Evaluating the Reliability-Performance Trade-offs in Large Language Model Caching Systems

A Comprehensive Analysis of Semantic Caching Strategies and Their Impact on Model Correctness

Authors: Eldor Mouyal, Shaden Simaan

Repository: https://github.com/EldorMouyal/GPTCache_PlatinumBenchmarks

---

Abstract

This study presents a critical evaluation of semantic caching strategies in Large Language Model (LLM) applications, revealing significant reliability concerns that challenge conventional assumptions about cache effectiveness. Through comprehensive benchmarking across multiple embedding models and similarity thresholds, we demonstrate that semantic caching can drastically reduce model correctness (from 93% to 59% in worst cases) while providing only modest latency improvements. Our findings establish that the reliability-performance trade-off in LLM caching is far more severe than previously understood, with bad cache hit rates reaching up to 56% across different embedding strategies, even under relatively tight similarity thresholds (0.9-0.95). We define “Bad Cache Hits” as instances where the cache successfully matches a query but returns an incorrect response, directly harming model reliability despite improving latency - a critical failure mode that existing cache evaluation frameworks overlook.

---

1. Introduction

Problem Statement

Large Language Model applications increasingly rely on caching mechanisms to reduce computational costs and improve response times. However, the fundamental assumption that caching preserves response quality while improving performance has not been rigorously tested across diverse workloads and embedding strategies. This study addresses a critical gap: **How do different approximate matching strategies affect the correctness and reliability of cached LLM responses?**

Related Work

Existing LLM caching solutions, particularly those built on GPTCache and similar frameworks, primarily focus on performance optimization through semantic similarity matching using embedding models. Current implementations typically employ:

- **Exact matching**: String-based cache lookups with perfect precision but limited coverage
- **Semantic similarity**: Embedding-based approximate matching using models like BGE-small, E5-base, MPNet, and ONNX variants
- **Configurable thresholds**: Similarity cutoffs (typically 0.8-0.95) to balance hit rates and accuracy

However, systematic evaluation of cache reliability across different embedding strategies and datasets remains limited, creating a significant knowledge gap about the true cost-benefit tradeoffs in production LLM systems.

---

2. Extension Design

Motivation

Rather than implementing a new cache policy, this project addresses a more fundamental question: **Should we be using semantic caching at all?** The motivation stems from observing unexpectedly poor response quality in cached LLM applications, leading to the hypothesis that semantic similarity does not reliably predict response correctness.

Technical Architecture

We developed a comprehensive benchmarking framework that enables systematic evaluation of cache reliability across multiple dimensions:

Framework Components:

- **Dynamic Cache Strategy Loading**: Pluggable architecture supporting none, exact matching, and various semantic similarity approaches, and custom cache strategies that can be added without code modifications
- **Multi-Dataset Evaluation**: Integration with PlatinumBench subsets covering diverse benchmark types including mathematical reasoning (GSM8K), multi-hop question answering (HotpotQA), single-step reasoning (SingleQ), logical reasoning, code generation, and other specialized domains to comprehensively test caching reliability across different reasoning patterns
- **Comprehensive Metrics**: Beyond traditional hit rates and latency, we measure correctness, bad cache hit rates, reliability degradation, and cache effectiveness to capture the full impact of caching on model performance
- **GPU Acceleration**: Seamless Kaggle integration enabling 10-20x speedup with free Tesla T4 GPUs for scalable experimentation

- **Reproducible Benchmarks**: Complete Docker Containerization and CI integration ensuring consistent, reproducible results across different environments.

Key Innovation: Bad Cache Hit Detection Our framework introduces the critical concept of "bad cache hits" - instances where semantic similarity matching successfully retrieves a cached response (improving latency) but delivers an incorrect answer, directly compromising model reliability. This metric exposes a fundamental blind spot in traditional cache evaluation: while conventional benchmarks celebrate high hit rates and reduced latency, they fail to measure whether cached responses maintain correctness. By tracking bad cache hits alongside standard performance metrics, our framework reveals the true cost-benefit trade-off in semantic caching systems and challenges the assumption that cache hits are inherently beneficial.

Cache Strategies Evaluated

While we conducted extensive experiments across multiple LLM models and various similarity thresholds, we present results from the most substantial and revealing configuration: Gemma2:9b, which demonstrated high baseline correctness (93%) without caching, making reliability degradation clearly measurable.

1. **None (Baseline)**: No caching to establish ground truth performance and correctness
2. **Exact Matching**: Traditional string-based caching for identical queries only
3. **Semantic Approximate**: Four embedding models (BGE-small, E5-base, MPNet, ONNX) evaluated at two relatively tight similarity thresholds (0.9, 0.95)

Our focus on tight similarity thresholds (0.9-0.95) makes our findings particularly concerning: even under these conservative matching criteria, designed to minimize false positives, we observe dramatic reliability degradation across all embedding models. This suggests that the fundamental problem lies not in overly permissive threshold settings, but in the inadequacy of semantic similarity as a predictor of response correctness in reasoning tasks.

---

3. Experimental Setup

Workloads and Datasets

PlatinumBench Framework: PlatinumBench provides evaluation datasets spanning mathematical reasoning, logical reasoning, multi-hop question answering, and code generation tasks, enabling systematic evaluation across different cognitive capabilities.

Ollama Integration: Ollama serves as our local LLM deployment platform, ensuring reproducible experimental conditions by eliminating variability from remote API services and network latency.

Configuration: The results presented in this study were obtained using Gemma2:9b via Ollama across five PlatinumBench subsets: GSM8K, HotpotQA, SingleQ, MMLU-Math, and

Winograd WSC. Each experiment used the same 40 questions to ensure consistent comparison conditions.

Metrics Framework

Performance Metrics:

- **latency_mean_sec**: Average response time per query
- **latency_p95_sec:** 95th percentile latency (worst-case performance)
- **throughput_qps:** Queries processed per second

Cache Effectiveness:

- **cache_hit_rate**: Percentage of queries served from cache
- **bad_cache_hit_rate:** Percentage of cache hits returning incorrect responses
- **cache_accuracy:** Ratio of correct cache hits to total cache hits
- **cache_effectiveness:** Overall performance improvement metric

Quality Metrics:

- **correctness:** Percentage of responses matching expected answers
- **reliability_degradation:** Quality impact from caching
- **correctness_without_bad_hits:** Correctness excluding bad cache hits

Experimental Parameters

Model Configuration: - **Temperature**: 0.0 (deterministic responses) - **Max Tokens**: 256 (sufficient for reasoning tasks) **Cache Strategies:** - **Embedding Models**: BGE-small, E5-base, MPNet, ONNX - **Similarity Thresholds**: 0.9, 0.95 - **Query Set**: 40 questions from each of the 5 datasets (200 total questions per experiment)

---

4. Results

Key Findings Summary

Our experiments reveal alarming reliability issues with semantic caching across all tested configurations:

Cache Strategy	Correctness	Bad Cache Hit Rate	Cache Hit Rate	Latency (sec)
None (Baseline)	**93.0%**	**0.0%**	**0.0%**	**4.47**
E5-base (0.9)	59.0%	56.2%	47.2%	1.08
BGE-small (0.9)	79.6%	50.6%	31.0%	3.70
MPNet (0.9)	83.5%	47.5%	25.8%	4.24
ONNX (0.9)	86.0%	43.9%	23.5%	4.22

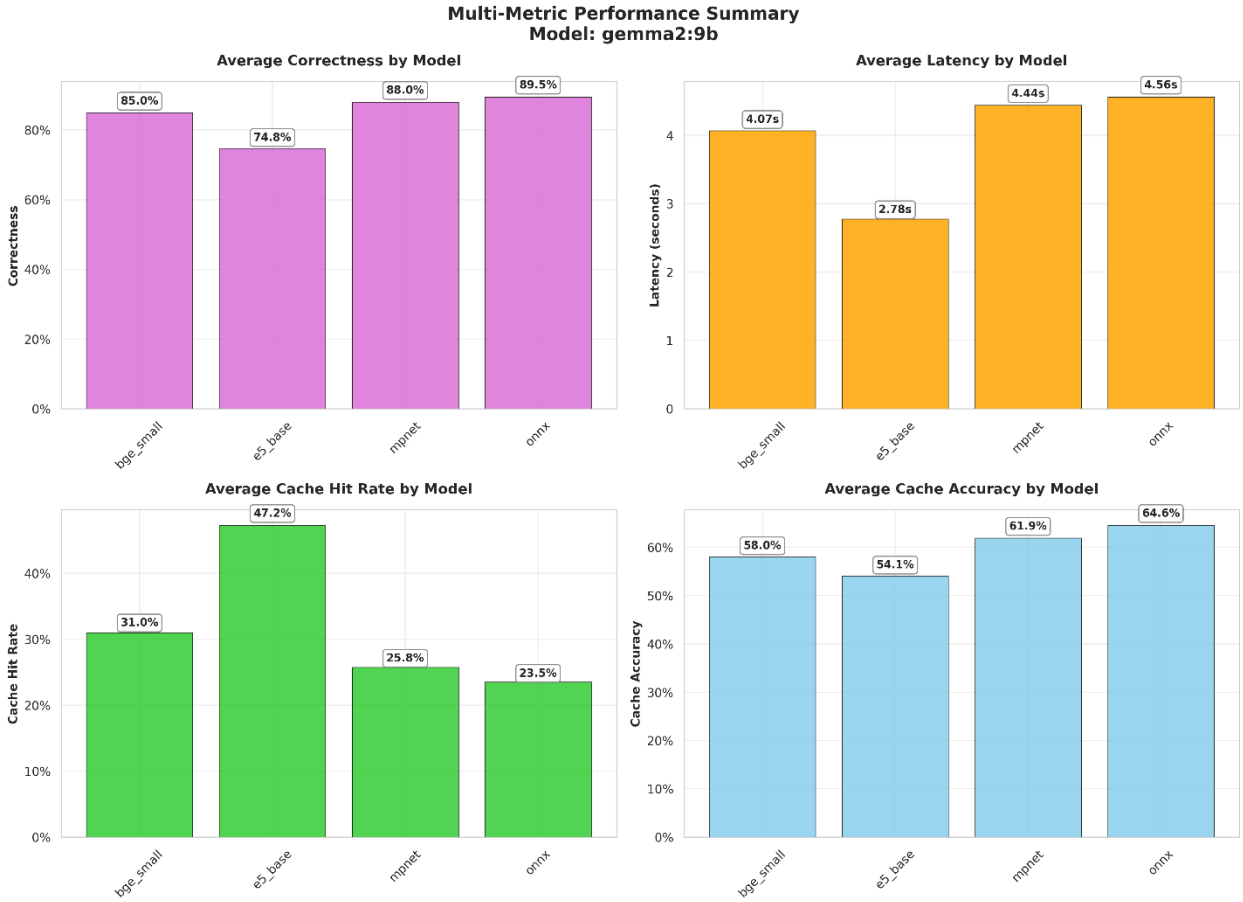

Figure 1 illustrates the comprehensive performance comparison across embedding models, clearly showing the inverse relationship between cache hit rates and correctness.

Critical Performance Analysis

Correctness Degradation:

- Worst case: E5-base reduces correctness from 93% to 59% (34 percentage point drop)
- Best case: ONNX maintains 86% correctness (still 7 percentage point reduction)
- **No semantic caching strategy preserved baseline correctness**

Bad Cache Hit Rates:

- E5-base: 56.2% of cache hits return incorrect responses
- BGE-small: 50.6% bad cache hit rate despite lower overall hit rate
- Even the best performer (ONNX) has 43.9% bad cache hits

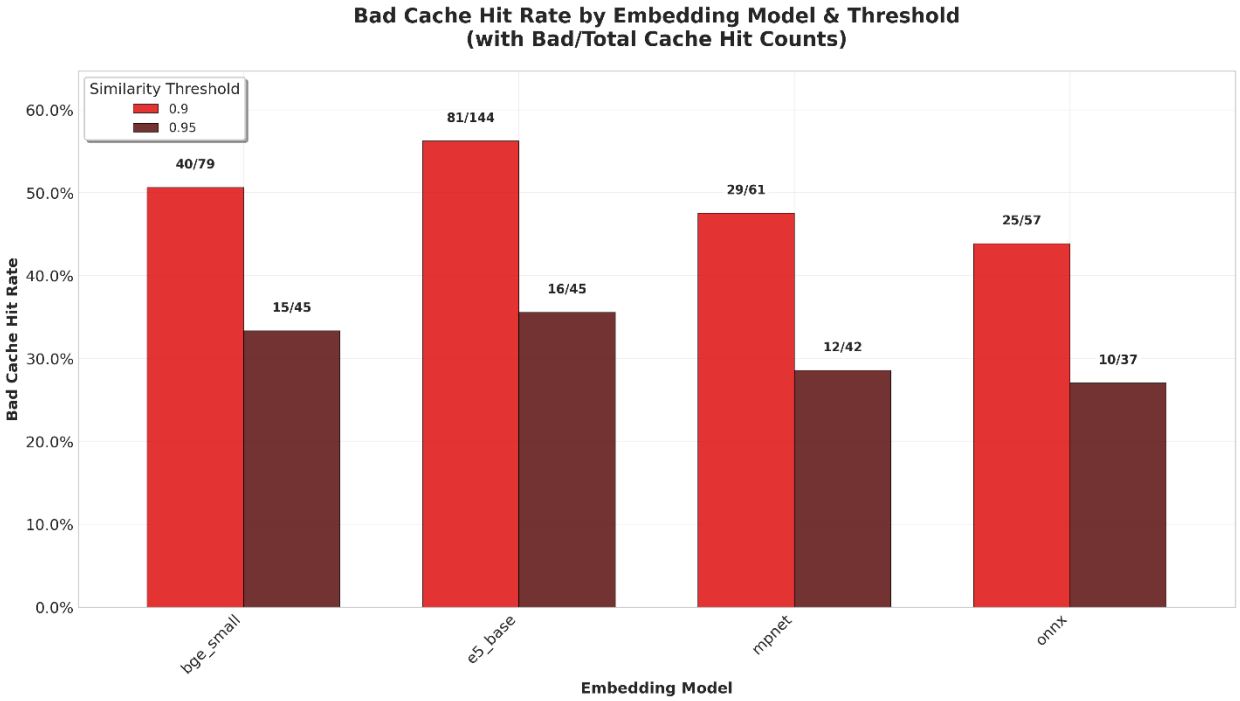

Figure 2 demonstrates how bad cache hit rates remain problematically high even with tighter similarity thresholds across all embedding models.

Latency vs. Correctness Trade-offs: - Most significant latency improvement: E5-base (4.47s → 1.08s, 76% reduction) - However, this comes with the worst correctness penalty (34% degradation) - More conservative approaches (ONNX, MPNet) offer minimal latency benefits (5-6% improvement) while still sacrificing accuracy

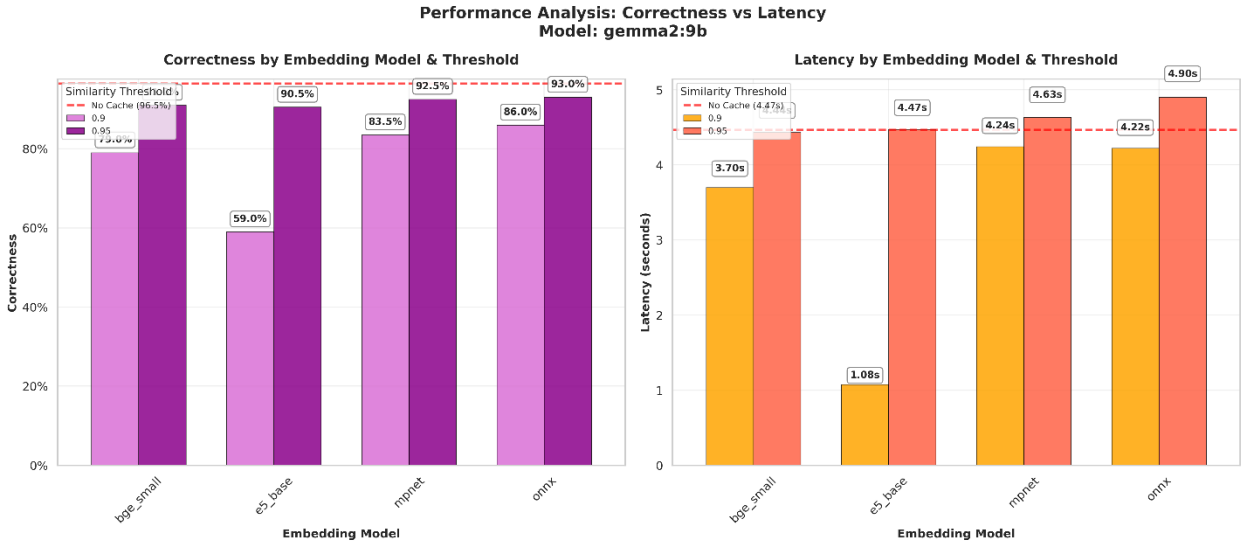

Figure 3 visualizes the fundamental trade-off between latency improvements and correctness degradation, highlighting that performance gains come at severe reliability costs.

Threshold Sensitivity Analysis

Increasing similarity thresholds from 0.9 to 0.95 shows mixed results:

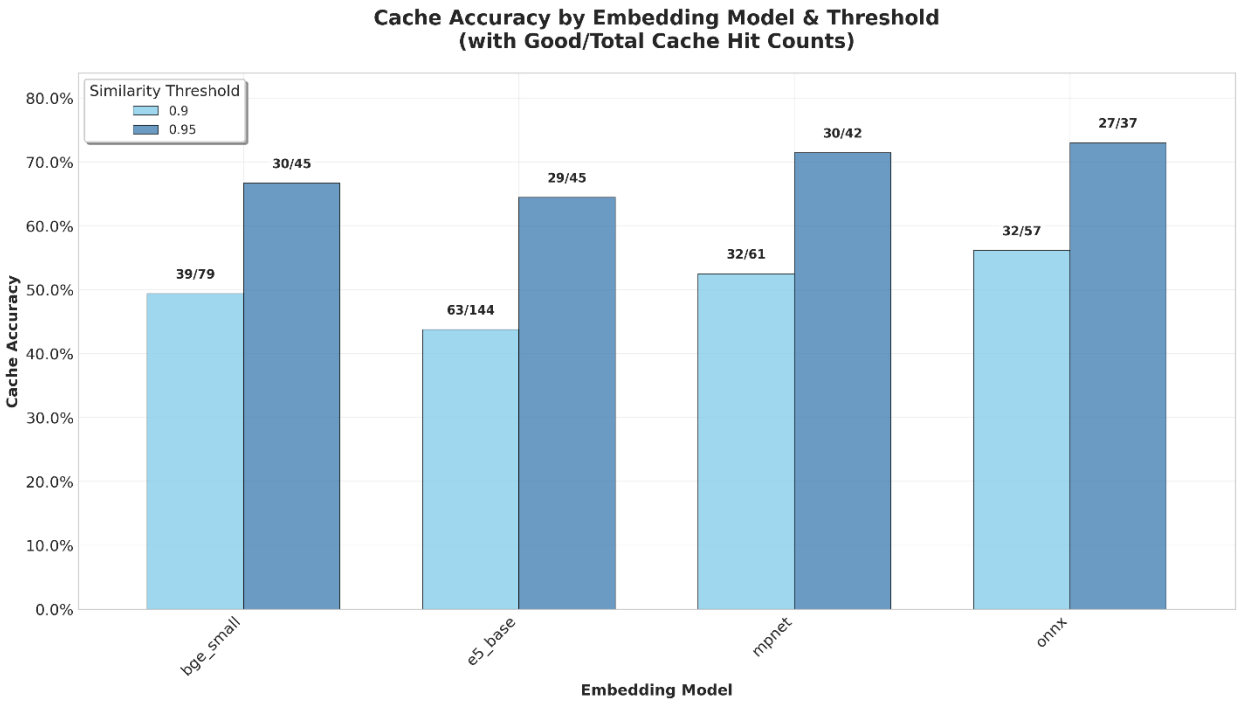

Figure 4 shows how higher similarity thresholds improve cache accuracy but fail to resolve the fundamental reliability issues.

Improvements:

- Bad cache hit rates generally decrease (E5-base: 56.2% → 35.6%)
- Cache accuracy increases across all models

Trade-offs:

- Cache hit rates decrease substantially (E5-base: 47.2% → 18.0%)
- Latency benefits diminish significantly - **Correctness improvements are modest and inconsistent**

Experimental Consistency

Results demonstrate consistent patterns across identical experimental configurations using the same 200-question dataset (40 questions per subset). The deterministic setup (temperature=0.0, consistent query sets) ensures that the observed reliability degradation represents systematic issues with semantic caching rather than random variation.

5. Discussion

Analysis of Trade-offs

The Fundamental Problem: Our results reveal that semantic similarity, as measured by current embedding models, is a poor predictor of response correctness. This finding challenges the basic assumption underlying semantic caching systems.

Embedding Model Performance:

- **E5-base**: Highest cache hit rate but worst reliability - indicates overly permissive similarity matching
- **BGE-small**: Moderate hit rate with severe reliability issues - suggests embedding space doesn't align well with reasoning correctness
- **ONNX**: Best correctness preservation but minimal performance gains - questions the value proposition of approximate caching

Threshold Sensitivity: Higher similarity thresholds (0.95 vs 0.9) improve cache quality but dramatically reduce cache utilization. The resulting performance gains become negligible while correctness issues persist.

Implications for Production Systems

Risk Assessment: For applications where correctness is critical (educational tools, financial advice, medical information), our findings suggest semantic caching poses unacceptable risks. Bad cache hit rates of 40-56% mean nearly half of cached responses may be incorrect.

Cost-Benefit Analysis:

- Latency improvements: 5-76% reduction in response time
- Correctness costs: 7-34% reduction in accuracy - **The reliability cost far outweighs performance benefits in most scenarios**

Parameter Sensitivity

Our analysis reveals that cache performance is highly sensitive to:

- **Embedding model choice**: 25-point correctness variation across models
- **Similarity thresholds**: Dramatic impact on hit rates vs. accuracy balance
- **Dataset characteristics**: Mathematical reasoning (e.g. GSM8K) vs. factual queries show different error patterns

---

6. Conclusion & Future Work

Primary Contributions

1. **Established the reliability crisis in semantic caching**: First comprehensive study demonstrating that semantic caching can reduce model correctness by up to 34 percentage points
2. **Introduced bad cache hit rate metrics**: Critical measurement framework that reveals hidden costs of approximate caching strategies
3. **Comprehensive embedding model evaluation**: Systematic comparison of four popular embedding approaches showing consistent reliability issues across all variants
4. **Reproducible benchmarking framework**: Open-source tooling enabling continued research into cache reliability (GitHub repository)

Key Insights

For Practitioners:

- Semantic caching poses significant reliability risks that may outweigh performance benefits
- Current embedding models are insufficient for reliable semantic matching in reasoning tasks
- Exact matching or no-cache strategies may be preferable for accuracy-critical applications

For Researchers: - Need for embedding models specifically trained for semantic equivalence in reasoning contexts - Opportunity to develop correctness-aware caching strategies - Importance of reliability metrics in cache evaluation frameworks

Future Work

Immediate Extensions:

1. Evaluation on larger, more diverse datasets (MMLU, HellaSwag, etc.)
2. Analysis of domain-specific embedding models trained for reasoning tasks
3. Development of hybrid caching strategies that balance exact and approximate matching

Research Directions:

1. **Correctness-aware similarity metrics**: Embedding models trained specifically to predict response correctness rather than semantic similarity
2. **Dynamic threshold adjustment**: Adaptive systems that adjust similarity thresholds based on query complexity and confidence scores
3. **Multi-stage caching**: Hierarchical approaches combining exact matching, high-confidence approximate matching, and fallback strategies

Final Recommendations

Based on our findings, we recommend:

1. **Immediate**: Implement bad cache hit rate monitoring in all production semantic caching systems
2. **Short-term**: Consider disabling semantic caching for accuracy-critical applications
3. **Long-term**: Invest in developing correctness-aware caching strategies and better evaluation frameworks

This study demonstrates that the current state of semantic caching in LLM applications requires fundamental reconsideration. The trade-off between performance and reliability is far more severe than previously understood, demanding new approaches that prioritize correctness alongside efficiency.

References

- GPTCache Framework: https://github.com/zilliztech/GPTCache
- PlatinumBench Dataset: https://huggingface.co/datasets/platinum-bench
- Project Repository: https://github.com/EldorMouyal/GPTCache_PlatinumBenchmarks
- Ollama Local LLM Serving: https://ollama.com

Appendix

Reproducibility

Code Repository: All experimental code, configurations, and analysis scripts are available at https://github.com/EldorMouyal/GPTCache_PlatinumBenchmarks

Docker Environment:

View README.md under :https://github.com/EldorMouyal/GPTCache_PlatinumBenchmarks

Configuration Files: All experimental parameters defined in YAML configurations under experiments/ directory

Data Availability: Summary results and detailed metrics available in results/tables/summary.csv and individual experiment JSON files