# Performance Pareto for Reasoning Models

## Introduction

Reasoning strategies are shown to increase the capabilities of Large Language Models (LLMs). However, using these strategies often comes with a massive computational overhead, since they often generate verbose outputs, which are not always necessary for correctness. In deployment scenarios where latency and energy matter, such verbose reasoning may become impractical, motivating the need to identify the optimal balance. A central question of this project is how to best allocate compute: scaling up model parameters or extending reasoning with longer outputs instead. For this study, we are going to analyze the Pareto frontier between inference FLOPs and task performance. This analysis can reveal where efficiency is maximized and where do diminishing returns appear.

The motivation behind this project is to guide efficient deployment of reasoning-capable LLMs in real-world scenarios. The current computational overhead of reasoning strategies limits scalability. By mapping out the compute-performance Pareto frontier, this work highlights where additional compute is truly justified and where diminishing returns set in. Understanding whether to allocate resources toward larger base models or toward smaller models with longer reasoning helps practitioners choose the most cost-effective setup for a given application. Moreover, in latency- and energy-constrained settings (e.g., edge devices, mobile assistants), verbose reasoning may be impractical. The Pareto analysis identifies reasoning strategies that maximize performance without overspending compute.

By quantifying trade-offs between parameters, tokens, and reasoning depth, the study provides evidence on how reasoning interacts with scale – whether reasoning is a substitute for more parameters, a complement, or sometimes even a liability. Additionally, different domains (math, code, commonsense reasoning) benefit differently from longer reasoning or multi-pass strategies. This project can inform when and where to apply each method most effectively.

## Testable Hypotheses

**Compute allocation.** A key hypothesis is that small models with a reasoning strategy can approach, or even match the performance of larger models that do not use explicit reasoning but have higher accuracy in general. This raises a question about the best choice of model sizes and reasoning strategies given a fixed compute cost.

**Diminishing returns of long Chain-of-Thought (CoT).** Extending CoT reasoning tends to improve performance initially, but beyond a certain point, the additional tokens no longer yield proportional accuracy gains. In some cases, excessively long reasoning may even introduce noise, redundancy, or spurious correlations that reduce accuracy.

**Parameter saturation.** Larger models can often reach the same accuracy as smaller models with much shorter reasoning traces. This suggests a saturation effect: once parameters are sufficiently scaled, long reasoning chains add little value, and compute efficiency shifts toward shorter inference paths with fewer attention-intensive operations.

**Multi-pass methods (SC/Deliberate).** Techniques like self-consistency or deliberate multi-pass reflection scale almost linearly with compute, since they require multiple generations per query. While this can be expensive, the benefits may be highly task-dependent. For example, domains such as math word problems or logical puzzles may see significant accuracy boosts, while simpler factual tasks may not justify the extra overhead.

## Project Outline

Stage 1: Build a minimal viable product (MVP) for our evaluation framework. This involves setting up robust logging and measurement tools for both accuracy and compute cost (in FLOPs). The emphasis is not on testing reasoning strategies, but rather on ensuring that we can reliably capture all relevant metrics. Stage 1 will also involve setting up and validating the inference environment (e.g., VLLM) to ensure that FLOP and latency estimates are consistent across runs. The MVP should establish a clear structure for the project.

Stage 2: Expand into running baseline evaluations across a set of models and benchmarks. This serves as an intermediate step, since it does not yet require advanced reasoning strategies (such as SC or Deliberate multi-pass) to be implemented. Instead, we can experiment with prompts and CoT setups while paying attention to the FLOP metric and model accuracy.

Stage 3: Introduce more advanced reasoning methods, including SC and deliberate multi-pass strategies. At this point the logging and visualization pipeline will already be in place.

Stage 4: Consolidate results into a comprehensive analysis of the compute-performance Pareto frontier. Identify Pareto-optimal strategies, map out diminishing returns, highlight task-dependent benefits of different reasoning approaches.

## Parameters + Metrics to Record

* **Model/setup:** model, arch, params\_B, layers, d\_model, heads, precision, quant, hardware, batch\_size, use\_kv\_cache, reasoning\_style
* **Tokens:** prompt\_tokens, gen\_tokens, passes, beam\_width, self\_consistency\_k
* **Measured:** latency\_ms, speed\_tok\_per\_s, energy\_j (optional)
* **Task metric:** dataset, metric\_name, accuracy
* **Notes:** Observations such as anomalies, special conditions or qualitative differences in model behavior.

## Plots

* **Main:** scatter of **FLOPs (log)** vs **accuracy**, with **Pareto frontier** overlay (non-dominated: min FLOPs, max accuracy).
* Use **color** for params\_B (or tokens) and **bubble size** for gen\_tokens (or passes).

## FLOPs (inference) — two estimators

1. **Dense-only (attention-length agnostic):**FLOPs≈c⋅params⋅tokens,c≈2\text{FLOPs} \approx c \cdot \text{params} \cdot \text{tokens},\quad c\approx 2
2. **Attention-aware with KV cache (rough but useful):**
   * Prefill (P prompt tokens, L layers, hidden d):Fprefill≈L(8Pd2+2P2d)F\_{\text{prefill}} \approx L\big(8Pd^2 + 2P^2 d\big)
   * Decode (G generated tokens):Fdecode≈L(8Gd2+4d(GP+G(G−1)2))F\_{\text{decode}} \approx L\big(8Gd^2 + 4d(GP + \tfrac{G(G-1)}{2})\big)
   * Plus a small dense anchor ∼params⋅(P+G)\sim \text{params}\cdot(P+G) for calibration.