# Pyramid MoA: A Probabilistic Framework for Cost-Optimized Anytime Inference

**Arindam Khaled**
Independent Researcher
arindamkhaled@gmail.com

April 12, 2026

**Abstract**

Large Language Models (LLMs) face a persistent trade-off between inference cost and reasoning capability. While larger "Oracle" models generally achieve higher accuracy, they are more expensive for high-volume deployment. Smaller, cost-effective models often struggle with complex tasks. The question is not which tier to choose, but how to allocate queries across tiers dynamically. We observe that the emerging practice of LLM cascading and routing implicitly solves an *anytime computation* problem—a class of algorithms, well-studied in classical AI, that produce valid solutions immediately and improve them as additional computation is allocated. In this work, we formalize this connection and propose **"Pyramid MoA"**, a hierarchical Mixture-of-Agents architecture governed by a decision-theoretic router that dynamically escalates queries only when necessary. We establish a *Probabilistic Anytime Property*, proving that expected solution quality is monotonically non-decreasing with computational depth under identifiable conditions on router precision. We derive a generalized escalation rule from Value of Computation theory that accounts for imperfect oracles, extending the classical monitoring framework of Hansen and Zilberstein to stochastic LLM inference. On the MBPP code generation benchmark, the Consensus Router intercepts **81.6%** of bugs. On the GSM8K/MMLU mathematical reasoning benchmark, the system nearly matches the Oracle baseline of **68.1%** accuracy at the near break-even operating point, and achieves **42.9%** compute savings in economy mode with a modest accuracy trade-off (63.2%). Crucially, the router transfers zero-shot to unseen benchmarks: on HumanEval it matches the Oracle baseline of **81.1%** accuracy, and achieves **62.7%** cost savings in economy mode (73.2% accuracy). On the highly complex MATH 500 benchmark, the system preserves the **58.0%** Oracle ceiling and enables up to **59.0%** savings in efficiency mode (40.2% accuracy). We further investigate context-aware escalation—where the Oracle receives Layer 1 outputs as context—and discover a *context-conditioned anchoring effect* consistent across four benchmarks: passing correct SLM reasoning improves Oracle accuracy by up to **19.2 percentage points**, while passing incorrect reasoning degrades it by up to **18.0 percentage points**, revealing a fundamental tension in hierarchical Mixture-of-Agents architectures. The framework adapts its behavior to task difficulty: enabling substantial cost savings on benchmarks where SLMs are strong (up to 42.9% on GSM8K/MMLU) while preserving Oracle-level accuracy on difficult out-of-distribution benchmarks (MATH 500, HumanEval).

## 1 Introduction

The rapid proliferation of Large Language Models (LLMs) has created a persistent tension between inference cost and reasoning capability. Recent Mixture-of-Agents (MoA) approaches [4] demonstrate that layering multiple models—leveraging the principle that an ensemble of weaker learners can outperform any individual model—produces stronger outputs than single-model inference. However, standard MoA executes all layers for every query regardless of difficulty, ignoring the heterogeneous computational needs of different inputs. While emerging works like *Sparse MoA* [6] and *Residual MoA* (RMoA) [7] have begun to explore adaptive termination, they often require architectural modifications or complex internal metrics.

We observe that this challenge—deciding how much computation to allocate to a given query—is an instance of a well-studied problem in classical AI: **anytime computation** [1, 2]. An anytime algorithm produces a valid solution immediately and monotonically improves it as additional computation is allocated. The associated *monitoring problem* [3] asks: when should we stop allocating computation and return the current solution? The LLM routing community has been independently reinventing these concepts—early exit strategies, cascade thresholds, confidence-based routing [8, 11, 9]—without the formal toolkit to analyze them.

We propose **Pyramid MoA**, a framework that explicitly bridges anytime computation theory and multi-model LLM inference. The name reflects the system's routing geometry: all queries enter at the broad base, where they are processed by a cost-effective ensemble of SLMs. A lightweight router then filters this workload, escalating only a narrowing subset of increasingly difficult queries to the expensive Oracle at the apex. This pyramid-shaped workload distribution—high query volume at the base, narrow volume at the top—ensures that heavy computational investment is concentrated exclusively on the tasks that require it most. By recasting the routing decision as an instance of the classical monitoring problem, we obtain formal guarantees and principled design criteria that ad-hoc cascading approaches lack. Our contributions are:

**1. Anytime Inference Framework:** We formalize multi-model LLM routing as a probabilistic anytime computation problem. Classical anytime algorithms guarantee per-instance monotonic improvement; we establish a *Probabilistic Anytime Property* (Theorem 1) showing that expected solution quality is monotonically non-decreasing with computational depth under identifiable conditions on router precision. We introduce *performance profiles* adapted from the anytime literature that characterize the cost-quality trade-off and diagnose dataset difficulty.

**2. Generalized Decision-Theoretic Router:** We derive an optimal escalation rule from Value of Computation theory, generalizing the classical monitoring framework to handle stochastic, imperfect oracles. Unlike prior formulations that assume near-perfect Oracle accuracy, our generalized decision rule (Equation 5) explicitly accounts for Oracle error, revealing two distinct barriers to escalation: the cost ratio and Oracle imperfection. The resulting router is lightweight, model-agnostic, and compatible with black-box APIs.

**3. Empirical Dynamic Range & Context-Aware Analysis:** We demonstrate that the framework adapts its behavior to task difficulty—enabling substantial cost savings on benchmarks where SLMs are strong while preserving Oracle-level accuracy on difficult out-of-distribution benchmarks—and transfers zero-shot to unseen benchmarks (HumanEval, MATH 500), validating the theoretical predictions across four diverse evaluations. We further reveal a context-conditioned anchoring effect in hierarchical MoA, consistent across four benchmarks and two task domains: passing incorrect SLM reasoning to the Oracle degrades its accuracy by 14.9 to 18.0 percentage points, validating the routing-based design as a necessary safeguard.

# 2 Methodology

## 2.1 From Anytime Search to Anytime Inference

In classical AI, an *anytime algorithm* [1, 2] produces a valid solution immediately and monotonically improves it as additional computation is allocated. This property is formalized through a *performance profile*—a mapping from computation time to solution quality—and a *monitoring problem*—determining when the marginal gain from further computation no longer justifies its cost [3].

We observe that the emerging practice of LLM cascading and routing (e.g., FrugalGPT [8], RouteLLM [11]) implicitly solves an anytime computation problem. A small model produces an initial answer (valid but potentially suboptimal), and a routing decision determines whether to allocate additional computation via a larger model. However, existing approaches lack the formal framework to analyze this trade-off rigorously. They typically rely on ad-hoc confidence thresholds without characterizing the conditions under which escalation provably improves outcomes.

We bridge this gap by recasting multi-model LLM inference as a **probabilistic anytime** computation problem. The key departure from the classical setting is that LLM inference is inherently stochastic: unlike deterministic search, where more computation guarantees monotonic improvement, a larger model may occasionally produce a worse answer than a smaller one on any individual query. Our framework addresses this by establishing monotonicity guarantees *in expectation* over query distributions rather than per-instance.

To make the connection precise, we establish the following correspondence:

Table 1: Mapping from classical anytime computation to Pyramid MoA.

| **Anytime Computation** | **Pyramid MoA** |
|---|---|
| Initial solution | Layer 1 (SLM ensemble) output |
| Extended computation | Escalation to higher-capability layers |
| Computation time axis | Model capacity / inference cost |
| Performance profile | Accuracy as a function of cost |
| Monitoring problem | Router's escalation decision at each layer |
| Quality guarantee (deterministic) | Quality guarantee (probabilistic) |

This mapping is not merely analogical. In Section 2.3, we show that under identifiable conditions on the router's precision, Pyramid MoA satisfies a formal probabilistic analogue of the classical anytime monotonicity property. In Section 2.4, we derive the optimal monitoring policy from the decision-theoretic framework, generalizing the classical Value of Computation to handle stochastic, imperfect oracles.

## 2.2 System Components

Our architecture consists of three components:

1. **Layer 1 (The Crowd):** An ensemble of cost-effective models: Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Gemma-2-9B.
2. **The Router:** A trained classifier predicting the probability of Layer 1 failure ($P_{fail}$).
3. **Layer 2 (The Oracle):** Llama-3.3-70B-Instruct, invoked only when $P_{fail} > t$, where $t$ is a tunable threshold.

**Note on Nomenclature (Routing vs. Context-Aware):** Unlike context-aware MoA approaches that synthesize a new response from peer outputs [4], our framework employs a *Routing-Based MoA*. The ensemble's collective signal is used to calibrate the escalation decision rather than to generate the output token sequence. The system components are strictly modular and API-compatible: any black-box model can serve as Layer 1 or Layer 2 without architectural modification, differentiating our approach from methods like RMoA [7] that require internal model access. We discuss extensions to context-aware Pyramid MoA—where the Oracle receives Layer 1 outputs as context for refinement—in Section 5.

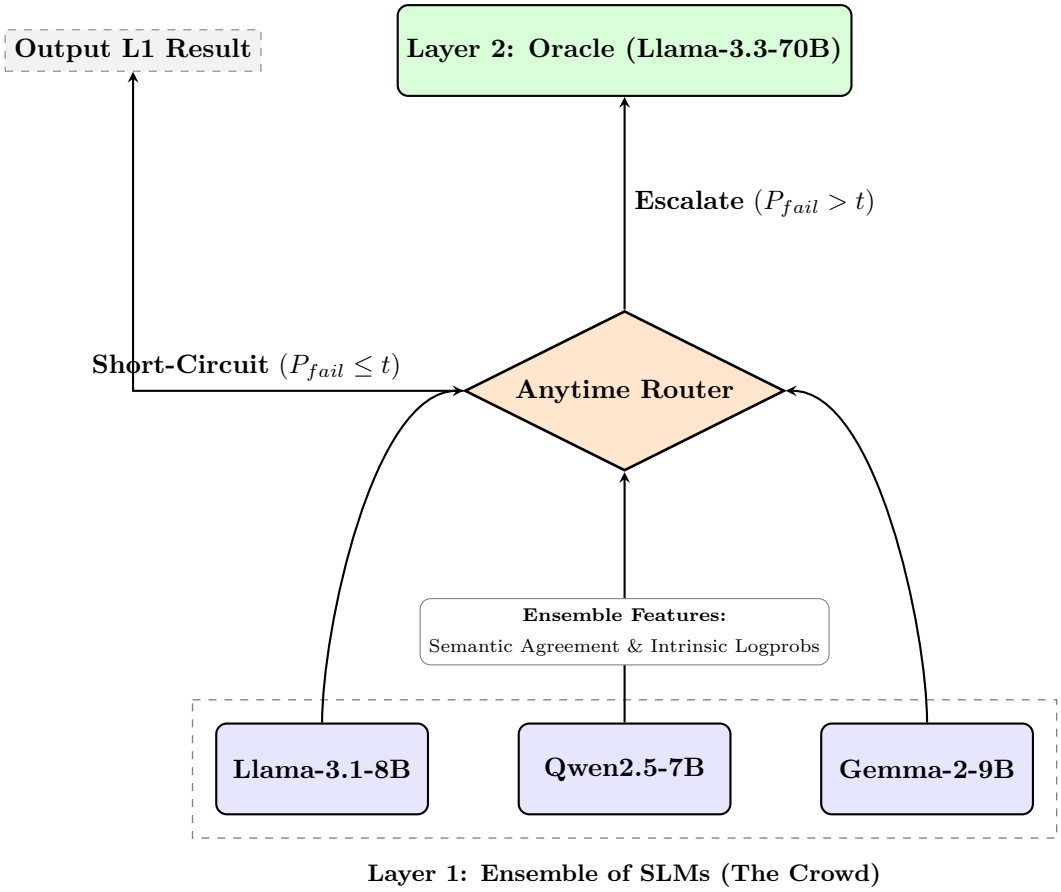



Figure 1: **Pyramid MoA Architecture:** The system extracts ensemble-wide features from the Layer 1 models to estimate $P_{fail}$. The router solves the anytime monitoring problem—deciding whether to allocate additional computation via the Oracle.

## 2.3 The Probabilistic Anytime Property

We now formalize the sense in which Pyramid MoA exhibits anytime behavior.

**Definition 1** (Deterministic Anytime Property)**.** *An algorithm $A$ satisfies the* anytime property *if, for any input $x$ and computation budgets $c_1 < c_2$, the solution quality satisfies $Q(A, x, c_2) \geq Q(A, x, c_1)$.*

This is the classical definition, originating with Dean and Boddy [1] and formalized by Zilberstein [2]. It requires per-instance monotonicity, which cannot hold in stochastic LLM inference—a 70B model can produce an incorrect answer on a query that an 8B model answers correctly. We therefore relax the requirement:

**Definition 2** (Probabilistic Anytime Property)**.** *A system $S$ satisfies the* probabilistic anytime property *with respect to a query distribution $\mathcal{D}$ if, for computational depths $d_1 < d_2$:*

$$\mathbb{E}_{x\sim\mathcal{D}}\big[Q(S, x, d_2)\big] \geq \mathbb{E}_{x\sim\mathcal{D}}\big[Q(S, x, d_1)\big] \tag{1}$$

**Theorem 1** (Monotonicity Condition)**.** *Let $R$ denote the set of queries escalated by the router (i.e., those with $P_{fail} > t$), and let $\bar{R}$ denote the queries retained at Layer 1. Define $\alpha_{L1}(R)$ and $\alpha_{L2}(R)$ as the accuracy of Layer 1 and Layer 2 on the escalated subset, and $p_R = |R|/N$ as the escalation rate. The Pyramid MoA system satisfies the probabilistic anytime property if and only if:*

$$\alpha_{L2}(R) \geq \alpha_{L1}(R) \tag{2}$$

*Proof.* The accuracy of the full system is $\text{Acc}_{\text{MoA}} = (1 - p_R) \cdot \alpha_{L1}(\bar{R}) + p_R \cdot \alpha_{L2}(R)$, while Layer 1 alone achieves $\text{Acc}_{L1} = (1 - p_R) \cdot \alpha_{L1}(\bar{R}) + p_R \cdot \alpha_{L1}(R)$. The improvement is therefore:

$$\text{Acc}_{\text{MoA}} - \text{Acc}_{L1} = p_R \cdot \big[\alpha_{L2}(R) - \alpha_{L1}(R)\big] \tag{3}$$

Since $p_R > 0$ whenever escalation occurs, this is non-negative if and only if $\alpha_{L2}(R) \geq \alpha_{L1}(R)$. □

**Remark 1** (Router quality amplifies the anytime property). A perfect router that escalates only queries where L1 fails maximizes $\alpha_{L2}(R) - \alpha_{L1}(R)$. A random router reduces the gap to the marginal difference $\alpha_{L2} - \alpha_{L1}$ across the full distribution. We verify this condition empirically in Section 3.3.

### 2.4 Generalized Decision-Theoretic Routing

We formalize the escalation decision as a decision-theoretic problem, generalizing prior formulations to handle the realistic case where the Oracle is imperfect. Let $P_{fail}$ be the router's estimated failure probability, $P_{oracle}$ the Oracle's success probability, $U_{correct}$ the utility of a correct answer, and $C_{esc}$ the escalation cost. The expected utilities of stopping vs. escalating are:

$$\mathbb{E}[U_{\text{stop}}] = (1 - P_{fail}) \cdot U_{correct}, \quad \mathbb{E}[U_{\text{esc}}] = P_{oracle} \cdot U_{correct} - C_{esc} \tag{4}$$

Escalation is optimal when $\mathbb{E}[U_{\text{esc}}] > \mathbb{E}[U_{\text{stop}}]$, yielding:

$$P_{fail} > \underbrace{\frac{C_{esc}}{U_{correct}}}_{\text{cost barrier}} + \underbrace{(1 - P_{oracle})}_{\text{imperfection barrier}} \tag{5}$$

This reveals two distinct barriers to escalation. The **cost barrier** reflects the computational expense, as in standard cascade formulations. The **imperfection barrier** captures the risk that escalation incurs cost without producing a correct answer. When $P_{oracle} \to 1$, the imperfection barrier vanishes and Equation 5 reduces to the standard cascade rule.

In practice, when the utility of a correct answer substantially exceeds the escalation cost ($U_{correct} \gg C_{esc}$), the cost barrier becomes negligible and the imperfection barrier dominates the escalation decision. This distinguishes our framework from purely cost-driven cascades: the primary reason to avoid escalation is not expense but Oracle unreliability.

**Connection to classical monitoring.** Our router solves a single-step version of Hansen and Zilberstein's monitoring problem [3], observing ensemble-level features as a proxy for current solution quality and making a binary stop/escalate decision. This reduction from a sequential to a single-step decision is justified by the two-tier architecture. We discuss extensions to deeper pyramids in Section 5.

### 2.5 Performance Profiles

A *performance profile* [2, 3] maps computational investment to expected solution quality. We define Pyramid MoA's performance profile as $\Pi(t) = \mathbb{E}_{x \sim \mathcal{D}}[Q(S, x) \mid \text{threshold} = t]$. At $t = 1.0$ (never escalate), cost is minimal and accuracy equals $\alpha_{L1}$. At $t = 0.0$ (always escalate), cost is maximal and accuracy approaches $\alpha_{L2}$. A well-calibrated router produces a *concave* profile—accuracy rises steeply at first and flattens with additional spending—enabling favorable operating points where most accuracy gain is captured at a fraction of the full cost. We present empirical performance profiles in Section 3.

## 3 Experimental Results

We evaluate Pyramid MoA across four benchmarks spanning code generation and mathematical reasoning. All experiments use a unified Layer 1 ensemble of Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and

Gemma-2-9B, with Llama-3.3-70B-Instruct as the Layer 2 Oracle. We train two domain-specific routers and evaluate their in-domain performance and zero-shot transfer.

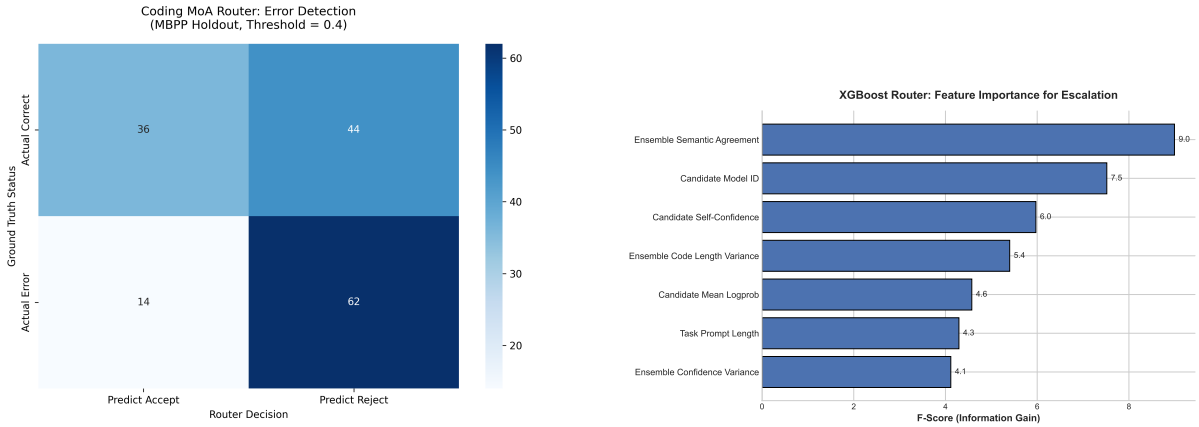
Table 2: Router configurations by domain.

| Feature | Consensus Router | Anytime Router |
|---|---|---|
| **Algorithm** | XGBoost | XGBoost |
| **Training Domain** | MBPP (Code) | GSM8K/MMLU (Math) |
| **Primary Signal** | Semantic Agreement | Token Log-Probabilities |
| **Routing Characteristic** | Error-catching (high recall) | Selective escalation (high precision) |
| **Transfer Evaluation** | HumanEval | MATH 500 |

## 3.1 Experiment I: Code Generation (MBPP)

For code generation tasks, we deployed the **Consensus Router** (XGBoost), prioritizing safety via semantic agreement among the ensemble's outputs.

On the MBPP holdout set ($N = 156$, threshold $t = 0.4$), the router achieved a **Recall of 81.6%**, intercepting 62 out of 76 erroneous code snippets (Figure 2a). Feature importance analysis (Figure 2b) validates the "Consensus" hypothesis: *Self-Reported Confidence* was the least predictive feature, while *Output Length Variance* and *Semantic Agreement* were dominant predictors. This confirms that for open-ended generation tasks, models are often "confident but wrong" [10], necessitating external peer-review signals.




(a) **Confusion Matrix (MBPP,** $t = 0.4$**)**

(b) **Feature Importance (MBPP)**

Figure 2: **Consensus Mechanism:** Evaluation on MBPP showing that peer-agreement signals significantly outperform intrinsic model confidence for error detection.

### 3.1.1 Zero-Shot Transfer: HumanEval

To test generalizability, we applied the MBPP-trained Consensus Router zero-shot to the HumanEval benchmark. As shown in Figure 3, the router successfully transfers: at the "Baseline Match" operating point, the system achieves the full Oracle accuracy of **81.1%** while still requiring substantially fewer Oracle calls than running the 70B model on all queries. In "Economy Mode," the system achieves 73.2% accuracy with **62.7%** compute savings versus the Oracle.

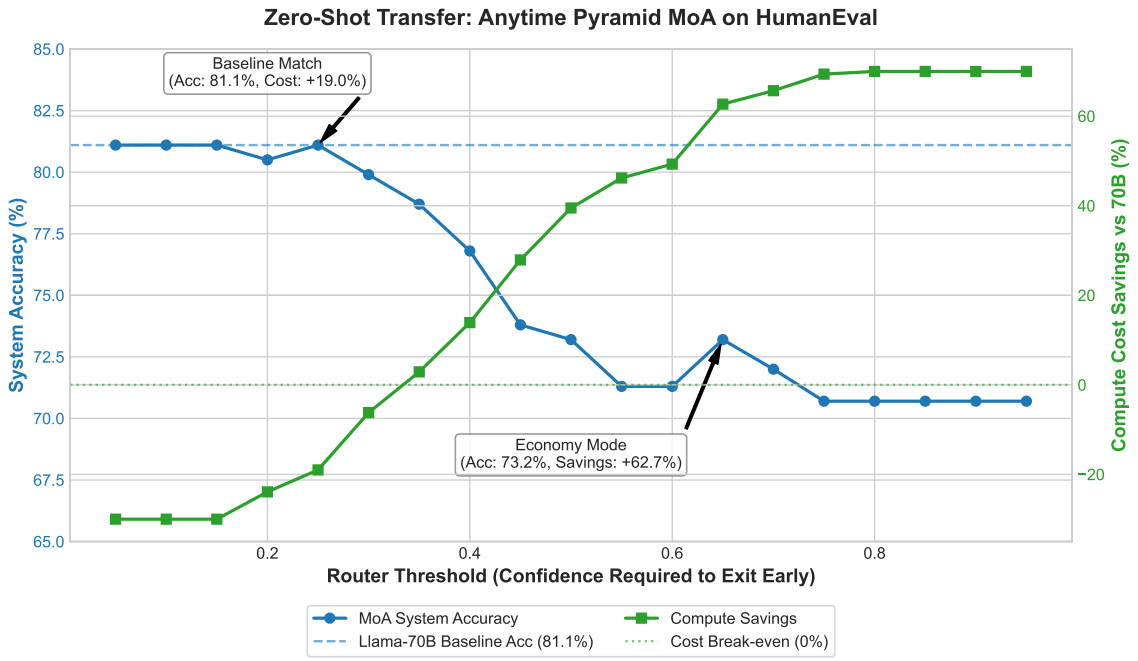



Figure 3: **Zero-Shot Transfer to HumanEval:** The MBPP-trained Consensus Router transfers effectively, achieving the Oracle baseline (81.1%) and enabling up to 62.7% cost savings in economy mode.

## 3.2 Experiment II: Mathematical Reasoning (GSM8K/MMLU)

For convergent mathematical tasks, we deployed the **Anytime Router** (XGBoost), utilizing intrinsic token log-probabilities (*avg_logprob*, *min_prob*) as the primary routing signal.

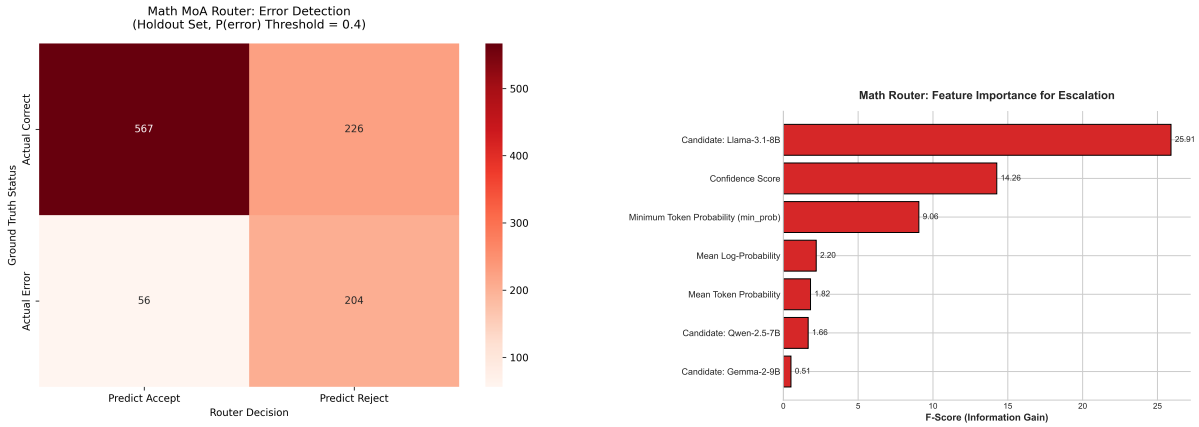
### 3.2.1 Router Analysis

On the GSM8K/MMLU holdout set ($N = 1{,}053$, threshold $t = 0.4$), the router achieved a **Recall of 78.5%** (204 of 260 errors detected); Figure 4a). Feature importance analysis (Figure 4b) reveals a markedly different signal structure from code generation: *Candidate: Llama-3.1-8B* (the primary model's correctness indicator) and *Confidence Score* are the dominant features, with *Minimum Token Probability* (*min_prob*) providing complementary signal. Unlike MBPP, intrinsic confidence is highly predictive for convergent tasks where answers are deterministic.

### 3.2.2 The Anytime Performance Profile

Figure 5 presents the full performance profile on the holdout set, sweeping the threshold from $t = 0.05$ to $t = 0.95$. The Oracle baseline (Llama-3.3-70B) achieves 68.1% accuracy. At the "Near Break-even" operating point ($t \approx 0.15$), the system achieves **67.8%** accuracy with **3.3%** compute savings—nearly matching the Oracle at reduced cost. In "Economy Mode" ($t \approx 0.35$), the system achieves 63.2% accuracy with **42.9%** compute savings. The characteristic concave shape confirms that the router allocates Oracle computation to the highest-value queries first, consistent with the theoretical prediction from Section 2.5.

Notably, a slight accuracy recovery is visible at higher thresholds ($t > 0.3$), where eliminating the remaining escalations marginally *improves* system accuracy. This reflects the imperfection barrier (Equation 5) in practice: at these operating points, a small number of escalations replace correct Layer 1 answers with incorrect Oracle answers. Removing those harmful escalations yields a net accuracy gain, providing direct empirical evidence that Oracle imperfection is not merely a theoretical concern but a measurable effect in the performance profile.




(a) **Confusion Matrix (GSM8K/MMLU, $t = 0.4$)** (b) **Feature Importance (Math Router)**

Figure 4: **Math Router Analysis:** The XGBoost router leverages candidate model correctness and token-level uncertainty signals for escalation decisions on convergent tasks.

#### 3.2.3 Zero-Shot Transfer: MATH 500

To test robustness against severe domain shifts, we evaluated the GSM8K/MMLU-trained router zero-shot on the MATH 500 dataset, which contains AIME-level calculus and algebra problems well outside the training distribution.

As shown in Figure 6, the router successfully detects the distributional shift. At the "Zero-Shot Baseline Match" point ($t \leq 0.25$), the system achieves the full Oracle accuracy of **58.0%**. In "Efficiency Mode," the system achieves 40.2% accuracy with **59.0% compute savings**. This behavior is *predicted* by the generalized decision rule (Equation 5): Layer 1 accuracy is low on this difficult distribution, producing high $P_{fail}$ estimates that trigger frequent escalation. At the same time, with $P_{oracle} \approx 0.58$, the imperfection barrier is substantial ($\approx 0.42$), meaning that even escalated queries face significant Oracle error risk. The router correctly identifies that heavy escalation is needed to preserve accuracy, while the imperfection barrier limits the gains achievable through escalation alone.

### 3.3 Empirical Verification of the Monotonicity Condition

Theorem 1 establishes that the probabilistic anytime property holds if and only if the Oracle outperforms Layer 1 on the router-escalated subset. Table 3 verifies this condition directly across all four benchmarks at a consistent threshold of $t = 0.4$.

Table 3: Verification of the monotonicity condition (Theorem 1) at threshold $t = 0.4$. $\alpha_{L1}(R)$ is computed using majority vote over the Layer 1 ensemble. In all cases, $\alpha_{L2}(R) > \alpha_{L1}(R)$, confirming strict probabilistic anytime improvement.

| **Benchmark** | $p_R$ | $\alpha_{L1}(R)$ | $\alpha_{L2}(R)$ | **Gap** | **ΔAcc** | **Satisfied?** |
|---|---|---|---|---|---|---|
| MBPP (Code) | 0.558 | 0.379 | 0.690 | +0.311 | +17.3% | ✓ |
| HumanEval (Code, OOD) | 0.561 | 0.696 | 0.804 | +0.109 | +6.1% | ✓ |
| GSM8K/MMLU (Math) | 0.271 | 0.632 | 0.768 | +0.137 | +3.7% | ✓ |
| MATH 500 (Math, OOD) | 0.110 | 0.145 | 0.582 | +0.436 | +4.8% | ✓ |

The table reveals how router quality and task difficulty interact. On MBPP, the router achieves the largest gap (+31.1pp) by precisely identifying queries where the ensemble disagrees—queries on which Layer 1 accuracy drops to 37.9% while the Oracle achieves 69.0%. On HumanEval, the MBPP-trained router transfers zero-shot with a +10.9pp gap, demonstrating that the learned consensus signal generalizes across code generation tasks. On GSM8K/MMLU, the corrected router escalates 27.1% of queries—targeting the

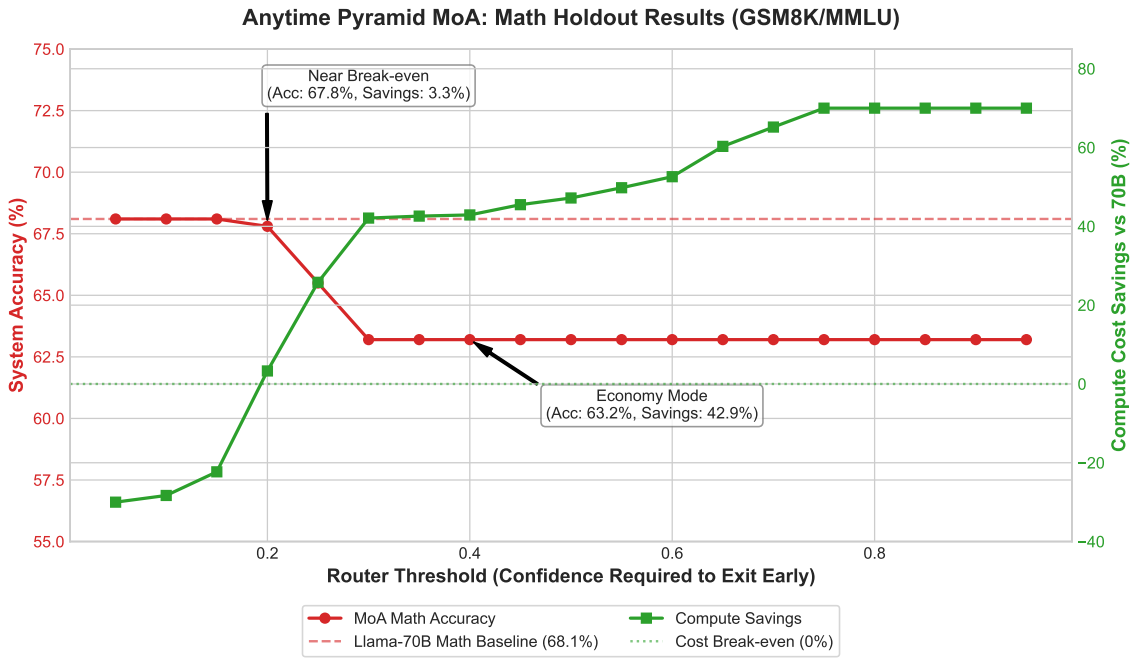



Figure 5: **Anytime Performance Profile (GSM8K/MMLU Holdout):** The dual-axis plot shows accuracy (red) and compute savings (green) as a function of router threshold. The concave accuracy profile confirms efficient allocation of Oracle computation. The monotonically decreasing accuracy curve as the threshold increases (i.e., as less computation is allocated) empirically demonstrates the probabilistic anytime property: expected solution quality is non-decreasing with computational depth.

subset where Layer 1 accuracy is 63.2%, well below the population average—and the Oracle achieves 76.8% on this harder subset, yielding a +13.7pp gap. On MATH 500, the router escalates only 11.0% of queries, selecting the most difficult problems where Layer 1 accuracy is just 14.5%. The Oracle achieves 58.2% on this subset—a +43.6pp gap—demonstrating that the framework concentrates expensive computation precisely where it yields the largest marginal improvement.

## 3.4 Context-Aware Escalation: Anchoring in Hierarchical MoA

The current framework employs *routing-based* mixture: the Oracle generates its answer independently without access to Layer 1 outputs. A natural question is whether passing the SLM ensemble's chain-of-thought reasoning and candidate answers as context to the Oracle improves accuracy on escalated queries—transforming the architecture from routing-based to *context-aware* Pyramid MoA.

We evaluate this across four benchmarks by providing the Oracle with all available Layer 1 outputs (chain-of-thought and final answers from up to three SLMs) alongside the original query, with instructions to review, reconcile, and correct the candidates. Table 4 presents the results segmented by whether the SLM majority answer was correct.

**The anchoring effect is general.** The context-conditioned anchoring pattern is consistent across all four benchmarks and both task domains. When the SLM majority is correct, context improves Oracle accuracy by +0.5 to +19.2pp. When the SLM majority is wrong, context degrades Oracle accuracy by −14.9 to −18.0pp across three of four benchmarks. The exception is HumanEval, where context has no effect on the majority-wrong subset—possibly because code generation tasks provide more objective verification signals (test execution) that resist anchoring. GSM8K shows a ceiling effect on the positive side (+0.5pp) because the Oracle already achieves 99.0% on the majority-correct subset, leaving little room for improvement. The negative side remains substantial (−14.9pp), confirming that even on easy benchmarks, incorrect SLM reasoning reliably degrades Oracle performance.

**Architectural tension on the escalated subset.** We examine this tension in detail on MATH 500,

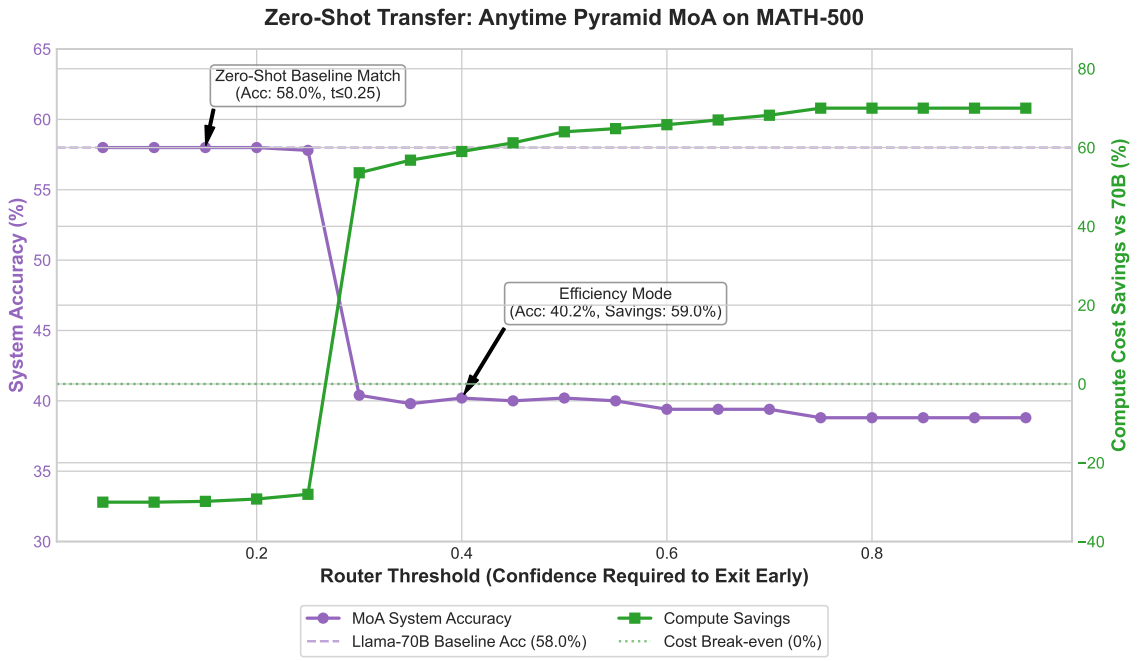



Figure 6: **Zero-Shot Transfer to MATH 500:** The GSM8K/MMLU-trained router transfers to out-of-distribution problems, preserving the Oracle ceiling (58.0%) and enabling efficiency gains at higher thresholds.

where the effect is most pronounced. On the 55 queries escalated by the router at $t = 0.4$, 95% have incorrect SLM majorities—precisely the subset where context is most harmful. On this escalated subset, passing SLM context degrades Oracle accuracy from 58.2% to 36.4% (−21.8pp). The router escalates records *because* it believes the SLMs are unreliable, yet the context-aware MoA design passes those unreliable outputs as context to the Oracle. This is the subset where context hurts the most.

**Implication.** The negative result validates the routing-based design: the router is doing real work by identifying genuinely hard problems (Layer 1 accuracy of 14.5% on the escalated subset), and blindly passing SLM context is actively harmful on exactly those problems. This motivates a natural extension—a *ternary* routing decision (escalate with context, escalate without context, or retain at Layer 1)—discussed in Section 5.

# 4 Related Work

**Anytime Algorithms & the Monitoring Problem:** The concept of trading computation time for solution quality originates with Dean and Boddy [1], who coined the term "anytime algorithm," and Horvitz [12], who independently developed the equivalent notion of "flexible computation." Zilberstein [2] extended this line to the composition of complex systems from anytime components, and Hansen and Zilberstein [3] framed the *monitoring problem* as a sequential decision process solvable by dynamic programming. Our work extends this line to generative AI, casting the multi-model routing decision as a single-step monitoring problem in the current two-layer architecture. The "early exit" strategy in deep learning, exemplified by BranchyNet [9], represents a related but architecturally distinct approach: early exit modifies the model's internal structure, whereas our framework operates on black-box model outputs.

**Mixture-of-Agents (MoA):** Collaborative layers of LLMs can outperform individual state-of-the-art models [4]. However, standard MoA implementations execute a pre-determined number of layers for all inputs, ignoring the heterogeneous difficulty of user queries. **Self-MoA** [5] demonstrates that iteratively querying a single strong model can sometimes outperform diverse ensembles. **Sparse MoA (SMoA)** [6] and **Residual MoA (RMoA)** [7] introduce early stopping and residual connections to reduce compute. Unlike RMoA, which relies on architectural residuals, our **Pyramid MoA** employs a distinct *decision-*

Table 4: Context-aware escalation across four benchmarks. "Baseline" denotes the Oracle solving independently; "With Context" denotes the Oracle receiving Layer 1 outputs. Results are segmented by whether the SLM majority answer was correct. The anchoring effect—context helps when SLMs are correct, hurts when SLMs are wrong—is consistent across both mathematical reasoning and code generation tasks.

| **Benchmark** | **SLM Majority** | $N$ | **Baseline** | **With Context** | $\Delta$ |
|---|---|---|---|---|---|
| MATH 500 | Correct | 239 | 70.3% | 89.5% | +19.2 pp |
| | Wrong | 261 | 59.0% | 41.0% | −18.0 pp |
| MMLU | Correct | 301 | 89.4% | 96.0% | +6.6 pp |
| | Wrong | 251 | 59.8% | 43.8% | −16.1 pp |
| GSM8K | Correct | 1079 | 99.0% | 99.4% | +0.5 pp |
| | Wrong | 121 | 88.4% | 73.6% | −14.9 pp |
| HumanEval | Correct | 112 | 89.3% | 97.3% | +8.0 pp |
| | Wrong | 52 | 63.5% | 63.5% | +0.0 pp |

*theoretic* approach grounded in anytime computation theory, providing formal monotonicity guarantees (Theorem 1) and a generalized escalation rule (Equation 5) that accounts for Oracle imperfection.

**LLM Cascading & Routing:** FrugalGPT [8] introduced cascading as a cost-reduction strategy. RouteLLM [11] trains preference-based routers using chatbot arena data to select between strong and weak models. Our framework subsumes simple cascading as a special case and provides the theoretical apparatus—performance profiles, the monotonicity condition, the generalized decision rule—to analyze when and why cascading works. The key distinction is that our router leverages ensemble agreement across multiple models rather than single-model confidence or pairwise preference data.

**Connection to Retrieval-Augmented Generation.** This finding parallels well-known challenges in retrieval-augmented generation (RAG), where passing incorrect retrieved documents to an LLM can cause anchoring—the model commits to plausible-but-wrong reasoning derived from the context rather than solving independently. In our setting, the Layer 1 ensemble outputs play the role of "retrieved candidates." On MATH 500, when those candidates are correct, passing them to the Oracle improves its accuracy by +19.2pp; when they are incorrect, context degrades Oracle performance by −18.0pp (59.0% → 41.0%). This pattern holds across all four benchmarks (Table 4), with the negative effect ranging from −14.9pp to −18.0pp. This suggests that effective context-aware Pyramid MoA requires not just passing Layer 1 outputs as context, but *filtering or weighting* those outputs based on confidence. A natural extension is to escalate with context only when the router's confidence in Layer 1 correctness exceeds a secondary threshold—effectively solving a two-threshold routing problem where the system decides not just *whether* to escalate, but *how* (with or without context). We defer this extension to future work but note that it directly motivates the ternary routing strategy outlined in Section 5.

# 5 Discussion & Conclusion

We presented Pyramid MoA, a framework that bridges classical anytime computation theory and modern multi-model LLM inference. By formalizing the routing decision as an instance of the anytime monitoring problem, we established a Probabilistic Anytime Property (Theorem 1) with provable monotonicity guarantees and derived a generalized escalation rule (Equation 5) that accounts for Oracle imperfection. Empirically, the framework demonstrates dynamic range across task difficulty: the Consensus Router achieves 81.6% bug recall on MBPP and transfers zero-shot to HumanEval with 62.7% cost savings; the Anytime Router nearly matches the 68.1% Oracle baseline on GSM8K/MMLU with up to 42.9% compute savings in economy mode, and transfers to MATH 500, preserving the 58.0% accuracy ceiling. Direct verification of the monotonicity condition (Theorem 1, Table 3) confirms that the probabilistic anytime property is strictly satisfied across all four benchmarks. Furthermore, context-aware escalation experiments reveal a consistent anchoring effect across all four benchmarks (−14.9 to −18.0pp when SLM majority is wrong), demonstrating that the routing-based design is not merely a cost optimization but a

necessary safeguard against context poisoning on hard queries.

**Limitations & Future Work.** The context-aware escalation analysis (Section 3.4) reveals that the Oracle cannot reliably discount incorrect SLM reasoning, even when the capability gap is substantial. A natural mitigation is a *ternary* routing decision: the router assigns queries to one of three actions—retain at Layer 1 (high SLM confidence), escalate *with* context (medium confidence, where SLMs are likely correct and context is beneficial), or escalate *without* context (low confidence, where SLM outputs are unreliable). The context-aware anchoring data suggests that such a ternary system could achieve accuracy *exceeding* the Oracle baseline: on medium-confidence queries where SLM context is correct, the Oracle with context outperforms the Oracle alone by up to +19.2pp (Table 4), a boost unavailable to the standalone Oracle. This extends the single-threshold decision rule to two thresholds and requires principled threshold transfer for zero-shot evaluation, which we leave to future work. Additionally, extending the architecture to deeper pyramids (e.g., 8B $\rightarrow$ 70B $\rightarrow$ 405B) would require the full sequential monitoring formulation from Hansen and Zilberstein [3]. A self-evaluating feedback loop—where Oracle corrections are used to LoRA fine-tune the base SLM—could reduce escalation rates over time. Finally, future work will incorporate comprehensive statistical significance testing across all benchmarks.

# References


[1] Thomas L Dean and Mark S Boddy. An Analysis of Time-Dependent Planning. *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, pages 49–54, 1988.

[2] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996.

[3] Eric A Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1-2):139–157, 2001.

[4] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents Enhances Large Language Model Capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

[5] Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking Mixture-of-Agents: Is Mixing Different Large Language Models Beneficial? *arXiv preprint arXiv:2502.00674*, 2025.

[6] Giang Do, Hung Le, and Truyen Tran. Sparse MoA. *arXiv preprint*, 2024.

[7] Zhentao Xie, Chengcheng Han, Jinxin Shi, Wenjun Cui, Xin Zhao, Xingjiao Wu, and Jiabao Zhao. Residual Mixture of Agents. *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. `https://aclanthology.org/2025.findings-acl.342/`

[8] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv preprint arXiv:2305.05176*, 2023.

[9] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. *23rd International Conference on Pattern Recognition (ICPR)*, 2016.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. *International Conference on Machine Learning (ICML)*, 2017.

[11] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data. *arXiv preprint arXiv:2406.18665*, 2024.

[12] Eric J Horvitz. Reasoning about Beliefs and Actions under Computational Resource Constraints. *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, 1987.