PARALLEL TEST-TIME SCALING WITH MULTI-SEQUENCE VERIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parallel test-time scaling—generating multiple candidate solutions for a single problem—is a powerful technique for improving large language model performance, but it faces two related challenges: selecting the correct solution from a large candidate pool is difficult, and generating every candidate to completion can be slow. Predicting candidate correctness can address both problems by guiding answer selection and informing when to stop generation. However, verifiers that score candidates independently cannot exploit the relationships between candidates' underlying representations. To address this, we introduce the Multi-Sequence Verifier (MSV), a lightweight verifier that predicts each candidate's correctness by jointly attending over the hidden representations of all sampled candidates, rather than processing each one in isolation. We apply MSV's predictions to best-of-N selection, confidence estimation, and parallel early stopping, evaluating each use case separately. Across challenging mathematical reasoning benchmarks, MSV improves best-of-64 accuracy by up to 6% relative to the strongest baselines, and a streaming variant of MSV improves the estimated accuracy-latency tradeoff over the evaluated baseline verifiers.

1 INTRODUCTION

Large language models (Brown et al., 2020) have become increasingly powerful, with much of their performance unlocked by test-time scaling strategies. One of the most effective strategies is parallel scaling, where a model generates multiple, independent candidate solutions for a single problem (Wang et al., 2022; Lightman et al., 2024; Snell et al., 2024). However, parallel scaling faces two bottlenecks that limit its usage: (1) the selection problem, as accurately identifying the correct solution from a large pool of candidates is difficult, and (2) the high inference latency required to generate numerous full solutions.

Both bottlenecks motivate learning to estimate candidate correctness (Cobbe et al., 2021; Lightman et al., 2024). These estimates serve distinct purposes. Best-of-N selection uses their ordering to choose an answer, whereas confidence estimation requires their numerical values to reflect empirical correctness rates (Guo et al., 2017; Kadavath et al., 2022). Early stopping uses the scores of intermediate answers to decide when to terminate generation (Zhang et al., 2025a; Yang et al., 2025). A useful verifier should therefore distinguish correct from incorrect candidates and provide informative confidence estimates as decoding progresses. We study whether conditioning on multiple candidates improves these predictions and their downstream uses.

Scoring candidates independently leaves the relationships between sampled solutions unused. Self-consistency (Wang et al., 2022) and weighted voting (WV; Li et al., 2022) exploit answer agreement, but their aggregation rules retain only a limited view of the candidate set. Self-consistency reduces each answer class to a count; weighted voting sums independently computed verifier scores within each class. Neither rule learns how the candidates' representations relate to one another. As a result, a rare correct answer can be outweighed by repeated incorrect answers, even when its individual verifier score is higher. Errors in the equivalence checker can further distort the scores by splitting equivalent answers into separate groups or merging non-equivalent answers into one group. More generally, when aggregation uses only the processed candidate information, the optimal accuracy of its selected answer cannot exceed that of a selector observing the joint inputs (Section 3.1). These limitations motivate learning how to combine agreement with the evidence in candidate representations, rather than prescribing a fixed aggregation rule.

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 introduce the Multi-Sequence Verifier (MSV), a non-generative verifier trained to predict each candidate's correctness from the representations of multiple sampled solutions and their answer agreement. By learning from these inputs jointly, MSV can use cross-candidate evidence that is unavailable to independent scoring and compressed away by fixed aggregation. In the Terminal Answers setting, MSV improves best-of-N selection and the confidence estimates associated with selected answers. On challenging math reasoning benchmarks, MSV improves best-of-64 accuracy by up to 6% relative to the best baselines.

We also study parallel early stopping, which uses intermediate answers from concurrently generated candidates to decide when to terminate the entire run. MSV learns to predict answer correctness jointly from candidate hidden states and agreement, using these predictions to select an available answer and stop all remaining generation. A streaming variant of Multi-Sequence Verifier (MSV) scores intermediate answers using the cross-sequence information available at answer completion and triggers this decision when a score reaches a threshold. In this framework, streaming MSV improves the estimated accuracy–latency tradeoff over the evaluated baseline verifiers. On MATH, it reaches the peak accuracy of the single-sequence verifier baselines with less than half their average stopping token position.

Our contributions can be summarized as follows:

- We propose the Multi-Sequence Verifier (MSV), a verifier architecture that predicts candidate correctness by modeling cross-sequence interactions.
- We demonstrate improvements in best-of-N selection accuracy and in the calibration of confidence estimates for the selected answers.
- We introduce a streaming MSV variant and use its set-conditioned correctness predictions to terminate the entire parallel decoding job, improving the estimated accuracy–latency tradeoff over the evaluated baseline verifiers.

2 Related Work

Parallel test-time scaling. A common approach to selecting among decoded sequences is to train a verifier to score candidate answers (Cobbe et al., 2021; Zhang et al., 2024). Self-consistency (Wang et al., 2022) and weighted voting (Li et al., 2022) use agreement across sequences to improve answer selection. Generative approaches also compare candidates in context (Chen et al., 2023; Liu et al., 2025; Toshniwal et al., 2025); for example, GenSelect (Toshniwal et al., 2025) prompts an LLM to select a solution from a group of candidates. MSV jointly scores candidate answers using a single transformer layer over their hidden representations. Our comparison with GenSelect in Section B.10 illustrates the verification overhead of generative selection in our evaluation setup, motivating a lightweight verifier for repeated intermediate-answer evaluation.

Adaptive scaling for efficiency. Adaptive scaling methods adjust sampling budgets and terminate candidate trajectories early (Li et al., 2024; Aggarwal et al., 2023; Xue et al., 2023; Huang et al., 2025). DeepConf combines confidence-based trace filtering with consensus-based stopping of further sampling (Fu et al., 2025). Pruning methods (Sun et al., 2024; Hong et al., 2025; Cheshmi et al., 2025; Wang et al., 2026) discard selected trajectories while continuing to decode those retained. Global stopping of parallel generation is also explored by Zheng et al. (2026), who stop when intermediate-answer consensus stabilizes, and Zhang et al. (2026), who stop when votes weighted by temporal confidence concentrate. Both use training-free controllers. Building on intermediate-answer extraction (Zhang et al., 2025a; Yang et al., 2025), MSV instead learns correctness predictions by jointly processing candidate hidden states and agreement, then uses these predictions to select an available answer and terminate the entire run. Pruning is complementary to this stopping rule and could reduce the active candidate set before MSV decides to end generation.

3 Method

3.1 Problem Setup

We consider *parallel decoding*, where an LM generates N sequences concurrently for a query q, sampling one token per active sequence at each decoding step. We study two settings: **Terminal Answers**, which uses final answers after decoding is complete, and **Streaming Answers**, which also uses intermediate answers extracted during decoding.

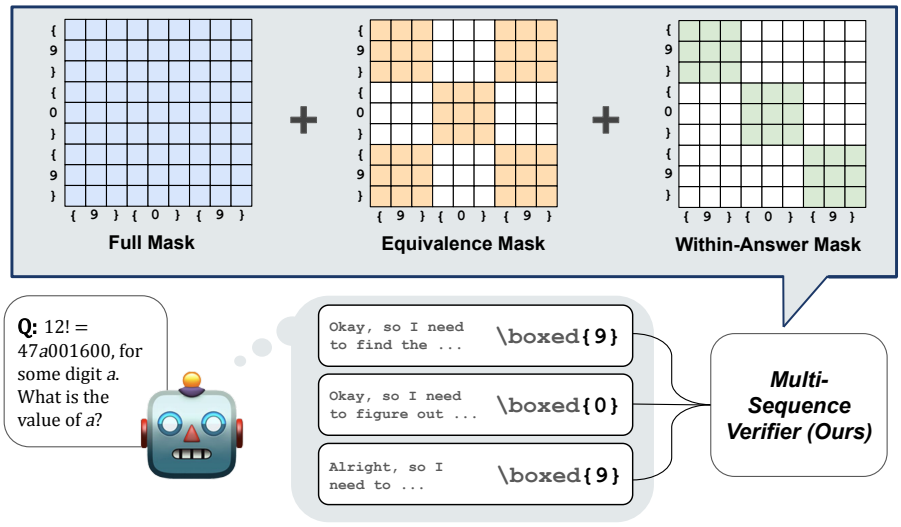


Figure 1: Illustration of our Multi-Sequence Verifier (MSV) that uses multiple attention masks in its Multi-Mask Transformer Block (MMTB) to predict the correctness of each answer. MSV takes answer-token hidden states as inputs; the masks determine which tokens exchange information within and across sequences.

Terminal Answers. Given a query q, the LLM decodes N parallel sequences until they all reach termination. To extract an explicit answer, we append an elicitation prompt such as `### Final Answer ### \boxed` to the end of each nth sequence, prompting the model to produce a boxed answer $a^{(n)}$. This results in one answer per sequence, denoted by $\{a^{(n)}\}_{n=1}^{N}$.

Streaming Answers. Unlike the **Terminal Answers** setting, we additionally extract *intermediate answers* whenever a delimiter, e.g., the token "Wait", is encountered. This approach of extracting intermediate answers at delimiters has been explored in prior work for single-sequence early stopping (Zhang et al., 2025a; Yang et al., 2025), which we generalize to the parallel decoding setting. Specifically, when the kth delimiter appears in the nth sequence, we immediately branch that sequence, append the elicitation prompt `### Final Answer ### \boxed`, and obtain an intermediate answer $a_k^{(n)}$. We also extract the terminal answer at the end of each sequence. Thus, if a sequence contains $K^{(n)} - 1$ delimiters, it may yield multiple answers $\{a_k^{(n)}\}_{k=1}^{K^{(n)}}$, where $a_{K^{(n)}}^{(n)}$ is the terminal answer. For notational consistency, in the **Terminal Answers** setting, we regard the single terminal answer as $a_1^{(n)}$ and $K^{(n)} = 1$.

Along with each answer, we store the representations of the answer tokens for later use with our Multi-Sequence Verifier. Specifically, if $a_k^{(n)}$ consists of $L_k^{(n)}$ tokens, we store $\{h_{k,i}^{(n)}\}$ for $i = 1, \ldots, L_k^{(n)}$, where $h_{k,i}^{(n)} \in \mathbb{R}^d$ denotes the d-dimensional hidden state of the ith token, output by the last transformer layer of the LLM.

Correctness and objective. Let a^* denote the ground-truth answer for the query q, and let the symbol $\sim$ be an equivalence relation that captures symbolic or semantic equality, e.g., SymPy (Meurer et al., 2017; Lewkowycz et al., 2022). Then, we define the correctness of each candidate answer $a_k^{(n)}$ as follows:

$$y_k^{(n)} = \mathbf{1}\left[a_k^{(n)} \sim a^*\right].$$

Our goal is to predict each candidate's correctness for answer selection, confidence estimation, and early stopping.

Theoretical motivation for joint verification. In **Terminal Answers**, candidate n has answer $a^{(n)}$, hidden states $h^{(n)} = \{h_i^{(n)}\}_{i=1}^{L^{(n)}}$, and correctness $y^{(n)}$. Write $X_n = (a^{(n)}, h^{(n)})$ for its verifier input. Candidate-wise processing $X_n \longrightarrow Z_n$ retains only $Z_n = (a^{(n)}, s^{(n)})$, where $s^{(n)}$ is an independently computed verifier score. A selection rule π then acts on $Z_{1:N}$, for example by choosing the highest-scoring candidate (best-of-N) or using weighted voting over equivalent answers. For any observation

O, let $V(O) := \sup_\pi \mathbb{E}[y^{(\pi(O))}]$ denote the optimal expected correctness. Since $Z_{1:N}$ is a garbling of $X_{1:N}$, Blackwell's informativeness theorem (Blackwell, 1953) gives

$$V(X_{1:N}) \geq V(Z_{1:N}).$$

The final answers and scores no longer expose the hidden states or the relationships among them. Selection rules acting on these summaries cannot recover any selection-relevant information discarded by independent scoring, which can reduce optimal expected utility. This motivates MSV's joint processing of candidate representations before producing scores.

3.2 Multi-Sequence Verifier

MSV predicts answer correctness by combining contextual answer representations with information about the other available candidates. We use answer-token hidden states because they are conditioned on the query and preceding reasoning, while keeping the verifier input shorter than the full reasoning traces. This compact readout gives the verifier access to features of the reasoning history without directly attending over every reasoning token. The architecture combines evidence from individual answers, relationships between candidates, and explicit agreement across sequences. Figure 1 illustrates the architecture, and Algorithm 1 gives its forward pass.

Input representation for MSV. Let t be a time index shared across parallel sequences, where all the sequences start generating from $t = 0$ and they generate one token at a time. Let $\tau_k^{(n)}$ denote the start time of decoding the kth elicited answer $a_k^{(n)}$ from the nth sequence, so that $\tau_k^{(n)} + L_k^{(n)}$ is its completion time. At readout time t, MSV receives the representations of all completed answers:

$$U^{(t)} = \mathrm{concat}\left(\left\{\left\{\{h_{k,i}^{(n)} + \mathsf{e}^{(n)}\}_{i=1}^{L_k^{(n)}}\right\}_{\tau_k^{(n)}+L_k^{(n)}\leq t}\right\}_{n=1}^{N}\right).$$

Here, $\mathrm{concat}(\cdot)$ denotes concatenation of all the representations along the sequence dimension, without adding any special separators between sequences. $\mathsf{e}^{(n)}$ is a learnable per-sequence embedding added to identify the sequence from which each token representation originates.

Multi-mask transformer blocks. To process the aggregated representation $U^{(t)}$, we introduce a **M**ulti-**M**ask **T**ransformer **B**lock (MMTB). The masks expose complementary relationships: global candidate context, revisions within a reasoning sequence, agreement between answers with the same extracted result, and the token structure of an individual answer. Separate attention outputs let the verifier learn how to combine these relationships without requiring unrestricted attention to discover every grouping. Each mask is applied to the same input, $U^{(t)}$. As in standard multi-head attention (Vaswani et al., 2017), we begin by computing the usual linear projections of $U^{(t)}$ into query, key, and value matrices $(Q_h^{(t)}, K_h^{(t)}, V_h^{(t)})_{h=1}^H$ where H is the number of heads and $d_{\mathrm{head}} = d/H$ for model width d. For a fixed collection of J masks $\{M_j\}_{j=1}^J$, we then compute the output for each mask:

$$A_{h,j}^{(t)} \;=\; \mathrm{softmax}\left(\frac{Q_h^{(t)}(K_h^{(t)})^\top}{\sqrt{d_{\mathrm{head}}}} \;+\; \log M_j\right) V_h^{(t)},$$

where $\log M_j$ adds 0 to permitted entries and $-\infty$ to masked entries, with $\mathrm{softmax}$ applied row-wise. The masked outputs are then combined via learnable mixture weights $\mathbf{w}_h = (w_{h,1}, \ldots, w_{h,J})$ for each head h:

$$\tilde{U}^{(t)} \;=\; \mathrm{concat}_{h=1}^H\left(\sum_{j=1}^J \alpha_{h,j} A_{h,j}^{(t)}\right) W_O, \quad \boldsymbol{\alpha}_h \;=\; \mathrm{softmax}(\mathbf{w}_h).$$

Here, heads are concatenated along the feature dimension, and the learned output projection $W_O \in \mathbb{R}^{d\times d}$ maps their combined output to the residual width. The collection of masks $\{M_j\}_{j=1}^J$ is a fixed hyperparameter of our model. In our instantiation, we use four complementary masks. The *full* mask permits all interactions, $(M_{\mathrm{full}})_{u,v} = 1$ for all positions u, v, enabling attention across all tokens in all sequences. The *within-sequence* mask restricts attention to tokens originating from the same sequence,

$$(M_{\mathrm{ws}})_{u,v} \;=\; \mathbf{1}\big[\mathrm{seq}(u) = \mathrm{seq}(v)\big],$$

allowing intermediate and terminal answers from one reasoning trajectory to be compared for revisions and consistency. Here, $\mathrm{seq}(u)$ denotes the index n of the sequence containing the token

u. The *equivalence* mask allows attention only between tokens whose answers are symbolically equivalent,

$$(M_{\text{eq}})_{u,v} \;=\; \mathbf{1}\big[\text{ans}(u) \;\sim\; \text{ans}(v)\big],$$

where $\text{ans}(u)$ is the answer $a_k^{(n)}$ containing the token u. Contextual representations of the same extracted result can differ because they arise from different reasoning histories. This mask uses candidate-to-candidate equivalence and does not require the ground-truth answer. Finally, the *within-answer* mask allows attention only between tokens inside a single answer instance $a_k^{(n)}$,

$$(M_{\text{wa}})_{u,v} \;=\; \mathbf{1}\big[(\text{seq}(u), \text{step}(u)) = (\text{seq}(v), \text{step}(v))\big],$$

where $\text{step}(u)$ identifies the answer step k to which the token belongs.

In the **Terminal Answers** setting, the within-sequence mask and within-answer mask are equivalent, reducing the number of masks to three. We further restrict the attention masks to be "causal" in the **Streaming Answers** setting, meaning that an answer $a_k^{(n)}$ may attend to $a_{k'}^{(n')}$ only if $\tau_k^{(n)} + L_k^{(n)} \geq \tau_{k'}^{(n')} + L_{k'}^{(n')}$. This is causality at the level of completed answers: all tokens within a completed answer are available together, but later-completing answers cannot affect its score. The same restriction applies during offline training, so collecting a full rollout does not expose future answers to an earlier prediction. We provide an ablation study for each mask in Section B.2.

The block's final output is then computed using standard residual connections and an MLP layer:

$$Z^{(t)} \;=\; (U^{(t)} + \tilde{U}^{(t)}) + \text{MLP}\big(\text{LN}(U^{(t)} + \tilde{U}^{(t)})\big),$$

Feature augmentation. The equivalence mask identifies equivalent answers but does not explicitly quantify agreement. We add the supporting fraction $\gamma_k^{(n)} \in [0, 1]$. In **Streaming Answers**, this fraction is computed when $a_k^{(n)}$ completes, using only each sequence's latest completed answer and excluding sequences with none. In **Terminal Answers**, it uses all N terminal answers. We project $\gamma_k^{(n)}$ through an MLP and add it to the answer's last-token representation from $Z^{(t)}$:

$$\bar{z}_k^{(n)} = z_{k,L_k^{(n)}}^{(n)} + \text{MLP}(\gamma_k^{(n)}).$$

This readout attends to all tokens in the answer and other answers permitted by the masks.

Constructing final predictions. To predict the correctness of $a_k^{(n)}$, we apply a linear head to its augmented representation and pass the resulting logit through a sigmoid:

$$\tilde{y}_k^{(n)} \;=\; \sigma\big(\mathbf{w}^\top \bar{z}_k^{(n)} + b\big),$$

where σ denotes the sigmoid function and $\mathbf{w}, b$ are learnable parameters. Additionally, in the **Terminal Answers** setting, instead of computing each $\tilde{y}_1^{(n)}$ independently, we first average the *logits* of symbolically equivalent terminal answers and then apply the sigmoid function as follows,

$$\tilde{y}_1^{(n)} = \sigma\left(\frac{1}{|\mathcal{C}(n)|} \sum_{m \in \mathcal{C}(n)} \big(\mathbf{w}^\top \bar{z}_1^{(m)} + b\big)\right) \tag{1}$$

where $\mathcal{C}(n) := \{m \in \{1, \ldots, N\} \mid a_1^{(m)} \sim a_1^{(n)}\}$. Logit averaging incorporates our a priori knowledge that symbolically equivalent answers share the same correctness label. We do not apply this averaging in **Streaming Answers**: scores are produced at different completion times and condition on different available evidence. Keeping a separate score for each completion preserves that causal readout, whereas averaging over the final equivalence class would introduce future information. We evaluate a causal averaging variant in Section B.4.

Training and Inference. For the **Streaming Answers** scenario, we run the sequences until all of them terminate, collect all the intermediate and final answers produced, and train the parameters to minimize the binary cross-entropy loss. Let T be the global time step when all the sequences terminate. We first compute $Z^{(T)}$, compute the predictions $\tilde{y}_k^{(n)}$, and minimize

$$\mathcal{L} := \sum_{n,k} \text{BCE}(\tilde{y}_k^{(n)}, y_k^{(n)}),$$

where the summation is over all valid n, k pairs. For the **Terminal Answers** scenario, we proceed similarly, except that we minimize BCE only for the terminal answers. BCE is a strictly proper loss: its population risk is minimized by the conditional probability of correctness given the verifier's input (Blasiok et al., 2023).

At inference time t, we compute $Z^{(t)}$ and predict $\{\{\tilde{y}_k^{(n)}\}_{\tau_k^{(n)}+L_k^{(n)}\le t}\}_{n=1}^{N}$. The causal attention structure in **Streaming Answers** enables an efficient online implementation via KV caching, described in Section C.2.

3.3 Applications

We use MSV's correctness predictions for best-of-N selection in **Terminal Answers** and for both stopping and selection in **Streaming Answers**.

Best-of-N selection and confidence estimation. In the **Terminal Answers** scenario, we use the verifier scores for *best-of-N* decoding (Nakano et al., 2021). From N candidate answers, we select the one, $a^\dagger$, with the highest score:

$$n^\dagger = \arg\max_n \tilde{y}_1^{(n)}, \qquad a^\dagger = a_1^{(n^\dagger)}.$$

Best-of-N selection depends only on the ordering of the final candidate scores. The selected score, $\tilde{y}^\dagger = \tilde{y}_1^{(n^\dagger)}$, also serves as an estimate of the selected answer's probability of correctness. These are distinct properties: a shared strictly increasing transformation of the final scores preserves the selected answer but can change its reported confidence. We therefore evaluate selection accuracy and post-selection calibration separately (see Figure 4).

Early stopping of parallel decoding. In the **Streaming Answers** setting, multiple sequences are decoded in parallel, and each intermediate answer $a_k^{(n)}$ yields a set-conditioned correctness score $\tilde{y}_k^{(n)}$. We maintain the candidate streams until they finish naturally or the stopping rule terminates all remaining streams. The rule stops the entire parallel decoding job once the highest available score reaches a threshold λ:

$$t^* \;=\; \min\left\{ t : \max_{n,k:\, \tau_k^{(n)}+L_k^{(n)}\le t} \tilde{y}_k^{(n)} \;\ge\; \lambda \right\}.$$

At t^*, we output the candidate with the highest available score. The threshold $\lambda \in [0, 1]$ controls the accuracy–latency tradeoff. Our experiments sweep this threshold to characterize the tradeoff (Section A). Calibration measured over intermediate answers does not by itself guarantee that answers selected by this stopping rule achieve accuracy λ; we evaluate the stopping policy through observed accuracy, token position, and estimated latency. If all sequences terminate without a score reaching the threshold, we select the terminal answer with the highest score $\tilde{y}_{K^{(n)}}^{(n)}$.

4 Experiments

4.1 Experimental Setup

We evaluate correctness prediction, answer selection, and early stopping with MSV. We denote MSV trained on groups of N sequences by MSV_N. We compare against three single-sequence verifiers: (1) "Probe" (Zhang et al., 2025a), which is a lightweight MLP operating on the last-token representation, (2) the open-source verifier Qwen2.5-Math-PRM-7B (PRM; Zhang et al., 2025b), which is a 7-billion-parameter model, and (3) MSV_1, which shares MSV_N's architecture but is trained and evaluated on a single sequence at a time. Comparing MSV_N with MSV_1 tests the benefit of training and evaluating the same verifier architecture with access to multiple candidates.

To compare MSV against heuristic aggregation, we adopt weighted voting (WV; Li et al., 2022), which aggregates correctness probabilities by summing them within each class of symbolically equivalent answers, and normalizing the summed probabilities across classes. In the experiments, these extensions are denoted as MSV_1+WV, Probe+WV, and PRM+WV. Self-consistency (Wang et al., 2022) is a verifier-free baseline that scores each answer by its frequency among sampled solutions. An additional comparison with GenSelect (Toshniwal et al., 2025) is reported in Section B.10.

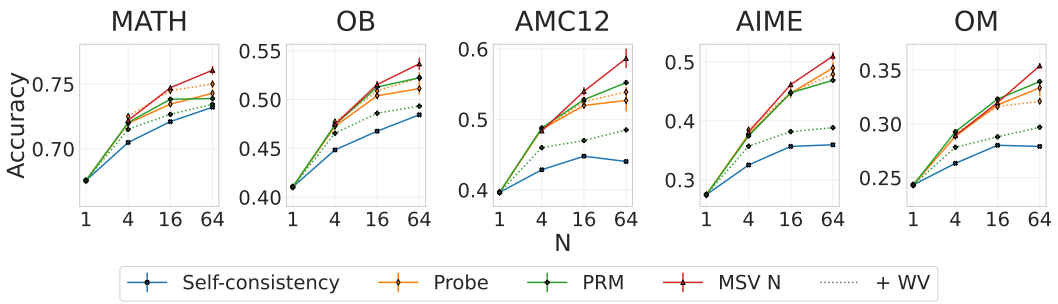


Figure 2: Best-of-N accuracy versus N in the **Terminal Answers** setting. Accuracy measures whether the answer selected from N sampled candidates is correct.

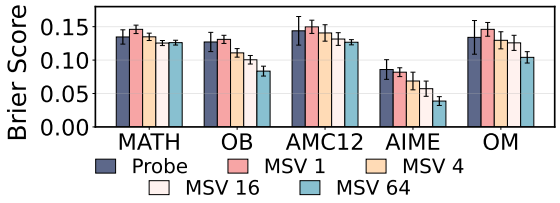


Figure 3: Brier scores (↓) in the **Terminal Answers** setting. Scores evaluate candidate-level correctness probabilities; lower values indicate more accurate probability predictions.

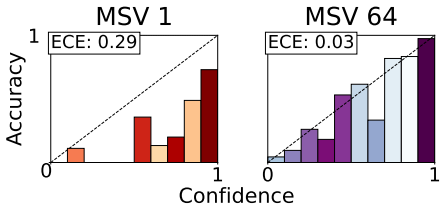


Figure 4: Reliability diagrams of verifier confidence on best-of-64 answers (AIME), on one representative seed. The diagonal represents perfect calibration.

We use the DeepSeek-R1-Distill-Qwen-1.5B model (Guo et al., 2025) as the base LM; experiments with three additional base models are reported in Section 4.5. The DeepMath-103K dataset (He et al., 2025) is used for training and validation of MSV and Probe, and evaluation is conducted on MATH (Hendrycks et al., 2021), OlympiadBench (OB; He et al., 2024), AMC12, AIME, and Omni-MATH (OM; Gao et al., 2024). For the main mathematical reasoning experiments, values reported with uncertainty are means and standard deviations over five random seeds. These seeds vary verifier initialization and training, with generated candidates and evaluation problems held fixed.

We provide further details of the experimental setup in Section A, along with a full report of all main experimental results in Section D. Section B reports additional ablations and experiments. Notably, we show in Section B.6 and Section B.7 that the training group size does not fix the inference candidate budget: without retraining, MSV_{64} can process 128 or 256 candidates via fixed-size grouping, or be transferred directly to 16 candidates.

4.2 Terminal Answers

Correctness prediction on Terminal Answers. Figure 3 shows the Brier scores of the single-sequence verifier baselines and MSV_N. The full results for AUROC, Brier score, ECE, and NLL can be found in Tables 15 and 16. MSV_N achieves lower Brier scores than Probe and MSV_1 in these comparisons. MSV_{64} achieves a ∼50% reduction in Brier score compared to Probe, on the AIME dataset. MSV_1 performs similarly to or worse than Probe in these comparisons, suggesting that the more expressive transformer block alone does not explain the gains of MSV_N, and that the additional information from multiple candidates is necessary for improved performance.

Best-of-N prediction. We apply MSV to best-of-N answer selection. Full numerical results are reported in Table 17. As shown in Figure 2, MSV_N improves best-of-N accuracy over single-sequence verifier baselines, with gains becoming more pronounced starting from $N = 16$. At $N = 64$, MSV_{64} achieves best-of-N accuracy of 58.7% on AMC12, compared to 55.2% for the strongest baseline, PRM, yielding approximately 6% relative improvement. We also see that self-consistency performs the worst across all datasets, and weighted voting (WV) does not provide a consistent improvement.

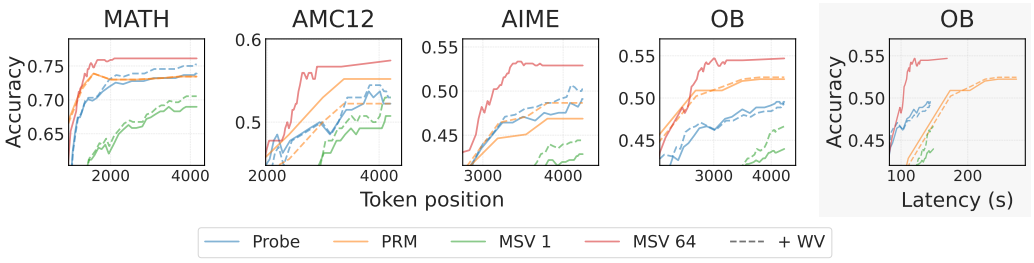


Figure 5: Accuracy tradeoffs in **Streaming Answers**, with $N = 64$. The first four panels show stopping token position; the rightmost shows estimated latency on OlympiadBench. Each curve sweeps the stopping threshold. Higher accuracy at the same stopping token position or estimated latency indicates a better tradeoff. Latency estimation is detailed in Section B.1.

Table 1: Ablated best-of-16 accuracy (AIME). ✓/× indicate included/removed components.

Feature γ	Equiv. mask	Accuracy
✓	✓	**0.4615**±0.0046
×	✓	0.4341±0.0100
✓	×	0.4550±0.0057
×	×	0.3717±0.0058

Table 2: Accuracy and latency of parallel early-stopping methods on AIME with $N = 16$.

Method	Accuracy	Latency (s)
ST-BoN	0.275	50.0
Parallel-Probe	0.337	52.3
MSV ($\lambda = 0.4$)	0.389	49.8
MSV ($\lambda = 1.0$)	0.456	66.8

Calibrated best-of-N scores. We also evaluate whether the confidence assigned to a selected answer reflects its correctness. A reliability diagram groups selected answers by predicted confidence and compares their mean confidence with observed accuracy. Points on the diagonal indicate perfect calibration; points below it indicate overconfidence. The reliability diagrams (Guo et al., 2017) in Figure 4 and the calibration metrics reported in Table 18 show that MSV_N is more calibrated on its own chosen answers than the single-sequence verifier baselines are on their chosen answers.

Agreement features and equivalence attention. The agreement feature supplies an explicit agreement fraction, while the equivalence mask lets matching answers exchange information; ablating both tests their individual and combined contributions. We jointly ablate the agreement feature γ and equivalence attention mask on AIME in **Terminal Answers**, retraining all four variants. Table 1 shows that both components improve mean best-of-16 accuracy when the other is absent, and their combination yields the highest mean. Removing both lowers best-of-16 accuracy from 46.15% to 37.17%. Further analysis is in Section B.3.

4.3 Streaming Answers

Streaming Answers uses intermediate correctness predictions to decide both which answer to return and when to stop generation. We use the "Wait" token as the delimiter, and experiments with two other delimiters in Section B.8 show similar trends for the tested delimiters. Results in Table 19 show that MSV improves discrimination and probabilistic prediction quality relative to single-sequence verifier baselines. The Brier-score curves on AIME in Figure 8 show improvements across token-position ranges, particularly later in decoding. Ablations of the attention masks and logit averaging in **Streaming Answers** are reported in Sections B.2 and B.4.

Parallel early stopping. We evaluate whether joint correctness predictions allow the entire parallel run to stop earlier while maintaining answer-selection accuracy, using both token-position and estimated wall-clock tradeoffs. We estimate latency by summing measured generation time with estimated verifier and other component costs (see Section B.1 for details). As described in Section 3.3, varying λ produces an accuracy–token-position tradeoff curve for each verifier, shown in Figure 5. On MATH, we find that the maximum achievable accuracy with single-sequence verifier baselines can be achieved by MSV_{64} with less than half the average stopping token position. The estimated wall-clock comparison on OlympiadBench in the rightmost panel of Figure 5 accounts for the different verification costs. PRM incurs a large wall-clock verifier inference cost due to its large number of parameters. The full curves for all datasets are in Figure 9.

Table 3: Best-of-64 accuracy on Omni-MATH under different equivalence checkers. Full results are in Table 21.

Checker	Checker accuracy	Probe+WV	MSV_{64}
SymPy (reference)	100.0%	0.321±0.002	**0.348**±0.007
Jaccard (≥ 0.8)	97.0%	0.316±0.002	**0.345**±0.008
Embedding (≥ 0.85)	89.4%	0.301±0.002	**0.344**±0.008

Table 4: Brier score and best-of-64 across different base LMs. Full results are in Tables 22 and 23.

	Brier		Best-of-64	
Base LM	**Best**	MSV_{64}	**Best**	MSV_{64}
R1-8B	0.067±0.010	**0.036**±0.009	0.649±0.009	**0.663**±0.005
Qwen3	0.077±0.006	**0.051**±0.025	0.529±0.009	**0.536**±0.011
Llama 3.2	0.132±0.005	**0.057**±0.006	0.338±0.005	**0.378**±0.005

Comparison with other parallel early stopping baselines. We compare with ST-BoN (Wang et al., 2026) and Parallel-Probe (Zheng et al., 2026). ST-BoN selects one trajectory early using cross-sample consistency and terminates the others, while Parallel-Probe periodically elicits intermediate answers and stops or prunes based on majority-consensus stability. In contrast, MSV learns set-conditioned correctness scores from cross-sequence representations and answer agreement, and stops all remaining generation once the highest score exceeds λ. In Table 2, MSV with $\lambda = 0.4$ achieves 38.9% accuracy at 49.8 seconds per problem, compared to 27.5% at 50.0 seconds for ST-BoN and 33.7% at 52.3 seconds for Parallel-Probe. We report implementation details in Section B.9.

4.4 Sensitivity to Equivalence Checker Quality

Errors in answer grouping can distort fixed aggregation rules such as weighted voting. We test whether MSV can learn to use this imperfect agreement signal more robustly than fixed aggregation. On Omni-MATH, we replace SymPy with Jaccard similarity or embedding cosine similarity, while retaining SymPy for correctness labels. In Table 3, MSV_{64} loses only 0.3 and 0.4 percentage points of best-of-64 accuracy with the Jaccard and embedding checkers, respectively, compared with 0.5 and 2.0 points for weighted voting with Probe. Full results for five alternative checkers are in Section D.3. These results suggest that MSV does not require a ground-truth symbolic equivalence checker: when trained and evaluated with the same fixed checker, it can learn to use even imperfect agreement signals effectively.

4.5 Different Base Language Models

To test whether MSV's gains extend beyond our main base model, we repeat the **Terminal Answers** experiments with DeepSeek-R1-Distill-Llama-8B, Qwen3-1.7B in thinking mode, and Llama-3.2-1B-Instruct. For the first two, we train on DeepMath-103K and evaluate on AIME. For Llama-3.2-1B-Instruct, which has near-zero accuracy on AIME, we use the MATH train/test splits. MSV_{64} achieves lower Brier scores and higher best-of-64 accuracy than the strongest single-sequence verifier baseline for each metric across all three models (Table 4), although the accuracy gains vary by model. These results extend the evaluation to both more powerful models and non-reasoning models. Full results and experimental details are in Section D.4.

5 Conclusion

In this work, we addressed two critical bottlenecks in parallel test-time scaling: the accurate selection of correct solutions, and the high inference latency from generating multiple candidates. We argued that existing single-sequence verifiers are limited by scoring candidate solutions in isolation. To overcome this, we introduced the Multi-Sequence Verifier (MSV), a novel architecture designed to jointly process an entire set of candidate solutions and model their interactions. Our experiments demonstrated that MSV improves best-of-N selection accuracy over strong baselines. Ablations support the contributions of agreement features and equivalence attention, while additional experiments assess sensitivity to equivalence checkers and performance across different base models.

Furthermore, we studied a parallel early-stopping framework for efficient inference. In this framework, a streaming variant of MSV achieved the same peak accuracy as the best single-sequence verifier on MATH with less than half the average stopping token position. We also compared MSV against two parallel early-stopping baselines, ST-BoN and Parallel-Probe, achieving higher accuracy at similar estimated latency on AIME. These findings underscore the importance of cross-sequence information and end-to-end training, and establish a new, effective approach to building and using verifiers for parallel test-time scaling.

AI USE STATEMENT

We used LLMs to assist with implementing methods, developing the theoretical motivation, designing additional experiments, interpreting results, and drafting and revising portions of the manuscript. Small language models generated training data for MSV. The manuscript was written primarily by the authors, with AI assistance for language editing and LaTeX formatting. The authors reviewed the AI-assisted work for correctness and take responsibility for the final content.

ETHICS STATEMENT

This work does not involve human subjects or new dataset releases, and we do not anticipate direct risks to privacy, security, or fairness. Our contribution is a method for improving verifier calibration to enhance downstream parallel scaling in large language models, with no specific application domain targeted. Improved verification reliability could benefit society through more trustworthy, efficient decision-making, though as with any capability-enhancing AI technique, misuse is possible if applied toward harmful ends. We identify no concerns unique to this method beyond those already associated with general advances in LLM reasoning.

REPRODUCIBILITY STATEMENT

We include source code for the experiment pipeline in the supplementary materials. The verifier architecture, training objective, and inference procedures are described in Section 3, with further method details in Sections C.1 and C.2. Experimental settings, baselines, model and dataset information, and training details are provided in Section A. Additional experiments and detailed results are reported in Sections B and D; online timing components and offline optimization costs are distinguished in Section B.1.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*, 2023.

David Blackwell. Equivalent comparisons of experiments. *The annals of mathematical statistics*, pp. 265–272, 1953.

Jaroslaw Blasiok, Parikshit Gopalan, Lunjia Hu, and Preetum Nakkiran. When does optimizing a proper loss yield calibration? *Advances in Neural Information Processing Systems*, 36:72071–72095, 2023.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

Seyyed Saeid Cheshmi, Azal Ahmad Khan, Xinran Wang, Zirui Liu, and Ali Anwar. Accelerating llm reasoning via early rejection with partial reward modeling. In *EMNLP (Findings)*, pp. 10433–10447, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. Slim-sc: Thought pruning for efficient scaling with self-consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34488–34505, 2025.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 2025.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models, 2022. *URL https://arxiv.org/abs/2206.14858*, 1, 2022.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier. *arXiv preprint arXiv:2206.02336*, 2022.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*, 2024.

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*, volume 2024, pp. 39578–39601, 2024.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairjudge rm: Perform best-of-n sampling with knockout tournament. *arXiv preprint arXiv:2501.13007*, 2025.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention. *arXiv preprint arXiv:2504.06261*, 2025.

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

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *Advances in Neural Information Processing Systems*, 37:32630–32652, 2024.

Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. Genselect: A generative approach to best-of-n. *arXiv preprint arXiv:2507.17797*, 2025.

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 Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *Advances in Neural Information Processing Systems*, 38:162137–162174, 2026.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. Dynamic voting for efficient reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3085–3104, 2023.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025a.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2024. *URL https://arxiv.org/abs/2408.15240*, 1, 2024.

Xuteng Zhang, Wenhao Zeng, Xiaodong Gu, Chao Hu, Haotian Lin, Yuling Shi, Min Wang, and Beijun Shen. Paratempo: Efficient parallel reasoning via temporal confidence. *arXiv preprint arXiv:2608.16425*, 2026.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025b.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, et al. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*, 2026.

A Experimental Details

A.1 Baseline Methods

A.1.1 Single-Sequence Baselines

To assess the benefit of multiple candidates, we compare MSV trained on $N > 1$ against the same architecture trained on $N = 1$. We abbreviate MSV trained on N sequences simply as MSV_N. As another baseline, we follow Zhang et al. (2025a) and train a 2-layer MLP "Probe" on the last token representations $h^{(n)}_{k,L^{(n)}_k}$ to predict the correctness of answers $a^{(n)}_k$. We find that the Probe serves as a strong baseline that often matches or even exceeds the performance of MSV_1.

We also consider the open-source verifier Qwen2.5-Math-PRM-7B (PRM; Zhang et al., 2025b), which is a 7-billion-parameter process reward model trained to score intermediate reasoning steps. Intermediate reasoning traces with special tokens `<extra_0>` are fed to the PRM, which outputs correctness scores for the traces; we use the softmax probability of the positive class as the correctness score. Specifically, in both the **Terminal Answers** and **Streaming Answers** setting, we feed to the PRM the currently decoded reasoning trace appended with the terminal/intermediate answer to evaluate.

We focus on non-generative, white-box methods in our paper. Generative verifiers (Zhang et al., 2024) and joint-selection approaches such as USC (Chen et al., 2023), PairJudge RM (Liu et al., 2025), and GenSelect (Toshniwal et al., 2025) offer alternative approaches to verification and selection. In our **Terminal Answers** evaluation, our GenSelect implementation requires approximately 130 seconds to judge a group of 16 solutions, compared to 0.006 seconds for MSV_{16}. This measured overhead motivates lightweight verification for repeated intermediate-answer evaluation.

We also evaluate a training-free correctness score based on the base LLM's token probabilities on the answer tokens (Yang et al., 2025). Specifically, we use the geometric mean of probabilities over all tokens in an answer as its score; this construction does not itself ensure calibration.

$$s^{(n)}_k = \left(\prod_{i=1}^{L^{(n)}_k} p_{\text{LLM}} \left(a^{(n)}_{k,i} \mid \text{tokens}_{<a^{(n)}_{k,i}} \right) \right)^{1/L^{(n)}_k}.$$

A.1.2 Aggregation Baselines

We complement the single-sequence verifier baselines with *Weighted Voting* (WV), an aggregation rule based on weighted best-of-N (Li et al., 2022): for each equivalence class of symbolically identical answers, we aggregate its answers' correctness probabilities by summing them, and normalize the probabilities across equivalence classes. Performing best-of-N with the WV probabilities is equivalent to weighted best-of-N, which picks the equivalence class with the biggest aggregate probability. In the streaming setting, we can also perform weighted voting for candidate $a^{(n)}_k$ using all answers completed by $t = \tau^{(n)}_k + L^{(n)}_k$: scores are summed within equivalence classes and normalized across the available classes. With few available candidates, normalization can produce high scores despite limited evidence. For instance, weighted voting will always assign a probability of one to the first output candidate, since there are no other candidates to normalize over. Therefore, we set a threshold $R \in \mathbb{Z}$ such that weighted voting is performed only when the number of candidates exceeds R at completion time $t = \tau^{(n)}_k + L^{(n)}_k$. We find that $R = 16$ yields good overall performance and use this setting throughout our experiments with WV in the **Streaming Answers** setting.

We also evaluate the training-free baseline *self-consistency* (Wang et al., 2022) in the **Terminal Answers** setting that only uses the vote count (number of symbolically identical answers) to score each answer. Specifically, we score each answer $a^{(n)}$ with

$$s^{(n)} = \sum_{m=1}^{N} \mathbf{1}[a^{(n)} \sim a^{(m)}]/N.$$

A.2 Details on Datasets and Models

The pre-trained model and datasets used in our experiments are summarized below.

Models.

- **DeepSeek-R1-Distill-Qwen-1.5B**: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B, licensed under MIT License.[1]
- **DeepSeek-R1-Distill-Llama-8B**: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B, licensed under MIT License.[2]
- **Qwen3-1.7B**: https://huggingface.co/Qwen/Qwen3-1.7B, licensed under Apache License 2.0.[3] We use Qwen3-1.7B in thinking mode.
- **Llama-3.2-1B-Instruct**: https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct, licensed under the Llama 3.2 Community License.[4]
- **Qwen2.5-Math-PRM-7B**: https://huggingface.co/Qwen/Qwen2.5-Math-PRM-7B, licensed under the Qwen License Agreement.[5]

Datasets.

- **`MATH-500`**: https://huggingface.co/datasets/HuggingFaceH4/MATH-500, licensed under MIT License.[6]
- **`Olympiad Bench`**: https://huggingface.co/datasets/Hothan/OlympiadBench, licensed under MIT License.[7]
- **`AMC12`**: https://huggingface.co/datasets/rulins/amc12_22-24, not openly licensed; usage is subject to MAA AMC policies.[8]
- **`AIME`**: https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024, licensed under CC0 License. [9]
- **`Omni-MATH`**: https://huggingface.co/datasets/KbsdJames/Omni-MATH, dataset files licensed under Apache-2.0 License.[10]
- **`DeepMath-103K`**: https://huggingface.co/datasets/zwhe99/DeepMath-103K, licensed under MIT License.[11]

We use the DeepSeek-R1-Distill-Qwen-1.5B model (Guo et al., 2025) as our base LLM that generates the reasoning traces and answers to reasoning problems. For the main mathematical reasoning experiments, verifier training data are generated with this base LLM on problems in the DeepMath-103K dataset, which underwent careful decontamination of public reasoning evaluation datasets, including all five of our evaluation datasets (He et al., 2025). This decontamination reduces overlap between verifier training and evaluation problems. We evaluate our model on the MATH dataset (Hendrycks et al., 2021), the AMC12 problems between 2022 and 2024 that are not geometry problems, the AIME problems between 1983 and 2024, OlympiadBench (He et al., 2024), and Omni-MATH (Gao et al., 2024). We sample AIME evaluation questions from the 1983–2024 problem set to obtain a larger evaluation set than the 60 questions in AIME 2024/2025. DeepMath-103K was decontaminated against this full AIME problem set (He et al., 2025).

We use 224 and 64 randomly chosen problems from DeepMath-103K as our training and validation sets, respectively. Each evaluation dataset is randomly subsampled to 448 problems (except for AMC12, which has 134 problems). We generate 64 responses from the base LLM on every problem. This creates training and validation sets with 14K and 4K sequences, respectively, and evaluation datasets with approximately 28K sequences each (8,576 for AMC12). We examine training-set size in Section B.11; the separate ARC-to-MMLU protocol is described in Section B.13.

[1] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B/blob/main/LICENSE
[2] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B/blob/main/LICENSE
[3] https://huggingface.co/Qwen/Qwen3-1.7B/blob/main/LICENSE
[4] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct/blob/main/LICENSE.txt
[5] https://huggingface.co/Qwen/Qwen2.5-Math-PRM-7B/blob/main/LICENSE
[6] https://github.com/openai/prm800k/blob/main/LICENSE
[7] https://github.com/OpenBMB/OlympiadBench/blob/main/LICENSE
[8] https://maa.org/student-programs/amc/maa-american-mathematics-competitions-policies/
[9] https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024
[10] https://huggingface.co/datasets/KbsdJames/Omni-MATH
[11] https://github.com/zwhe99/DeepMath/blob/main/LICENSE

A.3 METRICS

We evaluate discrimination with AUROC, probabilistic prediction quality with Brier score and NLL, and calibration with ECE. Let $\{(p_i, y_i)\}_{i=1}^{m}$ denote the dataset of m predicted probabilities and their corresponding binary correctness labels.

Area Under the Receiver Operating Characteristic Curve (AUROC). The Area Under the Receiver Operating Characteristic Curve (AUROC) measures the probability that a randomly chosen positive example receives a higher score than a randomly chosen negative example:

$$\text{AUROC} = \Pr\left[p^+ > p^-\right] + \frac{1}{2}\Pr\left[p^+ = p^-\right],$$

where p^+ and p^- are the predicted probabilities for a random positive and negative sequence, respectively. AUROC is computed over the evaluated answers and need not reflect their ordering within each problem. We therefore measure best-of-N selection accuracy directly.

Brier score. The Brier score (Brier, 1950) is the mean squared error between predicted probabilities and binary outcomes:

$$\text{Brier} = \frac{1}{m}\sum_{i=1}^{m}(p_i - y_i)^2.$$

Lower values indicate better probabilistic prediction quality, but do not by themselves establish lower calibration error.

Negative Log-Likelihood (NLL). The negative log-likelihood, or log loss, evaluates the quality of probabilistic predictions:

$$\text{NLL} = -\frac{1}{m}\sum_{i=1}^{m}\Big(y_i \log p_i + (1 - y_i)\log(1 - p_i)\Big).$$

Expected Calibration Error (ECE). The expected calibration error (Naeini et al., 2015; Guo et al., 2017) measures the discrepancy between a model's predicted confidences and the actual accuracies. To compute ECE, predictions are grouped into J bins based on their confidence scores. The ECE is the weighted average of the absolute difference between the mean confidence and the mean accuracy within each bin:

$$\text{ECE} = \sum_{j=1}^{J}\frac{|B_j|}{m}\left|\text{acc}(B_j) - \text{conf}(B_j)\right|,$$

where B_j is the set of indices of predictions whose confidence p_i falls into the j-th bin, $\text{acc}(B_j) = \frac{1}{|B_j|}\sum_{i\in B_j} y_i$ is the accuracy of that bin, and $\text{conf}(B_j) = \frac{1}{|B_j|}\sum_{i\in B_j} p_i$ is the average confidence of that bin. We use $J = 10$ in all our reports of ECE.

Best-of-N Accuracy. In **Terminal Answers**, the verifier selects the highest-scoring answer $a^\dagger$ from N candidates, with score $p^\dagger$. Best-of-N accuracy is the fraction of evaluation problems for which the selected answer is correct:

$$\text{Accuracy} = \frac{1}{|Q|}\sum_{q\in Q} y_q^\dagger, \text{ where } y_q^\dagger = \mathbf{1}[a_q^\dagger \sim a_q^*]$$

where a_q^* is the ground-truth answer for problem q.

Best-of-N ECE and Brier score. Best-of-N ECE and Brier scores are the ECE and Brier scores computed over the set $\{(p_q^\dagger, y_q^\dagger)\}$ of best-of-N confidence and correctness pairs.

Area under the tradeoff curve. For the delimiter comparison, AUTC integrates accuracy on the vertical axis against token count on the horizontal axis. Let $A(c)$ denote the accuracy along the tradeoff curve at token count c, and let $c_{\lambda=1}$ denote the token count at stopping threshold $\lambda = 1$. We report the unnormalized area

$$\mathrm{AUTC} = \int_0^{c_{\lambda=1}} A(c)\, dc.$$

This token-based summary is distinct from the estimated wall-clock tradeoff. Its magnitude depends on both accuracy and the integration range.

A.4 Details on Experiments

We adopt the setup and terminology of **Terminal Answers** and **Streaming Answers** from Section 3, and follow the experimental protocol of Section 4.1. Below we provide additional details necessary for reproducibility.

Threshold sweeps and uncertainty. The early-stopping curves sweep the score threshold λ to describe the available accuracy–latency tradeoff. Comparisons read from these curves are descriptive operating points, rather than evidence of performance at a threshold selected on held-out validation data. Learning-rate and epoch selection, described below, are separate from stopping-threshold selection. For the main mathematical reasoning experiments, the $\pm$ values summarize standard deviations over five verifier initialization and training seeds, with generated candidates and evaluation problems held fixed. They do not provide paired confidence intervals for differences between methods over evaluation problems. Such intervals should account for candidates from the same problem sharing a ground truth and generation context. Similarly, comparisons between calibration metrics and downstream accuracy do not isolate the effect of calibration from ranking or threshold choice; a separate calibration control is needed for that attribution.

Optimizer and schedule. All models are trained with AdamW, combined with the Hugging Face default `constant-with-warmup` scheduler (default warmup ratio). Gradient clipping is applied with `max_grad_norm` set to 1.0.

Learning rate selection. We sweep learning rates in $\{1\mathrm{e}{-5}, 5\mathrm{e}{-5}, 1\mathrm{e}{-4}, 5\mathrm{e}{-4}, 1\mathrm{e}{-3}, 5\mathrm{e}{-3}\}$ and choose the value with the lowest Brier score on the validation split.

Final learning rates. The probe MLP is trained with lr $= 1\mathrm{e}{-3}$. All other modules use lr $= 5\mathrm{e}{-5}$, except the MMTB mixture weights w (lr $= 1\mathrm{e}{-1}$) and the per-sequence embeddings $e^{(n)}$ (lr $= 1\mathrm{e}{-3}$).

Model capacity. The MSV verifier consists of a single Multi-Mask Transformer Block (one transformer layer) whose width and attention heads follow the base model. The probe is implemented as a 2-layer MLP with hidden size 1024.

Batching and epochs. We use a global batch size of 64 with no gradient accumulation. Training runs for 1 epoch on Terminal data and 2 epochs on Streaming data.

Decoding and prompting. We follow the elicitation protocol in the main paper, using the boxed `Final Answer` format and the "`Wait`" delimiter for Streaming. Decoding uses temperature $= 1.0$ without further heuristics. Each output is allowed up to 4096 tokens, with the elicited final answer truncated to 40 tokens. Chain-of-thought generation uses PyTorch's scaled dot-product attention (SDPA) backend and KV caching via HuggingFace's `generate` function.

Hardware. All experiments fit within a single RTX A6000 GPU (48 GB memory) per model.

Category	Setting (ours)
Optimizer	AdamW
LR schedule	`constant-with-warmup` (HF defaults)
LR sweep	$\{1e{-}5, 5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3\}$ (validation Brier score)
Final LRs	Probe: $1e{-}3$; Others: $5e{-}5$; w: $1e{-}1$; $e^{(n)}$: $1e{-}3$
Probe arch.	2-layer MLP, hidden size 1024
MSV depth	1 Multi-Mask Transformer Block
Batch	64
Gradient accumulation steps	1
Epochs	Terminal: 1; Streaming: 2 (4 only during LR sweep)
Gradient clipping	`max_grad_norm` $= 1.0$
Gradient checkpointing	Not used
Max sequence length	4096
Temperature	1.0
Answer length	≤ 40 tokens (after elicitation)

Table 5: Summary of implementation and hyperparameters.

Table 6: Measured and estimated online latency components and their summed total on AIME without early stopping ($\lambda = 1$), in seconds. Verifier costs include simulated caching savings. The total includes symbolic equivalence checking; it is computed from components rather than independently timed end to end.

(a) CoT gen.

N	**Latency**
4	52.8
16	62.4
64	131.6

(b) Answer gen.

N	**Latency**
4	1.5
16	1.8
64	3.8

(c) Verifier inference

N	**Probe**	**MSV$_N$**	**PRM**
4	0.0008	0.4	7.4
16	0.003	2.3	29.6
64	0.01	23.2	118.5

(d) Estimated online latency

N	**Probe**	**MSV$_N$**	**PRM**
4	54.6	55.0	62.0
16	64.5	66.8	94.1
64	135.7	158.9	254.2

B Additional Experiments

B.1 Latency Analysis in the Streaming Answers Setting

We report online latency components on the AIME dataset in the **Streaming Answers** setting. Online latency covers a problem from the start of concurrent generation to returning an answer; offline data preparation and verifier training are reported separately. Specifically, we report component costs when we never early stop, i.e., set the threshold to be $\lambda = 1$. We separately tabulate chain-of-thought generation, intermediate answer generation, and verifier inference, and include symbolic equivalence checking in the total.

Estimated Wall-Clock Time. We combine generation timing from batched computation on a single A6000 GPU with estimated verifier and other component costs. Tables 6a, 6b, and 6c report the average latency for chain-of-thought generation, intermediate answer generation, and verifier inference, respectively. The verifier costs simulate the savings from the online KV-caching procedure in Section C.2; they are not timings of a complete cached streaming implementation.

Summed Online Latency. Table 6d reports the sum of the measured and estimated components plus the time taken for symbolic equivalence checks. This component sum is distinct from an independently measured end-to-end runtime, which would also capture scheduling and any overlap between operations. The symbolic-checking time is included in the total but is not separately tabulated. At $\lambda = 1$, MSV$_N$ has higher total latency than Probe because of its verification overhead; this table alone does not establish an early-stopping speedup. The threshold-sweep curves in Figure 9 compare accuracy and latency when early stopping is enabled. Token-position curves measure decoding progress and do not include answer-elicitation, verification, or equivalence-checking time, so they should not be interpreted directly as wall-clock savings. PRM incurs greater estimated verification overhead, as it requires a full forward pass through a 7-billion-parameter model for each intermediate answer. MSV instead applies a single transformer block to the existing LLM hidden states.

Offline preparation and optimization. In the **Terminal Answers** setting, optimization of the trained verifiers takes around 50 minutes. In the **Streaming Answers** setting, optimization of Probe, MSV$_1$, MSV$_4$, MSV$_{16}$, and MSV$_{64}$ takes around 105, 105, 105, 110, and 145 minutes, respectively. These are verifier optimization times, not a complete accounting of offline preparation: response generation, answer elicitation, feature extraction, correctness labeling, and hyperparameter search are not separately timed here. We selected the number of epochs by cross-validation; additional epochs did not improve the validation performance of Probe, MSV$_1$, or MSV$_4$. For Q deployment queries, an amortized time accounting is $L_{\text{online}} + T_{\text{offline}}/Q$, where T_{offline} includes both preparation and optimization. The reported optimization times alone do not determine this amortized cost.

B.2 Ablation Study of Attention Masks

To assess the contribution of each attention mask in MSV, we conduct an ablation study by removing each mask individually and evaluating the resulting model's discrimination and probabilistic prediction quality. Table 7 reports the Brier score, AUROC, and negative log-likelihood (NLL) of the Streaming MSV$_{16}$ model on AIME, with each mask ablated in turn.

Table 7: Mask ablation results on AIME with MSV_{16} in the **Streaming Answers** setting.

Metric	MSV_{16}	w/o full	w/o equiv.	w/o within-seq	w/o within-ans
Brier ↓	**0.038**	0.040	0.045	0.048	0.050
AUROC ↑	**0.953**	**0.953**	0.922	0.945	0.949
NLL ↓	**0.150**	0.157	0.236	0.192	0.169

Table 8: Effect of adding causal logit averaging to MSV_{16} in the **Streaming Answers** setting. We report AUROC, Brier score, and NLL on the AIME dataset.

Method	AUROC ↑	Brier ↓	NLL ↓
MSV_1	0.906±0.008	0.064±0.003	0.241±0.011
MSV_{16}	**0.953**±0.007	**0.038**±0.003	**0.150**±0.011
MSV_{16} + logit averaging	0.936±0.0016	0.047±0.009	0.190±0.0333

The results show that all masks contribute to the overall performance, but the equivalence and within-sequence masks are especially important. Removing the equivalence mask increases the Brier score by 18.4% and substantially degrades AUROC and NLL, supporting the use of cross-sequence attention to equivalent answers in this setting. Removing the within-sequence mask also leads to noticeable degradation across all metrics. In contrast, removing the full mask has the smallest effect, suggesting that the more specialized masks capture the most relevant information for verification.

Motivation for Multiple Masks. Our decision to use multiple specialized masks was informed by preliminary experiments in which we allowed the verifier to attend to all tokens in the sequence, including chain-of-thought tokens. We initially hypothesized that, since attending to more tokens strictly increases the available information, specialized masks would be unnecessary.

Contrary to this expectation, we observed that restricting attention to only the answer tokens led to substantially better generalization. This observation motivated restricting the verifier to answer representations; it does not identify why unrestricted attention performed worse. Building on this insight, we tested more specialized masks that attend only to semantically related subsets of answers (e.g., equivalent answers within and across sequences). The reported ablation favors the full combination of masks over removing any one mask.

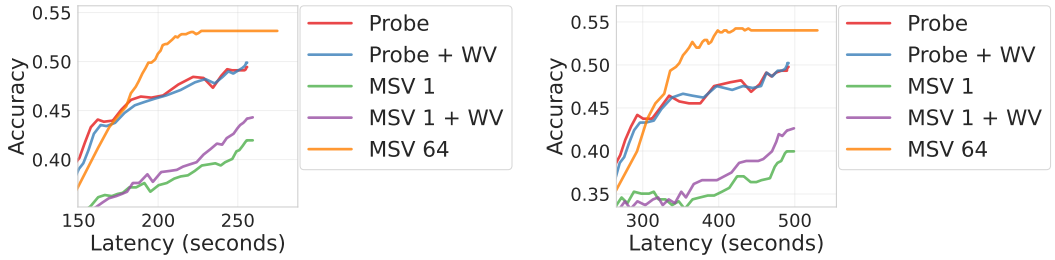
B.3 Agreement Features and Equivalence Attention

Table 1 gives the 2×2 ablation of γ and the equivalence mask on AIME in the **Terminal Answers** setting. All variants are trained from scratch with identical hyperparameters and evaluated over five seeds. This differs from the **Streaming Answers** mask ablation in Section B.2, which removes individual attention masks and measures correctness-prediction metrics. Here the outcome is best-of-16 selection accuracy. Neither experiment by itself separates the effects of ranking and calibration on selection.

Both components improve mean accuracy when the other is absent, and their combination yields the highest mean. The mask contributes 6.24 percentage points without γ and 0.65 points with it; γ contributes 8.33 points without the mask and 2.74 points with it. The equivalence mask retains agreement information even when γ is removed. In addition, **Terminal Answers** uses equivalence-based output aggregation (Equation (1)), so removing these two components should not be interpreted as removing all uses of answer equivalence. A comparison against a trained Probe+γ baseline with controlled output aggregation is needed to isolate gains over that alternative.

B.4 Ablation: Logit Averaging in Streaming Answers

To test the choice to omit logit averaging in **Streaming Answers** (Section 3.2), we train a variant that uses it during both training and inference. Since the predictions have to respect causality in the **Streaming Answers** setting, we average the logits over all symbolically equivalent answers that come before in time. We train the ablated model architecture with $N = 16$, and evaluate on the



(a) $N = 128$ **(b)** $N = 256$

Figure 6: Accuracy versus estimated latency curves in **Streaming Answers** on AIME. We use the grouping strategy with MSV_{64} for large N.

AIME dataset. We report AUROC, Brier score, and NLL in Table 8. Logit averaging worsens all three metrics in this comparison. One possible explanation is that averaging retains less informative early-answer logits in later predictions, but this mechanism is not isolated by the experiment.

B.5 Computational Complexity of MSV

In the **Streaming Answers** setting, N sequences are decoded in parallel. Assuming delimiters are uniformly distributed throughout a sequence, the total number of delimiters across all sequences—and thus the total number of forward passes through MSV—is $O(NL)$, where L is the average sequence length. For this accounting, hold the model width and maximum elicited-answer length fixed. With KV caching, a new answer containing ℓ tokens requires $O(\ell T_{\text{acc}})$ attention work against T_{acc} accumulated answer tokens (Section C.2). Since ℓ is bounded and $T_{\text{acc}} = O(NL)$ under these assumptions, each update costs $O(NL)$ while querying all tokens in the new answer. This yields a total MSV computational cost of $O(N^2L^2)$, whereas the decoding of the N sequences itself scales as $O(NL^2)$. Thus, MSV adds an extra factor in N, while the dependence on sequence length L matches that of the underlying decoding process. This is a common issue among methods that incorporate information between sequences, but is usually shown not to be a significant bottleneck for practical values of N (Du et al., 2023; Rodionov et al., 2025). We evaluate fixed-size grouping in Section B.6.

B.6 Fixed-Size Grouping for Large Candidate Sets

To address scalability concerns when N is large (e.g., on the order of hundreds), we use MSV_M with N parallel sequences, where M is kept constant as we scale N. This is achieved by partitioning the N sequences into N/M groups, each of size M, and processing each group through MSV_M independently. This reduces the computational complexity from $O(N^2L^2)$ to $O(MNL^2)$, or equivalently $O(NL^2)$ when treating M as a constant. We also note that grouping sequences to reduce computational load is a common strategy in the generative verifier literature, for instance using pairwise knockout rounds (Liu et al., 2025), or tournaments (Toshniwal et al., 2025). We therefore do not claim this to be a novel contribution of our paper.

We provide the full accuracy-latency tradeoff curves for $N = 128$ and $N = 256$ in Figure 6, which show that MSV_{64} (grouped) has a better estimated tradeoff than the single-sequence verifier baselines across both scales. These estimates are consistent with the intended cost reduction from fixed-size grouping; they do not independently establish end-to-end scaling behavior on deployed hardware.

B.7 Direct Transfer from 64 to 16 Candidates

We copy trained MSV_{64} parameters into an $N = 16$ instance, retaining the first 16 learned sequence embeddings, and evaluate on groups of 16 candidates without retraining. Table 9 compares this checkpoint with an MSV_{16} trained at the evaluation group size. Differences in means range from -0.46 to $+0.36$ percentage points for transferred minus separately trained models. These small observed differences are evidence of practical downward transfer in this setting, without constituting a statistical equivalence test.

Table 9: Best-of-16 accuracy for separately trained MSV_{16} and an MSV_{64} checkpoint applied directly to 16 candidates without retraining.

Dataset	Trained at $N = 16$	Trained at $N = 64$
MATH	0.7472±0.0009	0.7463±0.0021
OlympiadBench	0.5154±0.0037	0.5144±0.0056
AMC12	0.5399±0.0061	0.5403±0.0028
AIME	0.4615±0.0046	0.4569±0.0091
Omni-MATH	0.3198±0.0032	0.3234±0.0038

Table 10: Unnormalized accuracy–token-count AUTC with different delimiters on the AIME dataset with $N = 16$ parallel sequences. "Every 200 tokens" is a delimiter-free baseline that extracts answers at fixed token intervals.

Delimiter	Probe	MSV_1	MSV_{16}
"Wait"	1024.2	954.8	**1078.3**
"Alternatively"	985.9	980.0	**1012.4**
"But"	1039.3	1023.6	**1070.7**
Every 200 tokens	910.9	—	**1013.9**

The larger-budget experiments in Section B.6 use a different construction: candidates are partitioned into groups, and the verifier attends within each group. In particular, using two size-64 groups for 128 candidates retains the training group size rather than testing direct $64 \to 128$ transfer. The $64 \to 16$ results concern terminal answer selection and do not establish streaming early-stopping performance under a group-size mismatch.

B.8 Sensitivity to Delimiter Choice in Streaming Answers

In the **Streaming Answers** setting, we use delimiter tokens to segment sequences into intermediate answers. While our main experiments use "Wait" as the delimiter, we investigate the sensitivity of MSV to this choice by evaluating two alternative delimiters: "Alternatively" and "But", both of which frequently appear as the first token of a paragraph in mathematical reasoning. On the AIME dataset, "Wait", "Alternatively," and "But" occur roughly once every 200, 900 and 400 tokens, respectively. We also evaluate a delimiter-free approach that extracts an intermediate answer every 200 tokens, regardless of token content, to test whether gains persist without a lexical delimiter.

Table 10 reports the unnormalized area under the accuracy–token-count curve (AUTC) for each delimiter-verifier combination, integrated from zero to the token count at $\lambda = 1$. MSV_{16} has higher AUTC than Probe in all four tested settings, including extraction every 200 tokens. Absolute AUTC varies with the extraction rule, so this experiment supports sensitivity analysis over delimiter choices rather than invariance to them. The delimiter can be selected as a hyperparameter on a validation set.

B.9 Comparisons for Parallel Early Stopping

Table 2 summarizes answer-selection accuracy and estimated latency for the evaluated configurations. All methods use 16 candidates. The methods retain their respective probing schedules.

B.9.1 ST-BoN

We compare the official ST-BoN implementation (Wang et al., 2026) with MSV on the same 448 AIME questions using DeepSeek-R1-Distill-Qwen-1.5B, $N = 16$, and identical generation settings. ST-BoN uses $\alpha = 1$, corresponding to $\tau = c$, with mean $c = 26.02$. Both methods use batched decoding on one A6000 GPU. Latency is measured end-to-end on the 448 questions and averaged.

At approximately 50 seconds per problem, MSV_{16} achieves 38.9% accuracy versus 27.5% for ST-BoN, an improvement of 11.4 percentage points. With the maximum threshold, MSV reaches 45.6% at 66.8 seconds. The medium threshold was selected from the evaluation accuracy–latency curve.

Table 11: Best-of-16 accuracy comparison between GenSelect and MSV_{16} on five mathematical reasoning benchmarks in the **Terminal Answers** setting.

Dataset	GenSelect	MSV_{16}
MATH	0.724	**0.747**
OlympiadBench	0.500	**0.515**
AMC12	0.455	**0.540**
AIME	0.356	**0.462**
Omni-MATH	0.296	**0.320**

B.9.2 PARALLEL-PROBE

We evaluate Parallel-Probe (Zheng et al., 2026) on 448 AIME questions using DeepSeek-R1-Distill-Qwen-1.5B, with 16 candidate trajectories per question and a 4,096-token reasoning budget. The $N = 16$ experiment uses the first 16 trajectories for each question, with newly generated answer probes. We simulate its stopping and pruning rules on pregenerated trajectories sampled at temperature 1.0.

We elicit intermediate answers every 500 reasoning tokens and at the 4,096-token endpoint, using the same answer-elicitation suffix as our main experiments for a consistent comparison. Probes use greedy decoding with at most 40 output tokens. Following Parallel-Probe's stopping and pruning criteria, the controller stops when the majority answer remains unchanged for u checkpoints and prunes branches that disagree with the majority for k checkpoints, after a warmup of W checkpoints.

We sweep $W \in \{0, 2, 4, 6\}$ and $u, k \in \{2, 3, 4\}$, reporting $(W, u, k) = (6, 4, 4)$, which achieves the highest accuracy among the 36 configurations on the evaluation questions. This configuration achieves 33.71% answer-selection accuracy (Table 2). Its answer-probing protocol differs from the main MSV evaluation, including terminal probes for budget-truncated trajectories.

We estimate latency by replaying the stopping and pruning decisions and summing the corresponding prompt-processing, reasoning, and answer-probing costs using timing measurements from a single A6000 GPU. The estimate assumes that finished and pruned sequences are removed from the decoding batch and cached computation is reused across probes. This gives an average of 52.3 seconds per problem.

B.10 COMPARISON WITH GENSELECT

We additionally compare against GenSelect (Toshniwal et al., 2025), a generative verifier that prompts an LLM judge to directly select the best solution from a group of N candidates. Since our base model is DeepSeek-R1-Distill-Qwen-1.5B, we use the same model as the summarizer and judge. Table 11 reports best-of-16 accuracy on five datasets, comparing GenSelect against MSV_{16}.

MSV_{16} achieves higher accuracy than our GenSelect implementation on all five datasets. Directly measured inference time for selecting from 16 solutions averages approximately 130 seconds for GenSelect and 0.006 seconds for MSV_{16}, excluding candidate generation. These measurements concern a single terminal selection; streaming costs include repeated intermediate-answer evaluations (Section B.1).

B.11 TRAINING SET SIZE

Our training set consists of 224 problems sampled from DeepMath-103K, with 64 model responses per problem, yielding $224 \times 64 = 14{,}336$ sequences in total. We varied the number of training problems in preliminary **Terminal Answers** experiments with Probe. In Table 12, increasing the sample from 224 to 448 problems does not lower the AIME Brier score. This supports the selected size for this Probe comparison; it does not establish a sample-complexity result for MSV or for other datasets. This is also consistent with Zhang et al. (2025a) who report that a training set of 1,000 sequences was enough for Probe to generalize.

Table 12: Brier score of Probe on the AIME dataset as a function of training set size in the **Terminal Answers** setting.

# of problems	Brier score
56	0.0833
112	0.0821
224	0.0809
448	0.0812

Table 13: Brier score comparison between MSV_N using the last token versus the mean of answer tokens on the AIME dataset in the **Terminal Answers** setting.

Method	$N = 1$	$N = 4$	$N = 16$	$N = 64$
MSV_N	0.0819	0.0687	0.0572	0.0387
MSV_N (Mean)	0.0756	0.0641	0.0544	0.0382

B.12 MSV WITH MEAN ACROSS ANSWER TOKENS

In our main experiments, MSV obtains each logit from the transformer block output at the last answer-token position. This design choice was inherited from prior work on probes (Zhang et al., 2025a), which operate exclusively on the final hidden state. We hypothesized that the attention layer would be sufficiently expressive to aggregate relevant information from all answer tokens.

However, an alternative design is to use the mean of hidden states across all answer tokens instead of only the last token. To evaluate this variant, we compare MSV_N against MSV_N (Mean) in the **Terminal Answers** setting on the AIME dataset.

Table 13 shows that the mean variant provides a modest but consistent improvement in Brier score across all values of N. Both variants compute attention outputs for all tokens in the new answer. They differ in the readout: the standard variant uses the last-token output, while the mean variant averages outputs across answer tokens. Thus, this ablation compares readout choices and does not establish a reduction in attention queries for the standard variant.

B.13 ARC-CHALLENGE TRAINING AND MMLU EVALUATION

The verifiers in Table 14 use DeepSeek-R1-Distill-Qwen-1.5B. They are trained on 224 questions from the ARC-Challenge training split (Clark et al., 2018) and evaluated on 448 questions from the MMLU test split (Hendrycks et al., 2020), using the pooled `all` configuration. Questions retain their multiple-choice options, and target answers are choice labels. The preprocessing samples from the pooled test questions without stratifying by subject; it does not ensure coverage of every subject. We use precomputed terminal-answer representations and carry over the learning rates and one-epoch training schedule from the main **Terminal Answers** experiments, without selecting them on MMLU. We evaluate on MMLU after training; no separate ARC validation split is used for this experiment. The reported verifier results use the first five training seeds in the script, 10–14, with the precomputed candidates and evaluation questions held fixed. We report means and standard deviations across these five seeds. Learned verifiers select among 16 candidates per question; pass@1 is the single-answer reference. MSV_{16} improves mean selection accuracy by 5.0 percentage points over MSV_1. The evaluation supports cross-dataset transfer after training outside competition mathematics. It does not establish performance for each MMLU subject separately, zero-shot transfer from DeepMath training, or comparisons against self-consistency and weighted voting.

Table 14: MMLU accuracy after verifier training on ARC-Challenge. Verifier rows report best-of-16 mean accuracy ± standard deviation over five training seeds; pass@1 is a single-answer reference.

Method	Accuracy
Pass@1	0.478
Probe	0.485±0.005
MSV_1	0.490±0.009
MSV_{16}	0.540±0.005

Algorithm 1 Multi-Sequence Verifier (MSV) Forward Pass

Require: N sequences with hidden states $\{h_{k,i}^{(n)}\}$, readout time t

Ensure: Correctness predictions $\{\tilde{y}_k^{(n)}\}$ for all answers up to time t

Trainable parameters: Sequence embeddings $\{\mathsf{e}^{(n)}\}$, QKV and output projections (W_Q, W_K, W_V, W_O), mask weights $\{\mathbf{w}_h\}$, MLP & LN, feature MLP, prediction head $(\mathbf{w}, b)$

$U^{(t)} \leftarrow \text{concat}\left(\left\{\left\{\{h_{k,i}^{(n)} + \mathsf{e}^{(n)}\}_{i=1}^{L_k^{(n)}}\right\}_{\tau_k^{(n)}+L_k^{(n)}\le t}\right\}_{n=1}^{N}\right)$

Compute query, key, value projections: $(Q_h^{(t)}, K_h^{(t)}, V_h^{(t)})_{h=1}^{H}$

for each head $h = 1, \ldots, H$ and mask $j = 1, \ldots, J$ **do**

  $A_{h,j}^{(t)} \leftarrow \text{softmax}\left(\frac{Q_h^{(t)}(K_h^{(t)})^\top}{\sqrt{d_{\text{head}}}} + \log M_j\right) V_h^{(t)}$

end for

$\tilde{U}^{(t)} \leftarrow \text{concat}_{h=1}^{H}\left(\sum_{j=1}^{J} \alpha_{h,j} A_{h,j}^{(t)}\right) W_O \quad$ where $(\alpha_{h,1}, \ldots, \alpha_{h,J}) = \text{softmax}(\mathbf{w}_h)$

$Z^{(t)} \leftarrow (U^{(t)} + \tilde{U}^{(t)}) + \text{MLP}(\text{LN}(U^{(t)} + \tilde{U}^{(t)}))$

for each answer $a_k^{(n)}$ generated by time t **do**

  if Terminal Answers then

    $S_k^{(n)} \leftarrow \{1, \ldots, N\}$

    for $m \in S_k^{(n)}$ **do**

      $j_m \leftarrow 1$

    end for

  else if Streaming Answers then

    $S_k^{(n)} \leftarrow \{m : \tau_1^{(m)} + L_1^{(m)} \le \tau_k^{(n)} + L_k^{(n)}\}$

    for $m \in S_k^{(n)}$ **do**

      $j_m \leftarrow \max\{j : \tau_j^{(m)} + L_j^{(m)} \le \tau_k^{(n)} + L_k^{(n)}\}$

    end for

  end if

  $\gamma_k^{(n)} \leftarrow \frac{1}{|S_k^{(n)}|} \sum_{m \in S_k^{(n)}} \mathbf{1}[a_{j_m}^{(m)} \sim a_k^{(n)}]$

  $\bar{z}_k^{(n)} \leftarrow z_{k,L_k^{(n)}}^{(n)} + \text{MLP}(\gamma_k^{(n)})$

end for

if Terminal Answers then

  $\tilde{y}_1^{(n)} \leftarrow \sigma\left(\frac{1}{|\mathcal{C}(n)|}\sum_{m\in\mathcal{C}(n)}(\mathbf{w}^\top \bar{z}_1^{(m)} + b)\right)$ for all n

else if Streaming Answers then

  $\tilde{y}_k^{(n)} \leftarrow \sigma(\mathbf{w}^\top \bar{z}_k^{(n)} + b)$ for all (n, k)

end if

return $\{\tilde{y}_k^{(n)}\}$

C METHOD DETAILS

C.1 MULTI-SEQUENCE VERIFIER PSEUDOCODE

Algorithm 1 gives MSV's *offline* forward pass, which computes $Z^{(t)}$ from all answer tokens accumulated at readout time t. Below, we describe an *online* procedure that reuses previously computed states.

C.2 ONLINE INFERENCE WITH KV CACHING

Algorithm 1 processes all answers accumulated up to time t in a single forward pass, which would require recomputing the full attention outputs from scratch each time a new answer arrives. The online procedure exploits the causal structure of the attention masks in the **Streaming Answers** setting to avoid redundant computation via key-value (KV) caching, analogously to how autoregressive language model inference reuses cached states from prior tokens.

Because the causal constraint in the **Streaming Answers** setting means that an answer $a_k^{(n)}$ may only attend to answers $a_{k'}^{(m)}$ completed no later than itself, with $\tau_{k'}^{(m)} + L_{k'}^{(m)} \leq \tau_k^{(n)} + L_k^{(n)}$, the keys and values corresponding to all previously generated answers remain unchanged when a strictly later-completing answer arrives. Their attention outputs therefore do not need to be recomputed. The non-strict inequality also permits attention between answers that complete at the same time. To reproduce this mask online, all answers sharing a completion time must be processed together, with their keys and values available before computing their attention outputs.

Online procedure. When a new intermediate or terminal answer $a_k^{(n)}$ (with $L_k^{(n)}$ tokens) finishes decoding at time $\tau_k^{(n)} + L_k^{(n)}$, the online algorithm proceeds as follows. If several answers complete simultaneously, the query and cache updates below apply to their combined token set.

1. **Query only the new tokens.** Compute query vectors for the $L_k^{(n)}$ new tokens only, while reading key and value vectors for *all* tokens (new and cached) to form the attention output for the new tokens. KV vectors for the new tokens are appended to the cache.
2. **Extract the last-token output.** Since only the last token's representation $z_{k,L_k^{(n)}}^{(n)}$ is needed for prediction (its attention output can incorporate the other tokens within the answer), we pass only this representation to the MLP and prediction head.
3. **Update $\gamma_k^{(n)}$ and predict.** Compute $\gamma_k^{(n)}$ using the causal definition (Section 3), augment the last-token representation, and apply the prediction head and sigmoid to output the correctness score $\tilde{y}_k^{(n)}$.

Complexity. Each per-answer forward pass in the online algorithm attends $L_k^{(n)}$ query tokens against a key-value cache of size T_{acc} (the total number of answer tokens accumulated so far), giving an attention cost of $O(L_k^{(n)} \cdot T_{\text{acc}})$ per new answer, compared to $O(T_{\text{acc}}^2)$ for a full recomputation. These attention costs suppress fixed model-width factors. The MLP and feature augmentation operate on a single token per answer, so their cost does not grow with the cache length.

Equivalence to the offline algorithm. With answers grouped by completion time as above, the online and offline algorithms compute the same masked attention and prediction functions, up to floating-point differences. Each query can attend only to answers completed no later than its own answer. Processing these groups in completion order therefore preserves the available keys and values. The agreement feature must likewise use the latest answers available at that completion time, rather than the final rollout. We have verified this numerically in our implementation. The latency estimates reported in Section B.1 simulate the cost savings of this online algorithm rather than timing its complete execution; the accuracy experiments use the offline algorithm (which is simpler to implement for batch evaluation over pre-collected sequences).

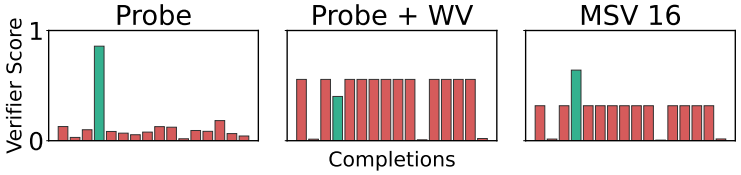


Figure 7: Probability estimates from Probe, Probe+WV, and MSV_{16} for a single problem in Omni-MATH, with 16 completions. Green represents the correct solution.

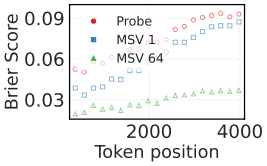


Figure 8: Brier score per token position, on AIME, **Streaming Answers**.

Table 15: AUROC (↑) and Brier score (↓) of verifiers in **Terminal Answers**.

Method	MATH		OlympiadBench		AMC12		AIME		Omni-MATH	
	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓
Token Probs	0.5320±0.0000	0.2504±0.0000	0.7704±0.0000	0.3696±0.0000	0.7350±0.0000	0.3540±0.0000	0.8999±0.0000	0.3073±0.0000	0.7814±0.0000	0.4467±0.0000
Probe	0.8790±0.0052	0.1346±0.0107	0.9022±0.0038	0.1271±0.0144	0.8994±0.0062	0.1438±0.0214	0.9461±0.0034	0.0859±0.0147	0.8836±0.0008	0.1340±0.0252
MSV 1	0.8566±0.0166	0.1460±0.0063	0.8926±0.0048	0.1310±0.0061	0.8890±0.0080	0.1498±0.0099	0.9469±0.0025	0.0819±0.0066	0.8711±0.0062	0.1460±0.0103
PRM	0.8156±0.0000	0.1747±0.0000	0.8958±0.0000	0.1689±0.0000	0.8521±0.0000	0.1807±0.0000	0.9374±0.0000	0.1472±0.0000	0.8826±0.0000	0.1584±0.0000
Token Probs + WV_4	0.7922±0.0000	0.1678±0.0000	0.8392±0.0000	0.2166±0.0000	0.7978±0.0000	0.2448±0.0000	0.8223±0.0000	0.2239±0.0000	0.8085±0.0000	0.2620±0.0000
Self-consistency$_4$	0.7876±0.0000	0.1686±0.0000	0.8290±0.0000	0.2180±0.0000	0.7893±0.0000	0.2477±0.0000	0.8001±0.0000	0.2274±0.0000	0.7986±0.0000	0.2627±0.0000
Probe + WV_4	0.7930±0.0033	0.1649±0.0008	0.8494±0.0018	0.2164±0.0017	0.8222±0.0040	0.2367±0.0010	0.8624±0.0037	0.2100±0.0017	0.8251±0.0025	0.2704±0.0028
MSV 1 + WV_4	0.7920±0.0023	0.1656±0.0005	0.8479±0.0009	0.2168±0.0013	0.8204±0.0026	0.2371±0.0014	0.8636±0.0012	0.2096±0.0008	0.8219±0.0012	0.2711±0.0021
PRM + WV_4	0.7942±0.0000	0.1653±0.0000	0.8491±0.0000	0.2126±0.0000	0.8136±0.0000	0.2403±0.0000	0.8444±0.0000	0.2171±0.0000	0.8200±0.0000	0.2602±0.0000
MSV 4	**0.8945**±0.0135	**0.1348**±0.0055	**0.9298**±0.0036	**0.1108**±0.0064	**0.9125**±0.0127	**0.1406**±0.0123	**0.9628**±0.0040	**0.0687**±0.0132	**0.8986**±0.0025	**0.1296**±0.0128
Token Probs + WV_{16}	0.8155±0.0000	0.1474±0.0000	0.8801±0.0000	0.1690±0.0000	0.8364±0.0000	0.1961±0.0000	0.8531±0.0000	0.1662±0.0000	0.8457±0.0000	0.1968±0.0000
Self-consistency$_{16}$	0.8149±0.0000	0.1482±0.0000	0.8751±0.0000	0.1704±0.0000	0.8299±0.0000	0.1991±0.0000	0.8422±0.0000	0.1696±0.0000	0.8398±0.0000	0.1976±0.0000
Probe + WV_{16}	0.8228±0.0018	0.1416±0.0007	0.9028±0.0022	0.1588±0.0010	0.8728±0.0030	0.1759±0.0025	0.9092±0.0055	0.1368±0.0039	0.8714±0.0026	0.1921±0.0006
MSV 1 + WV_{16}	0.8228±0.0033	0.1425±0.0003	0.9008±0.0010	0.1612±0.0008	0.8691±0.0029	0.1786±0.0018	0.9057±0.0024	0.1404±0.0015	0.8672±0.0010	0.1946±0.0008
PRM + WV_{16}	0.8149±0.0000	0.1457±0.0000	0.8888±0.0000	0.1647±0.0000	0.8466±0.0000	0.1922±0.0000	0.8693±0.0000	0.1588±0.0000	0.8524±0.0000	0.1953±0.0000
MSV 16	**0.9062**±0.0182	**0.1255**±0.0038	**0.9469**±0.0021	**0.1005**±0.0064	**0.9231**±0.0144	**0.1315**±0.0095	**0.9732**±0.0015	**0.0572**±0.0114	**0.9109**±0.0034	**0.1258**±0.0113
Token Probs + WV_{64}	0.8304±0.0000	0.1432±0.0000	0.8901±0.0000	0.1591±0.0000	0.8454±0.0000	0.1871±0.0000	0.8608±0.0000	0.1549±0.0000	0.8531±0.0000	0.1841±0.0000
Self-consistency$_{64}$	0.8273±0.0000	0.1440±0.0000	0.8875±0.0000	0.1605±0.0000	0.8410±0.0000	0.1900±0.0000	0.8517±0.0000	0.1583±0.0000	0.8505±0.0000	0.1849±0.0000
Probe + WV_{64}	0.8346±0.0030	0.1370±0.0009	0.9173±0.0026	0.1466±0.0014	0.8844±0.0050	0.1649±0.0033	0.9216±0.0066	0.1217±0.0050	0.8830±0.0029	0.1764±0.0012
MSV 1 + WV_{64}	0.8364±0.0019	0.1380±0.0005	0.9130±0.0014	0.1499±0.0009	0.8829±0.0034	0.1682±0.0023	0.9168±0.0031	0.1264±0.0018	0.8788±0.0012	0.1794±0.0008
PRM + WV_{64}	0.8274±0.0000	0.1418±0.0000	0.8997±0.0000	0.1546±0.0000	0.8576±0.0000	0.1834±0.0000	0.8784±0.0000	0.1473±0.0000	0.8613±0.0000	0.1827±0.0000
MSV 64	**0.9049**±0.0162	**0.1261**±0.0036	**0.9546**±0.0026	**0.0835**±0.0074	**0.9290**±0.0089	**0.1267**±0.0039	**0.9822**±0.0003	**0.0387**±0.0066	**0.9286**±0.0034	**0.1041**±0.0085

D Full Report of Main Experiments

This section reports correctness-prediction metrics, answer-selection accuracy, and early-stopping tradeoffs for all main benchmarks, followed by the equivalence-checker and base-model experiments.

D.1 Terminal Answers

Table 15 and Table 16 report AUROC, Brier score, ECE, and NLL in the **Terminal Answers** setting, including training-free baselines (Token Probs, Self-consistency). We find that MSV_N generally improves over the baselines with increasing N, following the trend reported in Section 4.2. Table 17 reports best-of-N accuracy. At $N = 4$, the strongest single-sequence verifier baseline has higher mean accuracy than MSV_4 on each dataset. MSV_{16} has the highest mean accuracy on four of five datasets, with PRM ahead on Omni-MATH; MSV_{64} leads on all five. Table 18 evaluates confidence in the selected answer using ECE and Brier score, measuring calibration error and overall probabilistic prediction quality, respectively.

The failure of WV is illustrated by an example in Figure 7: Probe initially assigns the highest score to the single correct solution, but WV overly suppresses it due to its low voting count of one.

Table 16: ECE (↓) and NLL (↓) of verifiers in **Terminal Answers**.

Method	MATH		OlympiadBench		AMC12		AIME		Omni-MATH	
	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓
Token Probs	0.1855±0.0000	0.9190±0.0000	0.4124±0.0000	1.1444±0.0000	0.3935±0.0000	1.1384±0.0000	0.4198±0.0000	0.8669±0.0000	0.5389±0.0000	1.2505±0.0000
Probe	0.0726±0.0401	0.4391±0.0485	0.0700±0.0517	0.4172±0.0437	0.1064±0.0581	0.4680±0.0730	0.1024±0.0483	0.3115±0.0488	0.1206±0.0589	0.4394±0.0817
MSV 1	0.0849±0.0190	0.4787±0.0291	0.0806±0.0232	0.4363±0.0213	0.1279±0.0304	0.4913±0.0364	0.1043±0.0250	0.2993±0.0221	0.1446±0.0288	0.4881±0.0345
PRM	0.1817±0.0000	0.5349±0.0000	0.1834±0.0000	0.5231±0.0000	0.1474±0.0000	0.5484±0.0000	0.2468±0.0000	0.4773±0.0000	0.2160±0.0000	0.5010±0.0000
Token Probs + WV_4	0.1615±0.0000	1.1805±0.0000	0.2416±0.0000	1.2018±0.0000	0.2616±0.0000	1.4551±0.0000	0.2677±0.0000	1.1803±0.0000	0.3306±0.0000	1.4633±0.0000
Self-consistency$_4$	0.1611±0.0000	1.1823±0.0000	0.2408±0.0000	1.2054±0.0000	0.2601±0.0000	1.4620±0.0000	0.2664±0.0000	1.1889±0.0000	0.3300±0.0000	1.4647±0.0000
Probe + WV_4	0.1655±0.0005	1.1793±0.0029	0.2457±0.0007	1.2096±0.0066	0.2633±0.0008	1.4417±0.0034	0.2703±0.0009	1.1423±0.0054	0.3313±0.0003	1.4971±0.0116
MSV 1 + WV_4	0.1654±0.0004	1.1819±0.0031	0.2461±0.0003	1.2106±0.0058	0.2641±0.0004	1.4449±0.0056	0.2714±0.0006	1.1412±0.0020	0.3321±0.0003	1.4995±0.0094
PRM + WV_4	0.1635±0.0000	1.1743±0.0000	0.2429±0.0000	1.1920±0.0000	0.2627±0.0000	1.4440±0.0000	0.2743±0.0000	1.1635±0.0000	0.3318±0.0000	1.4591±0.0000
MSV 4	**0.1065**±0.0177	**0.4677**±0.0362	**0.0821**±0.0267	**0.3663**±0.0206	**0.1306**±0.0357	**0.4793**±0.0424	**0.0908**±0.0374	**0.2472**±0.0384	**0.1265**±0.0309	**0.4424**±0.0423
Token Probs + WV_{16}	0.1212±0.0000	0.9249±0.0000	0.1566±0.0000	0.6566±0.0000	0.1752±0.0000	0.8604±0.0000	0.1545±0.0000	0.6618±0.0000	0.2235±0.0000	0.8289±0.0000
Self-consistency$_{16}$	0.1204±0.0000	0.9264±0.0000	0.1551±0.0000	0.6605±0.0000	0.1737±0.0000	0.8673±0.0000	0.1517±0.0000	0.6702±0.0000	0.2226±0.0000	0.8306±0.0000
Probe + WV_{16}	0.1293±0.0009	0.9282±0.0023	0.1644±0.0015	0.6382±0.0023	0.1800±0.0017	0.8120±0.0069	0.1579±0.0032	0.5808±0.0108	0.2256±0.0010	0.8281±0.0047
MSV 1 + WV_{16}	0.1290±0.0007	0.9299±0.0029	0.1655±0.0005	0.6443±0.0030	0.1824±0.0004	0.8198±0.0064	0.1611±0.0012	0.5922±0.0033	0.2275±0.0005	0.8366±0.0053
PRM + WV_{16}	0.1257±0.0000	0.9235±0.0000	0.1597±0.0000	0.6439±0.0000	0.1798±0.0000	0.8497±0.0000	0.1589±0.0000	0.6419±0.0000	0.2264±0.0000	0.8255±0.0000
MSV 16	**0.1127**±0.0108	**0.4615**±0.0291	**0.0869**±0.0197	**0.3289**±0.0200	**0.1317**±0.0197	**0.4724**±0.0414	**0.0844**±0.0234	**0.2079**±0.0290	**0.1268**±0.0205	**0.4385**±0.0411
Token Probs + WV_{64}	0.1158±0.0000	0.7592±0.0000	0.1346±0.0000	0.5194±0.0000	0.1549±0.0000	0.7096±0.0000	0.1254±0.0000	0.5219±0.0000	0.1970±0.0000	0.6686±0.0000
Self-consistency$_{64}$	**0.1149**±0.0000	0.7598±0.0000	0.1329±0.0000	0.5231±0.0000	0.1531±0.0000	0.7161±0.0000	0.1221±0.0000	0.5301±0.0000	0.1959±0.0000	0.6692±0.0000
Probe + WV_{64}	0.1205±0.0011	0.7788±0.0038	0.1437±0.0018	0.4894±0.0039	0.1611±0.0026	0.6564±0.0086	0.1288±0.0037	0.4265±0.0148	0.1999±0.0014	0.6564±0.0032
MSV 1 + WV_{64}	0.1202±0.0008	0.7732±0.0032	0.1451±0.0007	0.5024±0.0022	0.1664±0.0034	0.6646±0.0065	0.1347±0.0039	0.4444±0.0042	0.2020±0.0007	0.6662±0.0044
PRM + WV_{64}	0.1161±0.0000	0.7661±0.0000	0.1382±0.0000	0.5060±0.0000	0.1601±0.0000	0.7013±0.0000	0.1305±0.0000	0.5014±0.0000	0.2004±0.0000	0.6658±0.0000
MSV 64	0.1204±0.0109	**0.5443**±0.0501	**0.0556**±0.0135	**0.2904**±0.0230	**0.1085**±0.0122	**0.4986**±0.0234	**0.0350**±0.0166	**0.1501**±0.0213	**0.0867**±0.0106	**0.3705**±0.0394

Table 17: Best-of-N accuracy (↑), with verifiers at $N = 1, 4, 16, 64$.

N	Method	MATH	OlympiadBench	AMC12	AIME	Omni-MATH
1	—	0.6755±0.0000	0.4105±0.0000	0.3965±0.0000	0.2748±0.0000	0.2432±0.0000
4	Self-consistency$_4$	0.7049±0.0000	0.4484±0.0000	0.4286±0.0000	0.3251±0.0000	0.2633±0.0000
	Probe	0.7197±0.0006	0.4725±0.0008	0.4861±0.0041	0.3771±0.0012	0.2885±0.0005
	MSV 1	0.7155±0.0024	0.4681±0.0011	0.4854±0.0043	0.3758±0.0008	0.2878±0.0018
	PRM	0.7203±0.0000	0.4741±0.0000	**0.4879**±0.0000	0.3744±0.0000	**0.2927**±0.0000
	Probe + WV_4	**0.7250**±0.0009	**0.4770**±0.0003	0.4856±0.0030	0.3847±0.0010	0.2910±0.0009
	MSV 1 + WV_4	0.7224±0.0011	0.4755±0.0014	0.4845±0.0030	**0.3852**±0.0008	0.2900±0.0012
	PRM + WV_4	0.7151±0.0000	0.4655±0.0000	0.4599±0.0000	0.3570±0.0000	0.2783±0.0000
	MSV 4	0.7219±0.0012	0.4747±0.0021	0.4837±0.0020	0.3811±0.0018	0.2891±0.0015
16	Self-consistency$_{16}$	0.7210±0.0000	0.4676±0.0000	0.4478±0.0000	0.3566±0.0000	0.2801±0.0000
	Probe	0.7345±0.0014	0.5039±0.0023	0.5198±0.0037	0.4470±0.0025	0.3176±0.0020
	MSV 1	0.7238±0.0031	0.4895±0.0037	0.5231±0.0035	0.4325±0.0050	0.3102±0.0023
	PRM	0.7383±0.0000	0.5128±0.0000	0.5280±0.0000	0.4475±0.0000	**0.3231**±0.0000
	Probe + WV_{16}	0.7450±0.0028	0.5087±0.0019	0.5254±0.0072	0.4481±0.0096	0.3162±0.0037
	MSV 1 + WV_{16}	0.7440±0.0025	0.5104±0.0008	0.5235±0.0055	0.4460±0.0076	0.3133±0.0039
	PRM + WV_{16}	0.7266±0.0000	0.4860±0.0000	0.4701±0.0000	0.3817±0.0000	0.2879±0.0000
	MSV 16	**0.7472**±0.0009	**0.5154**±0.0037	**0.5399**±0.0061	**0.4615**±0.0046	0.3198±0.0032
64	Self-consistency$_{64}$	0.7321±0.0000	0.4844±0.0000	0.4403±0.0000	0.3594±0.0000	0.2790±0.0000
	Probe	0.7429±0.0030	0.5112±0.0062	0.5269±0.0154	0.4897±0.0098	0.3335±0.0077
	MSV 1	0.7246±0.0058	0.4629±0.0110	0.5164±0.0119	0.4621±0.0140	0.3179±0.0061
	PRM	0.7388±0.0000	0.5223±0.0000	0.5522±0.0000	0.4688±0.0000	0.3393±0.0000
	Probe + WV_{64}	0.7500±0.0032	0.5223±0.0058	0.5388±0.0073	0.4790±0.0067	0.3210±0.0017
	MSV 1 + WV_{64}	0.7442±0.0033	0.5210±0.0030	0.5313±0.0087	0.4714±0.0052	0.3179±0.0023
	PRM + WV_{64}	0.7344±0.0000	0.4933±0.0000	0.4851±0.0000	0.3884±0.0000	0.2969±0.0000
	MSV 64	**0.7607**±0.0033	**0.5366**±0.0064	**0.5866**±0.0138	**0.5098**±0.0073	**0.3540**±0.0023

Table 18: ECE (↓) and Brier score (↓) of confidence output on best-of-N answers, with verifiers at $N = 4, 16, 64$.

		MATH		OlympiadBench		AMC12		AIME		Omni-MATH	
N	**Method**	**ECE ↓**	**Brier ↓**	**ECE ↓**	**Brier ↓**	**ECE ↓**	**Brier ↓**	**ECE ↓**	**Brier ↓**	**ECE ↓**	**Brier ↓**
4	Self-consistency$_4$	0.1613±0.0000	0.1819±0.0000	0.2586±0.0000	0.2492±0.0000	0.2843±0.0000	0.2752±0.0000	0.2805±0.0000	0.2658±0.0000	0.3708±0.0000	0.3016±0.0000
	PRM	0.1308±0.0000	0.1642±0.0000	0.2016±0.0000	0.1754±0.0000	0.1386±0.0000	0.1944±0.0000	0.2363±0.0000	0.1638±0.0000	0.2241±0.0000	0.1807±0.0000
	PRM + WV$_4$	0.1607±0.0000	0.1823±0.0000	0.2554±0.0000	0.2515±0.0000	0.2687±0.0000	0.2791±0.0000	0.2785±0.0000	0.2679±0.0000	0.3724±0.0000	0.3087±0.0000
	Probe	0.1168±0.0448	**0.1501**±0.0171	0.1510±0.0689	0.1688±0.0327	0.1801±0.0639	0.1903±0.0333	0.1703±0.0579	0.1294±0.0297	0.2298±0.0777	0.1999±0.0498
	MSV 1	0.1316±0.0206	0.1658±0.0088	0.1675±0.0276	0.1812±0.0129	0.1985±0.0293	0.1964±0.0156	0.1661±0.0269	0.1230±0.0121	0.2514±0.0319	0.2186±0.0187
	Probe + WV$_4$	0.1797±0.0039	0.1928±0.0022	0.3066±0.0089	0.2830±0.0065	0.3050±0.0073	0.3010±0.0061	0.3204±0.0133	0.2889±0.0090	0.4396±0.0109	0.3680±0.0105
	MSV 1 + WV$_4$	0.1815±0.0034	0.1936±0.0018	0.3044±0.0066	0.2821±0.0053	0.3014±0.0063	0.2973±0.0033	0.3149±0.0079	0.2840±0.0033	0.4354±0.0091	0.3653±0.0078
	MSV 4	**0.1133**±0.0187	0.1505±0.0059	**0.1031**±0.0345	**0.1432**±0.0089	**0.1477**±0.0412	**0.1739**±0.0134	**0.1019**±0.0489	**0.1020**±0.0160	**0.1695**±0.0402	**0.1705**±0.0182
16	Self-consistency$_{16}$	0.1292±0.0000	0.1702±0.0000	0.1971±0.0000	0.2174±0.0000	0.2235±0.0000	0.2420±0.0000	0.1927±0.0000	0.2279±0.0000	0.2961±0.0000	0.2511±0.0000
	PRM	**0.1092**±0.0000	0.1582±0.0000	0.1946±0.0000	0.1795±0.0000	**0.1162**±0.0000	0.2000±0.0000	0.2288±0.0000	0.1726±0.0000	0.2227±0.0000	0.1978±0.0000
	PRM + WV$_{16}$	0.1334±0.0000	0.1678±0.0000	0.1908±0.0000	0.2165±0.0000	0.2138±0.0000	0.2387±0.0000	0.1841±0.0000	0.2219±0.0000	0.2986±0.0000	0.2518±0.0000
	Probe	0.1559±0.0369	0.1699±0.0208	0.2294±0.0679	0.2214±0.0460	0.2657±0.0607	0.2512±0.0454	0.2364±0.0601	0.1888±0.0429	0.3296±0.0823	0.2818±0.0680
	MSV 1	0.1740±0.0172	0.1909±0.0115	0.2519±0.0267	0.2415±0.0183	0.2725±0.0268	0.2574±0.0191	0.2407±0.0280	0.1877±0.0221	0.3542±0.0327	0.3085±0.0251
	Probe + WV$_{16}$	0.1353±0.0017	0.1704±0.0012	0.2073±0.0070	0.2203±0.0018	0.1988±0.0046	0.2372±0.0015	0.1633±0.0035	0.2143±0.0028	0.3195±0.0064	0.2761±0.0056
	MSV 1 + WV$_{16}$	0.1356±0.0007	0.1726±0.0012	0.2049±0.0053	0.2238±0.0021	0.2002±0.0019	0.2380±0.0007	0.1659±0.0040	0.2176±0.0018	0.3201±0.0056	0.2770±0.0047
	MSV 16	0.1160±0.0142	**0.1453**±0.0040	**0.1059**±0.0288	**0.1430**±0.0102	0.1462±0.0273	**0.1758**±0.0108	**0.0876**±0.0301	**0.1034**±0.0118	**0.1776**±0.0328	**0.1792**±0.0165
64	Self-consistency$_{64}$	0.1259±0.0000	0.1689±0.0000	0.1680±0.0000	0.2140±0.0000	0.2196±0.0000	0.2251±0.0000	0.1733±0.0000	0.2171±0.0000	0.2830±0.0000	0.2375±0.0000
	PRM	**0.0884**±0.0000	0.1600±0.0000	0.1667±0.0000	0.1860±0.0000	**0.1174**±0.0000	0.2062±0.0000	0.2167±0.0000	0.1814±0.0000	0.2326±0.0000	0.2152±0.0000
	PRM + WV$_{64}$	0.1269±0.0000	0.1652±0.0000	0.1694±0.0000	0.2073±0.0000	0.1885±0.0000	0.2339±0.0000	0.1606±0.0000	0.2127±0.0000	0.2757±0.0000	0.2443±0.0000
	Probe	0.1848±0.0276	0.1901±0.0200	0.3048±0.0552	0.2760±0.0498	0.3288±0.0573	0.3066±0.0502	0.3013±0.0588	0.2580±0.0503	0.4185±0.0796	0.3674±0.0778
	MSV 1	0.2128±0.0143	0.2210±0.0120	0.3555±0.0312	0.3227±0.0260	0.3520±0.0174	0.3318±0.0148	0.3162±0.0348	0.2662±0.0327	0.4504±0.0303	0.4059±0.0269
	Probe + WV$_{64}$	0.1259±0.0029	0.1653±0.0010	0.1736±0.0020	0.2051±0.0021	0.1823±0.0155	0.2306±0.0036	0.1222±0.0168	0.2083±0.0069	0.2888±0.0049	0.2558±0.0021
	MSV 1 + WV$_{64}$	0.1300±0.0016	0.1665±0.0009	0.1769±0.0059	0.2110±0.0008	0.1929±0.0086	0.2293±0.0019	0.1302±0.0100	0.2116±0.0035	0.2920±0.0033	0.2588±0.0030
	MSV 64	0.1323±0.0147	**0.1487**±0.0040	**0.0803**±0.0281	**0.1307**±0.0158	0.1731±0.0207	**0.2052**±0.0094	**0.0749**±0.0500	**0.1176**±0.0271	**0.1303**±0.0273	**0.1717**±0.0201

Table 19: AUROC (↑) and Brier score (↓) of verifiers in **Streaming Answers**.

Method	MATH		OlympiadBench		AMC12		AIME		Omni-MATH	
	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓	AUROC ↑	Brier ↓
Token Probs	0.7910±0.0000	0.3399±0.0000	0.8091±0.0000	0.4266±0.0000	0.7856±0.0000	0.3979±0.0000	0.8858±0.0000	0.3217±0.0000	0.7974±0.0000	0.4746±0.0000
PRM	**0.8692**±0.0000	0.1902±0.0000	**0.8542**±0.0000	0.1607±0.0000	**0.8658**±0.0000	0.1621±0.0000	**0.9309**±0.0000	0.1185±0.0000	0.8227±0.0000	0.1497±0.0000
Probe	0.8690±0.0135	**0.1592**±0.0101	0.8317±0.0089	0.1478±0.0232	0.8621±0.0108	0.1413±0.0195	0.9120±0.0141	0.0796±0.0105	**0.8229**±0.0142	0.1527±0.0242
MSV 1	0.8443±0.0223	0.1601±0.0125	0.8169±0.0135	**0.1425**±0.0076	0.8546±0.0104	**0.1343**±0.0028	0.9056±0.0076	**0.0642**±0.0028	0.7951±0.0125	**0.1381**±0.0084
Token Probs + WV_4	0.7743±0.0000	0.3266±0.0000	0.7837±0.0000	0.4051±0.0000	0.7596±0.0000	0.3775±0.0000	0.8601±0.0000	0.2990±0.0000	0.7757±0.0000	0.4522±0.0000
PRM + WV_4	0.8533±0.0000	0.2006±0.0000	0.8142±0.0000	0.1678±0.0000	0.8296±0.0000	0.1689±0.0000	0.8646±0.0000	0.1241±0.0000	0.7975±0.0000	0.1494±0.0000
Probe + WV_4	0.8528±0.0027	0.1619±0.0044	0.8104±0.0037	0.1430±0.0095	0.8219±0.0095	0.1395±0.0106	0.8702±0.0206	0.0842±0.0067	0.8009±0.0079	0.1331±0.0200
MSV 1 + WV_4	0.8367±0.0150	0.1696±0.0113	0.7974±0.0058	0.1420±0.0029	0.8427±0.0100	0.1330±0.0059	0.8826±0.0068	0.0763±0.0020	0.7699±0.0098	0.1228±0.0081
MSV 4	**0.8917**±0.0133	**0.1324**±0.0034	**0.8683**±0.0169	**0.1197**±0.0094	**0.9002**±0.0118	**0.1068**±0.0033	**0.9312**±0.0101	**0.0472**±0.0053	**0.8327**±0.0127	**0.1160**±0.0088
Token Probs + WV_{16}	0.7681±0.0000	0.3242±0.0000	0.7804±0.0000	0.4029±0.0000	0.7550±0.0000	0.3750±0.0000	0.8604±0.0000	0.2967±0.0000	0.7724±0.0000	0.4507±0.0000
PRM + WV_{16}	0.8430±0.0000	0.2075±0.0000	0.8018±0.0000	0.1715±0.0000	0.8156±0.0000	0.1725±0.0000	0.8497±0.0000	0.1266±0.0000	0.7901±0.0000	0.1508±0.0000
Probe + WV_{16}	0.8414±0.0015	0.1666±0.0024	0.8013±0.0037	0.1453±0.0085	0.8113±0.0111	0.1418±0.0088	0.8609±0.0214	0.0873±0.0060	0.7966±0.0076	0.1315±0.0193
MSV 1 + WV_{16}	0.8255±0.0150	0.1761±0.0118	0.7889±0.0056	0.1450±0.0027	0.8341±0.0102	0.1367±0.0059	0.8749±0.0068	0.0804±0.0020	0.7634±0.0099	0.1217±0.0078
MSV 16	**0.9037**±0.0201	**0.1104**±0.0105	**0.8946**±0.0229	**0.1054**±0.0076	**0.9105**±0.0115	**0.1041**±0.0049	**0.9532**±0.0069	**0.0375**±0.0029	**0.8591**±0.0029	**0.1100**±0.0063
Token Probs + WV_{64}	0.7676±0.0000	0.3238±0.0000	0.7798±0.0000	0.4026±0.0000	0.7527±0.0000	0.3749±0.0000	0.8592±0.0000	0.2966±0.0000	0.7718±0.0000	0.4504±0.0000
PRM + WV_{64}	0.8385±0.0000	0.2098±0.0000	0.7990±0.0000	0.1723±0.0000	0.8121±0.0000	0.1733±0.0000	0.8451±0.0000	0.1272±0.0000	0.7881±0.0000	0.1512±0.0000
Probe + WV_{64}	0.8388±0.0015	0.1679±0.0019	0.7993±0.0038	0.1457±0.0082	0.8089±0.0112	0.1421±0.0085	0.8588±0.0214	0.0878±0.0059	0.7963±0.0076	0.1310±0.0191
MSV 1 + WV_{64}	0.8225±0.0152	0.1778±0.0119	0.7871±0.0057	0.1455±0.0027	0.8323±0.0105	0.1375±0.0060	0.8731±0.0069	0.0812±0.0020	0.7622±0.0101	0.1214±0.0078
MSV 64	**0.9215**±0.0125	**0.1137**±0.0022	**0.8971**±0.0136	**0.0956**±0.0118	**0.8917**±0.0034	**0.1154**±0.0033	**0.9662**±0.0016	**0.0309**±0.0013	**0.8409**±0.0084	**0.1205**±0.0051

Table 20: ECE (↓) and NLL (↓) of verifiers in **Streaming Answers**.

Method	MATH		OlympiadBench		AMC12		AIME		Omni-MATH	
	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓	ECE ↓	NLL ↓
Token Probs	0.3732±0.0000	1.1043±0.0000	0.5255±0.0000	1.1625±0.0000	0.4944±0.0000	1.1466±0.0000	0.4886±0.0000	0.8724±0.0000	0.6045±0.0000	1.2632±0.0000
PRM	0.2060±0.0000	0.5667±0.0000	0.1883±0.0000	0.5046±0.0000	0.1775±0.0000	0.5076±0.0000	0.2456±0.0000	0.4127±0.0000	0.2298±0.0000	0.4819±0.0000
Probe	**0.0934**±0.0416	**0.5184**±0.0528	0.1317±0.0382	0.5206±0.0914	0.1292±0.0461	0.4841±0.0689	0.1160±0.0385	0.2847±0.0421	0.1656±0.0470	0.5194±0.0945
MSV 1	0.0939±0.0163	0.5356±0.0441	**0.0911**±0.0163	**0.5055**±0.0208	**0.0862**±0.0164	**0.4532**±0.0039	**0.0511**±0.0229	**0.2412**±0.0113	**0.1157**±0.0132	**0.4974**±0.0232
Token Probs + WV_4	0.3499±0.0000	0.9724±0.0000	0.5001±0.0000	1.0713±0.0000	0.4680±0.0000	1.0312±0.0000	0.4595±0.0000	0.8013±0.0000	0.5843±0.0000	1.1794±0.0000
PRM + WV_4	0.1974±0.0000	0.5887±0.0000	0.1705±0.0000	0.5201±0.0000	0.1690±0.0000	0.5224±0.0000	0.2097±0.0000	0.4243±0.0000	0.2165±0.0000	0.4822±0.0000
Probe + WV_4	**0.0628**±0.0368	0.4975±0.0160	0.0784±0.0407	0.4530±0.0219	0.0790±0.0444	0.4438±0.0281	0.0830±0.0286	0.2985±0.0248	0.1404±0.0546	0.4260±0.0553
MSV 1 + WV_4	0.0694±0.0249	0.5329±0.0369	**0.0521**±0.0087	0.4627±0.0083	**0.0396**±0.0171	0.4168±0.0163	**0.0240**±0.0113	0.2600±0.0072	**0.0843**±0.0154	**0.4068**±0.0196
MSV 4	0.0814±0.0133	**0.4637**±0.0294	0.0797±0.0143	**0.4289**±0.0270	0.0638±0.0093	**0.3709**±0.0172	0.0271±0.0120	**0.1814**±0.0187	0.0959±0.0116	0.4206±0.0287
Token Probs + WV_{16}	0.3441±0.0000	0.9451±0.0000	0.4967±0.0000	1.0605±0.0000	0.4643±0.0000	1.0155±0.0000	0.4566±0.0000	0.7951±0.0000	0.5826±0.0000	1.1731±0.0000
PRM + WV_{16}	0.1969±0.0000	0.6035±0.0000	0.1700±0.0000	0.5280±0.0000	0.1653±0.0000	0.5302±0.0000	0.2027±0.0000	0.4301±0.0000	0.2143±0.0000	0.4855±0.0000
Probe + WV_{16}	**0.0573**±0.0197	0.5039±0.0067	0.0730±0.0375	0.4563±0.0191	0.0734±0.0379	0.4476±0.0225	0.0854±0.0256	0.3080±0.0233	0.1374±0.0550	0.4180±0.0514
MSV 1 + WV_{16}	0.0709±0.0311	0.5404±0.0354	**0.0497**±0.0131	0.4651±0.0080	**0.0427**±0.0169	0.4228±0.0161	**0.0295**±0.0103	0.2705±0.0066	**0.0784**±0.0170	0.3969±0.0188
MSV 16	0.0666±0.0211	**0.3781**±0.0461	0.0606±0.0166	**0.3546**±0.0195	0.0563±0.0108	**0.3672**±0.0173	0.0333±0.0146	**0.1498**±0.0109	0.0883±0.0122	**0.3888**±0.0163
Token Probs + WV_{64}	0.3429±0.0000	0.9397±0.0000	0.4962±0.0000	1.0584±0.0000	0.4637±0.0000	1.0133±0.0000	0.4561±0.0000	0.7946±0.0000	0.5823±0.0000	1.1714±0.0000
PRM + WV_{64}	0.1959±0.0000	0.6084±0.0000	0.1697±0.0000	0.5299±0.0000	0.1677±0.0000	0.5321±0.0000	0.2016±0.0000	0.4315±0.0000	0.2138±0.0000	0.4866±0.0000
Probe + WV_{64}	**0.0583**±0.0146	0.5063±0.0047	0.0728±0.0365	0.4570±0.0186	0.0764±0.0338	0.4482±0.0215	0.0871±0.0255	0.3099±0.0230	0.1368±0.0551	0.4161±0.0506
MSV 1 + WV_{64}	0.0724±0.0316	0.5428±0.0351	**0.0504**±0.0134	0.4651±0.0079	**0.0444**±0.0162	0.4244±0.0159	0.0312±0.0093	0.2727±0.0065	**0.0769**±0.0176	**0.3946**±0.0187
MSV 64	0.0682±0.0151	**0.3989**±0.0322	0.0661±0.0236	**0.3370**±0.0346	0.0995±0.0065	**0.4117**±0.0210	**0.0097**±0.0048	**0.1202**±0.0027	0.1242±0.0069	0.3998±0.0284

D.2 Streaming Answers

Next, we analyze the full results for the **Streaming Answers** setting. Table 19 and Table 20 show AUROC, Brier score, ECE, and NLL. Consistent with the findings in Section 4.3, MSV_N generally improves on single-sequence verifier baselines across the reported metrics and datasets, with larger gains at higher N. Minor exceptions exist: for instance, MSV_{64} slightly deteriorates compared to MSV_{16} on some metrics, e.g., AMC12 Brier. Token Probs performs poorly across all datasets, so we omit it from all parallel early stopping experiments.

We provide the full accuracy–estimated-latency tradeoff curves for $N = 64$ in Figure 9, and for $N = 4$ and $N = 16$ in Figure 10. Consistent with the **Terminal Answers** setting, the advantage of MSV_N over single-sequence verifier baselines becomes more pronounced starting at $N = 16$. To be concrete, MSV_{16} improves the tradeoff on MATH, OB, and AIME, while it is on par with the best baseline Probe on AMC12 and OM.

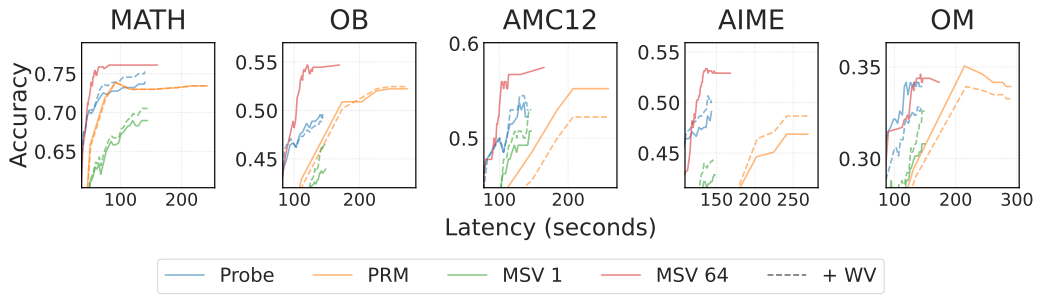
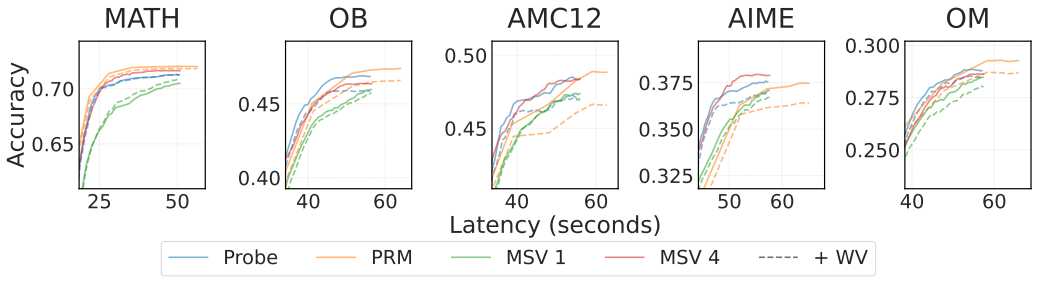
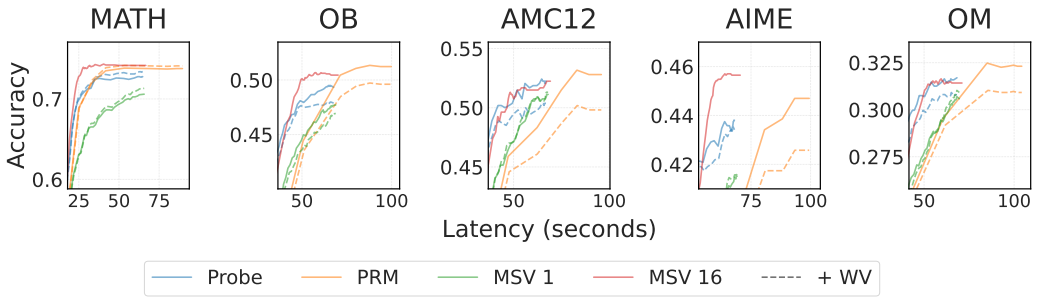


Figure 9: Accuracy versus estimated latency curves in **Streaming Answers**, with $N = 64$. Each curve sweeps the stopping threshold; higher accuracy at a fixed estimated latency is better. Latencies combine measured generation time and estimated component costs (Section B.1). Accuracy–token-position tradeoff curves are in Figure 5.



(a) $N = 4$



(b) $N = 16$

Figure 10: Accuracy versus estimated latency in **Streaming Answers**, for $N \in \{4, 16\}$. Each curve sweeps the stopping threshold. Latencies combine measured generation time and estimated component costs, including simulated caching savings (Section B.1).

Table 21: Best-of-64 accuracy on Omni-MATH under different equivalence checkers. Checker accuracy is the pairwise agreement with SymPy. Answer correctness is always evaluated with SymPy. MSV_{64} (SymPy) is trained with SymPy and logit averaging; MSV_{64} (w/o logit avg.) is retrained for each checker.

Checker	Checker accuracy	Self-consistency	Probe+WV	PRM+WV	MSV_1+WV	MSV_{64} (SymPy)	MSV_{64} (w/o logit avg.)
SymPy (reference)	100.0%	0.279±0.000	0.321±0.002	0.297±0.000	0.318±0.002	**0.356**±0.002	0.348±0.007
Exact string match	99.0%	0.277±0.000	0.321±0.002	0.295±0.000	0.318±0.002	**0.355**±0.003	0.345±0.005
Jaccard (≥ 0.8)	97.0%	0.277±0.000	0.316±0.002	0.292±0.000	0.315±0.003	**0.352**±0.002	0.345±0.008
Embedding (≥ 0.85)	89.4%	0.272±0.000	0.301±0.002	0.286±0.000	0.298±0.003	0.328±0.002	**0.344**±0.008
Embedding (≥ 0.6)	61.2%	0.257±0.000	0.264±0.003	0.257±0.000	0.263±0.002	0.256±0.001	**0.338**±0.009
Random	46.7%	0.246±0.000	0.246±0.001	0.243±0.000	0.246±0.001	0.251±0.003	**0.336**±0.015

D.3 Sensitivity to Equivalence Checker Quality

Experimental details. We experiment with five alternative checkers with varying agreement with SymPy—exact string match, Jaccard similarity (≥ 0.8), embedding cosine similarity at two thresholds (≥ 0.85 and ≥ 0.6), and random coin flip—while always using SymPy as the oracle for measuring correctness. We use the bge-small-en-v1.5 model (Xiao et al., 2024) for embedding cosine similarity. For each alternative checker, we retrain MSV with that checker and remove logit averaging in Equation (1), so the weak equivalence checkers are used only within Multi-Mask Transformer Block and feature augmentation, both of which are trained components. We also evaluate a separate MSV_{64} variant trained with SymPy and logit averaging, replacing its checker only at evaluation time.

Results. We report in Table 21 the full results. "Checker accuracy" is the accuracy of the weak checker in classifying whether two independently sampled answers on a problem are equivalent according to SymPy. First, we find that removing logit averaging drops the best-of-64 accuracy of MSV_{64} from 0.356 to 0.348, when evaluated using the ground truth checker SymPy. MSV_{64} trained with SymPy and logit averaging loses accuracy as the evaluation checker becomes less consistent with SymPy. This comparison changes both the checker distribution and the reliability of output aggregation. Self-consistency and weighted voting also quickly degrade to the average accuracy of the base language model. The MSV_{64} variant retrained with each checker and without logit averaging retains more of its accuracy under the weakest checkers. These results support tolerance to noisy agreement signals when training and evaluation use the same checker; they do not isolate retraining from the removal of logit averaging.

D.4 Additional Base Language Models

We repeat the **Terminal Answers** experiments with three additional base LMs: DeepSeek-R1-Distill-Llama-8B, Qwen3-1.7B in thinking mode, and Llama-3.2-1B-Instruct. For the first two, we train on DeepMath-103K and evaluate on AIME. Since Llama-3.2-1B-Instruct has near-zero accuracy on AIME, we train and evaluate it on the MATH train/test splits. Table 4 summarizes Brier scores and best-of-64 accuracy against the strongest single-sequence verifier baseline for each metric. Table 22 reports AUROC, Brier score, ECE, and NLL for different verifiers, and Table 23 reports best-of-64 accuracy. Unless otherwise stated, we reuse the same training setup and hyperparameters as in the main experiments.

DeepSeek-R1-Distill-Llama-8B on AIME. We first consider a larger base model, DeepSeek-R1-Distill-Llama-8B, in the AIME **Terminal Answers** setup. MSV_{64} achieves the highest AUROC and lowest Brier score among the compared verifiers. When used for best-of-64 selection, MSV_{64} also outperforms all baselines, including Probe with and without weighted voting.

Qwen3-1.7B (thinking mode) on AIME. We next evaluate Qwen3-1.7B in thinking mode on AIME. MSV_{64} achieves lower Brier score and higher AUROC than Probe, although its mean ECE is slightly higher. MSV_{64} also has the highest mean best-of-64 accuracy, with a smaller gain than for the other two additional models.

Table 22: Brier score, AUROC, ECE, and NLL of verifiers across different base language models. R1-8B denotes DeepSeek-R1-Distill-Llama-8B.

Base LM	Dataset	Metric	Probe	MSV_1	MSV_{64}
R1-8B	AIME	Brier ↓	0.0673±0.0104	0.0738±0.0151	**0.0363**±0.0090
		AUROC ↑	0.9672±0.0017	0.9750±0.0043	**0.9901**±0.0012
		ECE ↓	0.0479±0.0272	0.0694±0.0201	**0.0367**±0.0148
		NLL ↓	0.2555±0.0407	0.3240±0.0708	**0.1868**±0.0352
Qwen3-1.7B	AIME	Brier ↓	0.0848±0.0333	0.0770±0.0059	**0.0509**±0.0246
		AUROC ↑	0.9530±0.0174	0.9557±0.0053	**0.9863**±0.0011
		ECE ↓	0.0627±0.0656	**0.0438**±0.0169	0.0639±0.0448
		NLL ↓	0.2884±0.1024	0.2705±0.0149	**0.1945**±0.0722
Llama-3.2-1B	MATH	Brier ↓	0.1368±0.0363	0.1324±0.0050	**0.0565**±0.0056
		AUROC ↑	0.7856±0.0137	0.7294±0.0304	**0.9513**±0.0010
		ECE ↓	0.0982±0.1043	0.0936±0.0157	**0.0430**±0.0159
		NLL ↓	0.4264±0.0904	0.4931±0.0732	**0.2043**±0.0150

Table 23: Best-of-64 accuracy across different base language models. R1-8B denotes DeepSeek-R1-Distill-Llama-8B.

Base LM	Dataset	Probe	Probe+WV	MSV_1	MSV_1+WV	MSV_{64}
R1-8B	AIME	0.6116±0.0098	0.6357±0.0131	0.6047±0.0088	0.6492±0.0091	**0.6625**±0.0054
Qwen3-1.7B	AIME	0.5223±0.0130	0.5089±0.0179	0.4545±0.0097	0.5288±0.0089	**0.5357**±0.0109
Llama-3.2-1B	MATH	0.2393±0.0143	0.3375±0.0051	0.1670±0.0201	0.2946±0.0192	**0.3777**±0.0051

Llama-3.2-1B-Instruct on MATH. Finally, we consider Llama-3.2-1B-Instruct, a non-reasoning model whose accuracy on AIME is close to zero. For this model we therefore train and evaluate on the MATH train/test splits. Even in this lower-accuracy regime, MSV_{64} achieves lower Brier score and higher AUROC than Probe, and MSV_{64} achieves the highest best-of-64 accuracy.

Across these three base LMs, MSV_{64} has lower Brier score and higher mean best-of-64 accuracy than the compared baselines; the size of the gains and the ECE results vary by model.