-think	63.9 ± 1.3	–	1714.8 ± 43.2	1714.8 ± 43.2
StreamingThinker (SFT)	93.2 ± 0.8	169.1 ± 6.0	120.4 ± 4.8	289.5 ± 9.6
GRPO	94.4 ± 0.9	158.726 ± 5.841	112.038 ± 4.316	270.764 ± 8.295
TSDPO	95.2 ± 0.7	151.934 ± 5.127	115.486 ± 4.594	267.420 ± 7.882
GSM-symbolic P2				
Read-then-think	46.6 ± 1.5	–	1941.7 ± 56.4	1941.7 ± 56.4
StreamingThinker (SFT)	81.3 ± 1.1	229.5 ± 9.4	129.1 ± 5.7	358.6 ± 12.1
GRPO	87.2 ± 1.0	211.284 ± 8.026	125.417 ± 5.214	336.701 ± 10.472
TSDPO	88.7 ± 0.9	197.843 ± 7.382	130.074 ± 5.639	327.917 ± 9.956
MetaMathQA				
Read-then-think	83.1 ± 1.0	–	1375.6 ± 40.7	1375.6 ± 40.7
StreamingThinker (SFT)	85.8 ± 0.9	153.4 ± 5.8	123.1 ± 4.9	276.5 ± 8.8
GRPO	90.6 ± 0.8	141.327 ± 5.972	114.682 ± 4.641	256.009 ± 8.104
TSDPO	92.7 ± 0.6	123.146 ± 5.284	114.709 ± 4.516	237.855 ± 7.492
PubMedQA				
Read-then-think	61.1 ± 1.4	–	334.9 ± 16.8	334.9 ± 16.8
StreamingThinker (SFT)	75.9 ± 1.2	399.6 ± 14.7	117.3 ± 5.2	516.9 ± 15.8
GRPO	78.9 ± 1.1	381.506 ± 13.284	110.936 ± 4.472	492.442 ± 14.391
TSDPO	81.0 ± 0.8	354.917 ± 12.506	115.738 ± 4.731	470.655 ± 13.726

values. Accuracy is expressed as a percentage and its standard deviation in percentage points, while reasoning lengths are in tokens.

TSDPO has higher mean accuracy and lower mean total length than GRPO on every benchmark. Mean accuracy increases from 94.4 to 95.2 on GSM-symbolic P1, from 87.2 to 88.7 on P2, from 90.6 to 92.7 on MetaMathQA, and from 78.9 to 81.0 on PubMedQA. These descriptive results are consistent with the main comparison. Three seeds do not by themselves establish statistical significance, and corresponding uncertainty estimates are not available for Qwen3-1.7B.

G Per-Dataset Post-Input Latency

Timing definitions and evaluation setup. We evaluate Qwen3-1.7B with vLLM on an H100 GPU, repeating evaluation three times for each dataset. Following StreamingThinker (Tong et al., 2025), input arrives at a simulated rate of 150 words per minute. Here, *real-time streaming* means that reasoning can begin during input reception; it does not specify a hard response deadline. Let t_0, t_{in}, and t_{out} denote the arrival of the first input, arrival of the last input, and completion of the final answer, respectively. We distinguish post-input completion latency from end-to-end latency:

$$L_{\text{post}} = t_{\text{out}} - t_{\text{in}}, \qquad L_{\text{e2e}} = t_{\text{out}} - t_0 = (t_{\text{in}} - t_0) + L_{\text{post}}. \tag{22}$$

Neither quantity is the time to the first intermediate reasoning output.

What the reported latency measures. Table 10 reports throughput-based estimates of post-input completion latency. For read-then-think, the estimate divides total generated length by decoding throughput; for streaming methods, it divides final deep-reasoning length by throughput, since intermediate reasoning overlaps input reception. The means and reported variation summarize three evaluation runs. Table 5 averages the four dataset means with equal weight; its speedups are ratios of mean latencies, rather than averages of per-dataset speedups. These estimates characterize the

Table 10: Per-dataset latency analysis for Qwen3-1.7B with vLLM on an H100 GPU at 150 input words per minute. Throughput is in generated tokens per second. Post-input latency is estimated from generated length and throughput: total length for read-then-think and deep-reasoning length for streaming methods. Latency entries show the mean and reported variation over three evaluation runs; they exclude input reception time.

Dataset	Method	Throughput↑	Post-input latency (s)↓
GSM-symbolic P1	Read-then-think	141.8	11.27 ± 0.28
	StreamingThinker (SFT)	135.2	0.87 ± 0.09
	GRPO	136.9	0.98 ± 0.10
	TSDPO	137.4	0.96 ± 0.09
GSM-symbolic P2	Read-then-think	140.7	13.27 ± 0.33
	StreamingThinker (SFT)	134.4	0.99 ± 0.10
	GRPO	136.2	1.09 ± 0.12
	TSDPO	136.7	1.17 ± 0.13
MetaMathQA	Read-then-think	141.4	9.07 ± 0.24
	StreamingThinker (SFT)	135.0	0.90 ± 0.09
	GRPO	136.8	1.16 ± 0.12
	TSDPO	137.2	0.94 ± 0.09
PubMedQA	Read-then-think	140.9	3.08 ± 0.15
	StreamingThinker (SFT)	134.6	0.90 ± 0.08
	GRPO	136.4	1.05 ± 0.11
	TSDPO	136.9	1.00 ± 0.10

remaining decoding work, rather than a direct measurement of every component of serving delay. They do not separately measure input prefill, scheduling, communication, or unfinished streaming work at the last input arrival. First-output latency and full end-to-end latency are not reported in this table.

Per-dataset trade-offs. TSDPO's estimated post-input latency ranges from 0.94 to 1.17 seconds, compared with 3.08–13.27 seconds for read-then-think. Relative to GRPO, TSDPO reduces latency by 0.02 seconds on GSM-symbolic P1, 0.22 seconds on MetaMathQA, and 0.05 seconds on PubMedQA, but increases it by 0.08 seconds on GSM-symbolic P2. This exception is consistent with Table 1: P2 deep-reasoning length rises from 148.704 to 160.210 tokens, although total length falls from 384.488 to 370.256 tokens and accuracy rises from 0.758 to 0.788. Thus, fewer total tokens need not imply a shorter wait after input completion. StreamingThinker (SFT) has the lowest estimated post-input latency on each dataset, whereas TSDPO improves its accuracy on all four. These are descriptive comparisons; three runs and the reported variation do not establish significance for small latency differences.

Overlap and input-rate sensitivity. An idealized decomposition makes the assumptions explicit:

$$L_{\text{post}} \approx B + \frac{\ell_{\text{deep}}}{r} + \delta, \qquad t_{\text{in}} - t_0 \approx \frac{60 N_{\text{words}}}{v}, \tag{23}$$

where B is residual streaming backlog in seconds, ℓ_{deep} is deep-reasoning length, r is decoding throughput in tokens per second, δ collects other serving overhead, and v is input speed in words per minute. The reported streaming estimate uses the ℓ_{deep}/r term. Throughput spans 134.4–141.8 tokens per second; comparing this directly with 150 words per minute would require a token-to-word conversion and does not alone establish that backlog is negligible. If streaming work keeps pace with input, backlog remains small, and deep-reasoning length is unchanged, moderate changes in input rate can leave post-input latency approximately unchanged. End-to-end latency still includes input reception time. Faster or burstier arrival may create backlog or alter reasoning, invalidating this approximation. We evaluate one input rate and do not claim empirical robustness across input speeds.

Inference cost versus training cost. Answer-conditioned scoring is used only during training, so it is absent from these inference estimates. Its additional forward-pass cost is separate from serving

latency; the present measurements do not quantify training-time overhead or establish equal training cost to GRPO.