Test-Time Compute Scaling and Language Model Performance on Reasoning Benchmarks

Assignee Research

June 6, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: How does test-time compute scaling improve language model performance on reasoning benchmarks v14. 11 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 3.5/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: ARS: Adaptive Reasoning Suppression for Efficient Large Reasoning Language Models. Research question: How does test-time compute scaling improve language model performance on reasoning benchmarks v14.

2 Methodology

Systematic literature search across multiple databases yielded 14 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 3.5/10.

3 Results

14 papers retrieved. 11 claims extracted; 0 independently verified. Quality review score: 3.5/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
ARS consistently achieves superior length reduction while maintaining competitive accuracy across all model scales.	×	0.11
ARS operates through three core components: (1) Multi-checkpoint certainty estimation, (2) Progressive threshold adaptat	×	0.14
ARS establishes multiple checkpoints {c1, c2, . . . , ck} at regular intervals during generation.	×	0.02
At each checkpoint ci, ARS estimates model certainty through tentative answer probing.	×	0.05
The heuristic difficulty estimation is used to schedule the mode of operation in ARS.	×	0.03
ARS uses different policies (CoDFastPolicy, ElasticModeratePolicy, DeepReflectPolicy) based on the difficulty of the que	×	0.03
ARS sets a maximum token limit of 1200 tokens per response to prevent excessive generation.	×	0.02
ARS aims to minimize the expected output length E[T] while preserving reasoning accuracy.	×	0.06
ARS uses specific keywords T = {"Wait", "But", "Alternatively", . . .} to identify reflection behaviors that often lead	×	0.05
ARS uses a probing prompt to generate a tentative answer ai at each checkpoint ci.	×	0.03
ARS computes the certainty score based on the tentative answer ai at each checkpoint ci.	×	0.04

References

- http://arxiv.org/abs/2601.01982v1
- http://arxiv.org/abs/2510.00071v2

- http://arxiv.org/abs/2508.15478v2