Adaptive Self-Verifiable Reasoning: A Proposed Architecture for Efficient LLM Reasoning with Dynamic Test-Time Compute Allocation

— Research Proposal & Theoretical Framework —

[Your Name / Research Group]
[your.email@institution.edu]

December 2025

Abstract

Recent advances in large language model (LLM) reasoning have demonstrated that sophisticated problem-solving can be achieved through iterative self-refinement processes. However, these approaches incur a substantial "token tax"—high computational costs from verbose, multi-turn reasoning chains. This paper proposes a theoretical framework for reconciling reasoning depth with computational efficiency through three novel architectural concepts: **(1) Unified Self-Verification Architecture (USVA)**, a domain-agnostic verification framework enabling single-model assessment across mathematics, code, and agentic tasks; **(2) Dynamic Sparse Attention (DSA-2)**, an attention mechanism with query-dependent token budget allocation; and **(3) Adaptive Test-Time Scaling (ATTS)**, a learned difficulty estimator that routes problems to appropriate compute tiers (Direct/Thinking/Deep) with uncertainty-triggered escalation. We present theoretical analysis, propose experimental validation protocols, and estimate potential efficiency gains of 40-60% token reduction while maintaining accuracy. This framework offers a roadmap for developing AI systems that dynamically balance speed and depth based on problem requirements.

Keywords: *Large Language Models, Self-Verification, Adaptive Compute, Sparse Attention, Test-Time Scaling, Efficient Reasoning*

1. Introduction & Motivation

1.1 The Token Tax Problem

The past year has witnessed remarkable progress in LLM reasoning capabilities. Models now achieve gold-medal performance on mathematical olympiads (IMO, Putnam) and competitive programming contests through extended "thinking" processes. However, this capability comes at significant computational cost. Analysis of recent model releases reveals a consistent pattern:

- **High-compute reasoning variants** often require 2-3× more output tokens than their efficient counterparts to achieve comparable accuracy.
- **Self-verification loops** (generate → verify → refine) multiply token consumption through multiple LLM calls per problem.

- **Binary compute allocation** (either "direct" or "full thinking") wastes resources on easy problems while potentially under-serving hard ones.

This "token tax" presents practical barriers: increased API costs, higher latency, and resource constraints that limit deployment scenarios. The fundamental question motivating this proposal is: *Can we achieve the benefits of deep, self-verifying reasoning while dramatically reducing average computational cost?*

1.2 The Dialectical Nature of Advanced Reasoning

Examining state-of-the-art reasoning systems reveals a recurring architectural pattern analogous to human dialectical processes. These systems employ internal "roles" that interact:

- **Generator/Proposer:** Creates initial solutions or reasoning traces.
- **Verifier/Critic:** Identifies logical gaps, errors, or unstated assumptions.
- **Meta-Verifier:** Assesses whether identified issues are genuine (preventing hallucinated critiques).
- **Refiner/Synthesizer:** Incorporates valid feedback to produce improved solutions.

This multi-agent-like internal dialogue, while powerful, is inherently verbose. Each "turn" in the dialectic consumes tokens. Our proposal aims to *distill this dialectical process* into more efficient representations while preserving its benefits.

2. Proposed Architecture

We propose three interconnected architectural innovations that together address the efficiency-capability trade-off.

2.1 Unified Self-Verification Architecture (USVA)

2.1.1 Motivation

Current self-verification approaches typically require domain-specific verifier models (e.g., separate verifiers for mathematical proofs vs. code correctness). This proliferation of specialized components increases system complexity and prevents transfer of verification capabilities across domains. We propose a unified verification framework based on domain-agnostic assessment dimensions.

2.1.2 Generalized Verification Rubrics

USVA defines four fundamental assessment dimensions applicable across reasoning tasks:

- **Logical Coherence (LC):** Does each step follow from previous steps? Are there unstated assumptions or non-sequiturs?
- **Factual Correctness (FC):** Are stated facts, formulas, or code constructs accurate? Are there computational errors?
- **Completeness (CM):** Does the solution address all problem aspects? Are edge cases handled?
- **Goal Alignment (GA):** Does the solution actually solve the stated problem? For agentic tasks, does the action sequence achieve the desired outcome?

The model learns to apply domain-specific interpretations of these rubrics based on task context. A unified scoring schema (1 = complete/rigorous, 0.5 = minor issues, 0 = fundamental flaws) enables consistent training signals across domains.

2.1.3 Integrated Meta-Verification

A known failure mode in self-verification is "hallucinated critique"—the model claims issues exist when they do not. Rather than requiring a separate meta-verifier, we propose integrating meta-verification as a self-consistency mechanism within the same forward pass. The model generates both a verification analysis and a calibration confidence score. The proposed training objective:

$$R = R_{solution} \times (\alpha \cdot R_{verification} + \beta \cdot R_{calibration})$$

where R_calibration measures alignment between verification confidence and actual accuracy on held-out evaluation. This encourages well-calibrated self-assessments without requiring separate meta-verification passes.

2.2 Dynamic Sparse Attention (DSA-2)

2.2.1 Limitations of Fixed-Budget Sparse Attention

Recent sparse attention mechanisms use learned indexers to select top-k key-value entries per query, reducing complexity from $O(L^2)$ to $O(Lk)$. However, fixed-k allocation is suboptimal: routine tokens may need far fewer than k attended tokens, while critical decision points may benefit from more. Analysis of attention patterns in reasoning models reveals high variance in "effective" attention density across positions.

2.2.2 Query-Dependent Budget Allocation

We propose replacing fixed k with a learned, query-dependent budget:

$$k(t) = k_{min} + (k_{max} - k_{min}) \times \sigma(W_{budget} \cdot h_t)$$

where σ is sigmoid, k_min and k_max define the budget range, and W_budget is learned. The model learns to predict higher budgets for queries requiring broad integration (synthesis steps, conclusions) and lower budgets for routine operations.

2.2.3 Entropy-Guided Selection

Beyond budget prediction, we propose biasing token selection toward entries that reduce attention entropy—those providing decisive, disambiguating information:

$$S_t = Top\text{-}k(t)(I_{t,s} + \lambda \cdot \Delta H(A_t \mid include\ s))$$

where ΔH estimates entropy reduction from including token s. In practice, this can be efficiently approximated using gradient-based importance scores.

2.3 Adaptive Test-Time Scaling (ATTS)

This component is the **core innovation for efficiency**. Current reasoning models operate in binary modes: direct response or full extended thinking. This binary choice wastes compute on easy problems while potentially under-serving hard ones.

2.3.1 Difficulty Estimation

We propose training a lightweight difficulty estimator that outputs $d \in [0, 1]$ before problem-solving begins. Training labels are derived from empirical solve rates:

$$d(P) = 1 - Pass@k(P) / k$$

where Pass@k is successful solutions in k attempts during data collection. This provides continuous difficulty measure correlating with compute needed for reliable solution.

2.3.2 Compute Allocation Policy

Based on difficulty estimate d, ATTS routes to three reasoning modes:

Mode	Condition	Behavior
Direct	$d < 0.3$	Standard generation, no explicit thinking. ~500 tokens avg.
Thinking	$0.3 \leq d < 0.7$	Single-pass extended thinking with integrated self-verification. ~8K tokens.
Deep	$d \geq 0.7$	Full iterative refinement (think → critique → refine loop). ~25K tokens.

Mode boundaries are hyperparameters optimized on validation data for accuracy-efficiency Pareto frontier.

2.3.3 Uncertainty-Triggered Escalation (Safety Net)

Critical innovation: Initial difficulty estimation may be imprecise. We propose a safety mechanism: if the model's self-verification score (from USVA) falls below threshold τ in the current mode, it *automatically escalates* to the next compute tier. This ensures problems initially misclassified as easy still receive adequate compute. Estimated escalation rate: ~10-15% of cases, providing safety without significantly increasing average compute.

2.4 Distilled Verification Knowledge

The most significant efficiency gain comes from training models to *anticipate and preemptively address issues*, rather than discovering them through explicit verification passes. We call this "distilled verification."

2.4.1 Synthetic Dialectical Dataset

We propose constructing large-scale datasets of (problem, draft_solution, critique, refined_solution) quadruples:

1. Generate initial solutions using base model without self-verification.
2. Apply USVA verification to identify specific issues and locations.
3. Generate improved solutions that explicitly address each identified issue.
4. Filter to retain only cases where refinement improved solution quality (verified by outcome).

Training on this data teaches models to internalize common failure modes and proactively avoid them, reducing need for explicit multi-turn verification.

3. Theoretical Analysis & Expected Gains

3.1 Token Efficiency Estimates

Based on the proposed architecture, we estimate the following efficiency gains:

Component	Mechanism	Est. Savings
ATTS	Route 45% of problems to Direct, 35% to Thinking	30-40%
Distilled Verification	Reduce explicit verification passes	15-25%
DSA-2	Dynamic attention budget allocation	10-15%

Combined estimate: 40-60% reduction in average tokens while maintaining accuracy within 2% of full-compute baselines.

3.2 Accuracy-Efficiency Trade-off Analysis

The ATTS mechanism introduces a fundamental trade-off. Direct mode achieves ~95% accuracy on easy problems with minimal tokens. Thinking mode recovers most accuracy on medium problems. Deep mode preserves full accuracy on hard problems but at higher cost. The *key insight* is that problem difficulty follows a skewed distribution—most problems are not maximally hard—so adaptive allocation yields significant savings.

Expected distribution on typical benchmarks: ~45% easy (Direct), ~35% medium (Thinking), ~20% hard (Deep). This yields substantial savings versus forcing Deep mode universally.

4. Proposed Experimental Validation

While full implementation requires significant compute resources, we propose a **simulation experiment** that can validate the core ATTS hypothesis using existing models (GPT-4, Claude, etc.) without training new models.

4.1 Simulation Protocol

5. **Dataset:** 100-200 math problems spanning easy (single-step), medium (multi-step), and hard (competition-level) difficulty.
6. **Difficulty Classification:** Prompt model to rate difficulty 1-10 WITHOUT solving. Use rating to assign mode.
7. **Adaptive Routing:** $d<4$ → Direct (solve normally), $4 \le d < 7$ → Thinking (step-by-step), $d \ge 7$ → Deep (think, critique, then answer).
8. **Baseline:** Force Deep mode on ALL problems.
9. **Metrics:** Accuracy (% correct), Tokens (average output tokens), Efficiency Ratio (accuracy / tokens).

4.2 Expected Outcomes

If ATTS hypothesis is valid, we expect:

- 30-50% token reduction vs. always-Deep baseline

- Accuracy within 5% of always-Deep baseline
- Difficulty estimator achieving ~80%+ classification accuracy

These results would provide strong evidence for the viability of the full architecture.

5. Challenges & Limitations

We acknowledge several significant challenges:

1. **Difficulty Estimation Accuracy:** Misclassification wastes compute (easy → Deep) or sacrifices accuracy (hard → Direct). Escalation mitigates but doesn't eliminate this.
2. **Distillation Fidelity:** If synthetic critiques are shallow, distilled models internalize these limitations. Data quality is paramount.
3. **Novel Problem Adaptability:** Highly efficient distilled models may struggle with truly out-of-distribution problems that would benefit from open-ended exploration.
4. **Evaluation Complexity:** Final-answer accuracy alone doesn't capture reasoning quality. New metrics for soundness and robustness are needed.
5. **Training Compute:** Full implementation requires substantial resources for dataset generation and model training. This proposal is currently theoretical.

6. Conclusion & Future Directions

This proposal presents a theoretical framework for achieving efficient, self-verifiable AI reasoning. The core insight is that the "token tax" of advanced reasoning can be substantially reduced through: (1) unified verification across domains, (2) adaptive compute allocation based on problem difficulty, and (3) distillation of dialectical reasoning patterns into efficient representations.

Future work directions include:

1. Implementing and validating the simulation experiment (Section 4)
2. Developing better difficulty estimation through uncertainty quantification
3. Extending USVA to multimodal reasoning tasks
4. Exploring continuous (rather than discrete) compute allocation
5. Creating comprehensive evaluation benchmarks for efficient reasoning

The proposed architecture represents a *feasible research direction* based on logical extensions of current AI capabilities. We invite feedback, collaboration, and experimental validation from the research community.

References

[1] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv:2004.05150.

[2] Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv:1904.10509.

[3] Graves, A. (2016). Adaptive computation time for recurrent neural networks. arXiv:1603.08983.

[4] Lightman, H., et al. (2023). Let's verify step by step. arXiv:2305.20050.

[5] Madaan, A., et al. (2023). Self-refine: Iterative refinement with self-feedback. NeurIPS 36.

[6] Schuster, M., et al. (2021). Confident adaptive language modeling. arXiv:2207.07061.

[7] Zelikman, E., et al. (2022). STaR: Bootstrapping reasoning with reasoning. NeurIPS 35.

Appendix A: ATTS Workflow Diagram Description

The ATTS system operates as follows:

Input: Problem P enters the system.

Stage 1 - Difficulty Estimation: Lightweight classifier produces $d \in [0,1]$.

Stage 2 - Mode Selection: Based on d, route to Direct ($d<0.3$), Thinking ($0.3 \leq d<0.7$), or Deep ($d \geq 0.7$).

Stage 3 - Solution Generation: Execute selected mode.

Stage 4 - Self-Verification: USVA produces verification score v.

Stage 5 - Escalation Check: If $v < \tau$ and not already in Deep mode, escalate to next tier and repeat from Stage 3.

Output: Final solution with verification score.