ColomboAI-MC-1: A Mixture-of-Models Intelligence System for Adaptive, Cost-Efficient, and Sovereign AI Inference

Wilfried Kouadio[1] Andrew Li[1]

[1]ColomboAI, Cairo Lab

Wilfried Kouadio – CEO of ColomboAI, Machine Learning Engineer and Researcher

Andrew Li – AI Lead at ColomboAI, Machine Learning Engineer

Preprint – August 14, 2026

Abstract

The rapid proliferation of open and proprietary language models has created a deployment paradox: model diversity is increasing faster than the ability of developers and enterprises to select, evaluate, govern, and economically operate those models. A single fixed model necessarily inherits one capability profile, one cost structure, one latency regime, one context window, and one deployment topology. We introduce *ColomboAI-MC-1* (MC-1), a Mixture-of-Models (MoM) intelligence system that moves conditional computation from the *intra-model* level of Mixture-of-Experts (MoE) to the *inter-model* level. Rather than routing tokens among subnetworks inside a single trained model, MC-1 routes requests—and, when advantageous, reasoning stages—across independently trained models according to estimated capability, task requirements, cost, latency, privacy, policy, availability, and confidence. The system combines a Query Intelligence layer, a continuously updated Model Intelligence Layer, a constrained utility router, confidence-aware escalation, selective multi-model deliberation, enterprise policy enforcement, and closed-loop self-evaluation. We formalize the routing problem as constrained expected-utility maximization, distinguish the best-fixed-model baseline from oracle and realizable routers, and show why heterogeneous model complementarity creates a principled opportunity to exceed any fixed member of a model pool. We further propose an evaluation protocol centered on quality, cost per correct answer, latency, routing regret, oracle recovery, calibration, and policy compliance rather than accuracy alone. MC-1 has been developed at ColomboAI since March 2025 as a unified inference abstraction intended to make model choice an infrastructure concern rather than an application-level burden. This paper presents the architecture, mathematical formulation, systems design, evaluation methodology, and research agenda for treating open-model ecosystems as a programmable collective intelligence substrate.

1 Introduction

Language-model deployment is increasingly a model-selection problem. Modern applications can choose among small local models, large open-weight models, reasoning-specialized models, coding models, multimodal models, long-context models, and frontier hosted models. These alternatives differ not only in raw capability but also in task specialization, inference price, response latency, memory footprint, context length, privacy properties, licensing constraints, hardware compatibility, and availability. The conventional deployment pattern—select one model identifier and send every request to it—compresses this heterogeneous landscape into a static decision made by the application developer.

This static abstraction is increasingly inefficient. A model sufficiently capable for the hardest 1% of requests is frequently over-provisioned for the remaining 99%. Conversely, a small economical model selected for average traffic may fail disproportionately on difficult reasoning, coding, long-context, multilingual, or tool-use tasks. The central hypothesis of MC-1 is therefore simple:

> **Model selection should be a learned, policy-constrained runtime decision, not a permanent application configuration.**

This principle is analogous to the motivation behind sparse Mixture-of-Experts (MoE). Sparse MoE architectures increase effective capacity by conditionally activating a small subset of internal experts rather than using all parameters for every token [1, 2, 3]. MC-1 elevates conditional computation by one level of abstraction. In a Mixture-of-Models (MoM) system, the experts are independently trained models, potentially hosted on different hardware, in different clouds, under different governance regimes, or locally on user-controlled infrastructure.

The idea is increasingly supported by a growing routing literature. FrugalGPT showed that learned cascades can preserve or improve quality while substantially reducing cost [4]; RouteLLM formalized preference-based routing between models [5]; Zooter demonstrated reward-guided expert routing across heterogeneous LLMs [6]; MixLLM studied dynamic routing in mixed model pools [7]; and LLMRouterBench evaluated more than 400,000 instances across 21 datasets and 33 models, confirming strong model complementarity while also identifying a substantial gap between practical routers and oracle selection [8]. In 2026, work explicitly using the term Mixture-of-Models reported that capability-aware routing can surpass the strongest fixed model in its tested pool while reducing inference cost [9].

This paper describes MC-1, developed at ColomboAI beginning in March 2025, as a general architecture for operationalizing this direction. Our goal is broader than prompt routing. We treat model selection as a continuously learned systems problem involving task understanding, capability estimation, economic optimization, confidence calibration, policy enforcement, fallback

behavior, and feedback-driven adaptation.

Contributions. We make six principal contributions:

1. We formalize *Mixture-of-Models Intelligence* as conditional computation across independently trained models, clearly distinguishing it from MoE and output-level ensembling.
2. We introduce the MC-1 architecture: Query Intelligence, a Model Intelligence Layer (MIL), constrained utility routing, cascading, selective deliberation, enterprise policy enforcement, and self-evaluation.
3. We formulate routing as constrained expected-utility maximization over quality, cost, latency, privacy, availability, and operator policy.
4. We state the complementarity condition under which oracle per-query selection is strictly superior to any fixed model and define *oracle recovery* as a key router metric.
5. We propose a reproducible evaluation protocol that emphasizes intelligence-per-dollar, cost-per-correct-answer, latency, routing regret, calibration, and policy compliance.
6. We articulate MC-1 as an inference-control abstraction for open models: applications invoke one stable endpoint while the model pool can evolve independently.

2 Background and Related Work

2.1 From Mixture-of-Experts to Mixture-of-Models

Sparse MoE replaces dense activation with conditional expert selection. Shazeer et al. introduced a sparsely gated expert layer with a trainable routing network [1]. GShard scaled conditional computation to hundreds of billions of parameters [2], and Switch Transformer simplified routing to top-1 expert activation while demonstrating strong scaling behavior [3]. In these systems, routing generally occurs *within* a single parameterized model and is trained jointly or semi-jointly with the model.

A MoM system differs in four important ways. First, experts are complete independently trained models. Second, experts may have incompatible tokenizers, architectures, modalities, providers, licenses, or serving stacks. Third, model membership can change after deployment without retraining the other experts. Fourth, routing can incorporate operational variables—price, region, accelerator availability, data residency, contractual policy, or provider health—that are external to model weights.

2.2 Model Routing and Cascading

FrugalGPT formalized economical cascades among LLM APIs and reported settings in which cost could be reduced dramatically while preserving or improving quality [4]. AutoMix used self-verification and partially observable decision processes to decide when to escalate to a stronger model [10]. RouteLLM learned routing functions from preference data and demonstrated that useful routers can transfer when model pairs change [5]. Zooter used reward signals to discover latent expertise and route queries to models without invoking all candidates [6]. MixLLM optimized quality, cost, and latency for streams of heterogeneous requests [7].

A recent survey organizes model-routing approaches by when decisions are made, what information is used, and how routing decisions are computed, emphasizing that practical systems are compositional rather than reducible to a single router algorithm [11]. The vLLM Semantic Router likewise demonstrates composable signal-driven decisions combining heuristics, classifiers, policy checks, and multiple selection algorithms in production-oriented serving [12].

2.3 Ensembling and Deliberation

Routing selects one model before or during execution; ensembling runs multiple models and combines outputs. LLM-Blender uses ranking and fusion to exploit complementary outputs [13]. Mixture-of-Agents (MoA) uses layered collaboration among multiple LLM agents and has demonstrated that cross-model collaboration can improve generation quality [14]. These methods can be powerful but typically spend more inference compute than single-path routing. MC-1 therefore treats deliberation as a selective escalation mode rather than the default execution strategy.

2.4 Benchmarking Routers

LLMRouterBench is particularly relevant because it evaluates routing as a field rather than a single method. Its 2026 benchmark includes more than 400,000 examples, 21 datasets, 33 models, and 10 representative routing baselines. The study confirms model complementarity but also finds that several sophisticated routers fail to reliably beat simple baselines under unified evaluation [8]. More recent selection-valid diagnostics further caution that an outcome oracle reveals opportunity but is not itself a deployable policy; large oracle gaps can coexist with limited realizable recovery by prompt-only routers [15]. This distinction motivates MC-1's use of richer signals, continuous telemetry, calibrated escalation, and closed-loop learning.

3 Problem Formulation

Let x denote an incoming request, including prompt, conversation state, modality, tool context, and application metadata. Let $\mathcal{M} = \{m_1, \ldots, m_K\}$ denote the available model pool. A model m_i has measurable or estimable properties including quality $Q_i(x)$, monetary cost $C_i(x)$, latency $L_i(x)$, risk $R_i(x)$, availability A_i, and a capability vector $\mathbf{c}_i \in \mathbb{R}^d$.

The request is represented by a requirement vector $\mathbf{r}(x) \in \mathbb{R}^d$ containing estimated needs such as reasoning depth, coding complexity, mathematical complexity, long-context sensitivity, modality, tool use, language, factuality, structured-output reliability, and safety sensitivity. A deployment policy p specifies hard constraints and soft preferences.

A fixed-model system selects a single m_j for all x. A routing policy π selects a model conditional on the request and policy:

$$\pi(x, p) \to m_i \in \mathcal{M}. \tag{1}$$

Table 1: Representative related work and the design dimension most relevant to MC-1. Reported numerical results are those claimed by the cited works in their own evaluation settings and are not directly comparable across studies.

Work	Paradigm	Core idea	Relevance to MC-1
FrugalGPT [4]	Cascade	Learn economical combinations of LLM calls	Demonstrates cost–quality gains from conditional model use
AutoMix [10]	Cascade	Self-verification with confidence-aware escalation	Motivates calibrated fallback and escalation
Zooter [6]	Router	Reward-guided discovery of latent model expertise	Supports capability-specialized routing
RouteLLM [5]	Router	Preference-trained strong/weak model selection	Formalizes learned model choice under cost constraints
LLM-Blender [13]	Ensemble	Rank and fuse outputs from multiple models	Supports selective multi-model synthesis
Mixture-of-Agents [14]	Ensemble	Layered collaboration across multiple LLM agents	Motivates deliberation for high-value hard tasks
MixLLM [7]	Router	Dynamic routing under quality/cost/latency objectives	Reinforces online, multi-objective routing
LLMRouterBench [8]	Benchmark	400K+ examples, 21 datasets, 33 models	Establishes complementarity and oracle-gap metrics
Brick [9]	Mixture-of-Models	Capability-vector and difficulty-aware geometric routing	Directly supports the emerging MoM abstraction

The simplest MC-1 objective is a constrained expected utility:

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}_{x\sim\mathcal{D}}[U(\pi(x,p),x,p)] \quad \text{s.t.} \quad g_j(\pi,x,p) \le 0, \tag{2}$$

where g_j encode hard requirements such as data residency, maximum cost, provider allowlists, minimum context window, or local-only inference.

A useful utility decomposition is

$$U_i(x,p) = w_q\widehat{Q}_i(x) - w_c\widehat{C}_i(x) - w_l\widehat{L}_i(x) \tag{3}$$

$$- w_r\widehat{R}_i(x) + w_a\widehat{A}_i + w_s S_i(x,p), \tag{4}$$

where normalized terms are estimated before generation and S_i captures policy-specific strategic preferences (e.g., local inference, preferred hardware, contractual commitments, or energy targets).

3.1 Best Fixed Model, Oracle, and Realizable Router

For binary correctness $Y_i(x) \in \{0,1\}$, define the best fixed model performance as

$$P_{\text{fixed}} = \max_i \mathbb{E}[Y_i(x)]. \tag{5}$$

The outcome oracle is

$$P_{\text{oracle}} = \mathbb{E}\left[\max_i Y_i(x)\right]. \tag{6}$$

Because expectation of a maximum is at least the maximum of expectations,

$$P_{\text{oracle}} \ge P_{\text{fixed}}. \tag{7}$$

The inequality is strict whenever models exhibit complementary correctness on a non-zero-measure set of requests: i.e., there exist x_a, x_b and models $i \ne j$ such that $Y_i(x_a) > Y_j(x_a)$ while $Y_j(x_b) > Y_i(x_b)$, and no single model dominates almost surely.

The deployable challenge is to recover this opportunity from *pre-answer signals*. We define oracle recovery:

$$\rho = \frac{P_{\text{router}} - P_{\text{fixed}}}{P_{\text{oracle}} - P_{\text{fixed}}}, \tag{8}$$

for $P_{\text{oracle}} > P_{\text{fixed}}$. This metric separates two questions: (1) whether the model pool contains complementary capability, and (2) whether the router can identify it before seeing outcomes. This distinction is central to fair evaluation [15].

4 The MC-1 Architecture

Figure 1 summarizes MC-1. The architecture is designed around a stable application-facing endpoint and a dynamic model-facing control plane.

4.1 Query Intelligence

The Query Intelligence layer converts raw requests into structured pre-answer signals. It should be intentionally lightweight because router overhead reduces the economic benefit of routing. Signals can include:

- semantic domain and subdomain;
- estimated difficulty and reasoning depth;
- code, mathematics, data-analysis, or retrieval requirements;
- context length and expected output length;
- modality (text, image, audio, structured data);
- language and locale;
- tool/function-calling requirements;
- required determinism or structured-output reliability;
- privacy, safety, and compliance sensitivity;
- estimated value or criticality of the request.

Signals may be extracted through heuristics, embeddings, compact classifiers, historical similarity, or a small routing model. Production systems can compose these signals rather than rely-

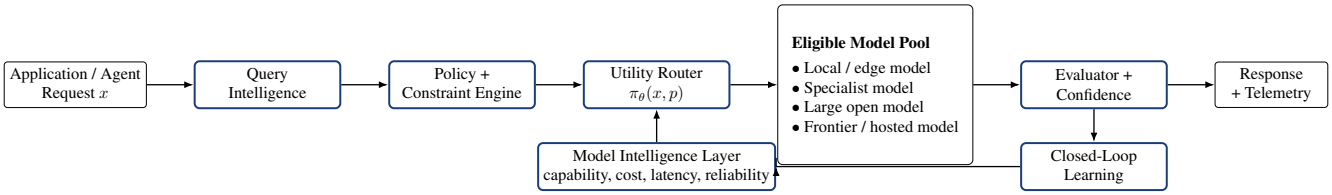

Figure 1: High-level MC-1 architecture. Query requirements and enterprise policy are matched against continuously measured model capabilities. Evaluation telemetry updates the Model Intelligence Layer and may trigger escalation or selective multi-model deliberation.

ing on a single semantic classifier, consistent with recent signal-driven routing work [12].

4.2 Model Intelligence Layer

The MIL maintains a dynamic capability profile for each candidate model. Rather than storing a scalar leaderboard score, it records a vector:

$$\mathbf{c}_i = [c_i^{\text{code}}, c_i^{\text{math}}, c_i^{\text{reason}}, c_i^{\text{vision}}, c_i^{\text{tool}}, \ldots]. \tag{9}$$

Each capability estimate should be accompanied by uncertainty, recency, sample count, and operating conditions. A model that is excellent at short coding tasks may be weak at repository-scale software engineering; a model strong in benchmark mathematics may be slow, expensive, or difficult to host locally. MIL therefore also stores operational features: throughput, time-to-first-token, decode rate, context capacity, price, accelerator type, quantization, region, provider health, and policy labels.

The model profile is updated from four sources: controlled benchmark runs, production outcomes, evaluator judgments, and operator feedback. We advocate exponentially weighted updates or Bayesian estimates rather than permanent benchmark constants because model endpoints, quantizations, serving kernels, and provider behavior change over time.

4.3 Utility Router

The router filters infeasible candidates using hard constraints, then ranks feasible candidates by predicted utility. A practical implementation can be decomposed into:

1. **Eligibility**: remove models violating policy or capability minima.
2. **Prediction**: estimate quality, cost, latency, and failure risk for each eligible model.
3. **Selection**: maximize policy-weighted expected utility.
4. **Confidence**: estimate uncertainty in the routing decision.

This decomposition makes routing explainable: the system can report *why* a model was selected and which alternatives were rejected.

4.4 Confidence-Aware Cascading

Some requests should not be committed irrevocably to the first model. MC-1 supports a cascade in which a low-cost model attempts the task, a verifier estimates answer adequacy, and the system escalates only when confidence is insufficient. Given ordered models $m_1, \ldots, m_n$ with increasing expected cost, the cascade stops at the first m_j satisfying

$$\Pr(\text{acceptable} \mid x, y_j, m_j) \geq \tau_p, \tag{10}$$

where τ_p is policy-dependent. This is especially useful for high-volume workloads containing many easy requests and a long tail of difficult ones.

4.5 Selective Multi-Model Deliberation

For high-value tasks, the optimal action may be to invoke more than one model. MC-1 supports a deliberate mode in which multiple candidates produce independent analyses or artifacts that are then ranked, cross-checked, or synthesized. Because deliberation multiplies inference cost, the router should treat it as a distinct action a_{delib} with its own expected utility rather than a universal default.

4.6 Enterprise Policy and Sovereign Inference

Many deployments are not free to choose any model. Policies may require local-only execution, data residency, specific model licenses, provider allowlists, maximum price per request, minimum reliability, or use of approved accelerators. MC-1 models these restrictions as first-class constraints rather than post-hoc filters.

This enables a sovereign deployment pattern in which the same logical endpoint can route among organization-controlled open models inside a private cloud, on-premises cluster, workstation, or edge device, while optionally permitting escalation to external providers for approved workloads. Importantly, sovereignty is not a property of a model alone; it is a property of the full routing and serving path.

4.7 Closed-Loop Self-Evaluation

Every routed request generates telemetry. Let z_t contain model selection, predicted utility, realized latency/cost, evaluator score, user feedback, tool outcome, and any escalation event. The router updates its estimates from z_t to reduce future regret. The objective is not uncontrolled self-modification, but measured adaptation under evaluation gates and rollbackable policies.

Algorithm 1: MC-1 constrained routing

Input: request x, policy p, model pool $\mathcal{M}$
1. Extract requirement vector $\mathbf{r}(x)$ and routing signals.
2. Fetch current model profiles $\{\mathbf{c}_i, o_i\}$ from MIL.
3. Filter $\mathcal{M}$ by hard policy/capability constraints.
4. Predict $\widehat{Q}_i, \widehat{C}_i, \widehat{L}_i, \widehat{R}_i$ for feasible m_i.
5. Compute utility $U_i(x, p)$ and router confidence.
6. If confidence is high, invoke $\arg\max_i U_i$.
7. Else choose policy-allowed cascade or deliberation path.
8. Evaluate outcome; record telemetry and update MIL/router.

Output: response, route metadata, evaluation telemetry.

Figure 2: Reference routing loop. Implementations can replace individual estimators without changing the overall control abstraction.

5 Routing Algorithms

5.1 Capability Matching

One interpretable selection strategy scores geometric alignment between request requirements and model capability:

$$S_i(x) = \mathrm{sim}(\mathbf{r}(x), \mathbf{c}_i) - \lambda_c \widehat{C}_i(x) - \lambda_l \widehat{L}_i(x) - \lambda_r \widehat{R}_i(x). \quad (11)$$

This family is attractive because it exposes capability dimensions explicitly. Brick provides recent evidence that capability-vector routing can be competitive in a MoM setting [9].

5.2 Learned Preference Routing

A learned router can estimate pairwise preference

$$P(m_i \succ m_j \mid x, p) \quad (12)$$

and select the candidate with maximum aggregate utility. Preference training can use human labels, evaluator models, application outcomes, or synthetic comparisons. RouteLLM demonstrates the practicality of preference-based routing [5].

5.3 Contextual Bandit Adaptation

Production traffic is non-stationary. New models arrive, providers change pricing, user populations drift, and task distributions evolve. We therefore view routing as a contextual bandit with context $s_t = (x_t, p_t, \mathrm{MIL}_t)$, action $a_t \in \mathcal{M} \cup \{a_{\mathrm{cascade}}, a_{\mathrm{delib}}\}$, and delayed reward

$$r_t = w_q q_t - w_c c_t - w_l l_t - w_r r_t^{\mathrm{risk}}. \quad (13)$$

Exploration must be policy-constrained and safety-aware; high-risk production traffic should not be used for unconstrained experimentation. Offline replay, shadow routing, and canary allocations are preferable for early learning.

5.4 Pseudocode

6 Why a Mixture of Models Can Beat a Single Model

The potential advantage of MoM is not that routing magically improves a model. It is that different models make different errors. If error sets are not identical, per-query selection can exploit complementary competence.

Suppose two models have average accuracy 0.80 and 0.78. If their failures are perfectly correlated, routing cannot improve quality; the stronger model dominates. If, however, each solves a meaningful subset of cases the other misses, then an oracle can exceed 0.80. The value of routing therefore depends more on *complementarity* than on raw model count. LLMRouterBench explicitly confirms strong complementarity while finding diminishing returns from indiscriminately enlarging the pool [8]. This suggests that model curation is itself an optimization problem.

A second advantage is economic specialization. Let m_s be a small inexpensive model and m_l a strong expensive model. If a fraction α of traffic can be handled by m_s without quality loss, expected cost under perfect simple/complex routing is

$$\mathbb{E}[C] = \alpha C_s + (1 - \alpha) C_l, \quad (14)$$

which is strictly below C_l whenever $C_s < C_l$ and $\alpha > 0$. The practical problem becomes estimating the set of requests for which m_s is sufficient.

Third, the model pool can change faster than a monolithic model can be retrained. A newly released specialist can enter the pool, receive shadow traffic, be benchmarked, and gradually acquire routes. This makes MoM a systems-level mechanism for continuous capability acquisition.

7 The MC-1 Evaluation Protocol

A routing system should not be evaluated solely by average benchmark accuracy. A router that gains 0.5 points at 5× cost may be inferior to one that loses 0.2 points while cutting cost 80%. Conversely, a cost-saving router that systematically misroutes difficult safety-critical tasks is unacceptable. We therefore propose a multidimensional protocol.

7.1 Research Questions

A complete evaluation should answer:
1. **RQ1 – Quality:** Does MC-1 outperform the best fixed model under equal or lower budget?
2. **RQ2 – Efficiency:** What quality is achieved per dollar, per joule, and per second?
3. **RQ3 – Routing:** How much of the oracle opportunity does the router recover?
4. **RQ4 – Robustness:** Does performance persist under distribution shift and model-pool changes?
5. **RQ5 – Calibration:** Are escalation and confidence thresholds reliable?
6. **RQ6 – Governance:** Does the system obey hard policy constraints with zero violations?

7.2 Task Families

The benchmark suite should cover heterogeneous domains because routing is only meaningful when models have different strengths. Recommended families include:
- general knowledge and reasoning (e.g., MMLU-Pro [16]);
- mathematical reasoning and competition-style problems;

- code generation and repository-scale software engineering (e.g., SWE-bench [17]);
- long-context retrieval and synthesis;
- structured extraction and constrained generation;
- multilingual understanding and generation;
- tool/function calling and agentic execution;
- multimodal understanding;
- open-ended preference evaluation.

7.3 Candidate Model Pool

For reproducibility, every evaluation should disclose exact model checkpoints or API versions, quantization, inference provider, decoding parameters, context configuration, hardware class, and timestamp. The pool should intentionally include heterogeneous candidates rather than multiple near-duplicates. We recommend at least four strata: small/local, medium generalist, specialist, and large/high-capability.

7.4 Baselines

At minimum, compare against:
1. best fixed model by held-out validation;
2. cheapest fixed model;
3. random routing with matched model frequencies;
4. simple semantic/domain routing;
5. cost-only routing;
6. a learned preference router;
7. cascade baseline;
8. outcome oracle (upper-bound diagnostic only).

7.5 Metrics

We recommend reporting:

Task quality: accuracy, pass@1, success rate, preference win rate;

Economics: mean cost/query, cost/correct, cost/success;

Latency: TTFT, p50/p95 end-to-end latency, throughput;

Routing: regret, oracle recovery ρ, route entropy;

Calibration: ECE/Brier score for confidence and escalation;

Reliability: timeout rate, failover success, availability;

Governance: hard-constraint violation rate.

For cost-sensitive comparisons, plot a Pareto frontier rather than a single scalar score. A strong router should shift the frontier outward: higher quality for a fixed budget or lower cost for a fixed quality level.

7.6 Ablations

To identify where gains originate, disable one component at a time:
- no Model Intelligence Layer (uniform model priors);
- no difficulty estimation;
- no cost or latency term;
- no escalation;
- no online adaptation;
- no policy features;

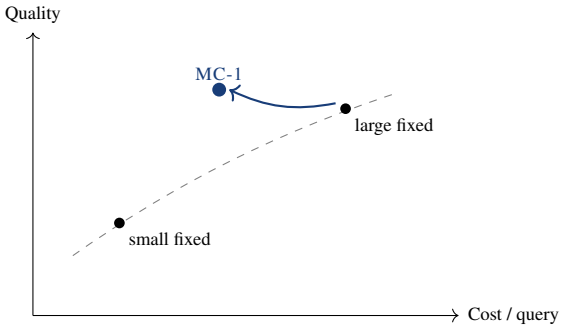

Figure 3: Conceptual evaluation objective. A useful router should improve the quality–cost Pareto frontier rather than optimize either axis in isolation. This figure is illustrative, not an empirical MC-1 result.

- no historical task-level performance statistics;
- reduced model pool.

The reduced-pool ablation is important because recent work suggests careful curation can matter more than simply adding models [8].

7.7 Statistical Discipline

Router evaluation risks post-selection bias. Model pools, thresholds, and router families should be tuned on validation data and evaluated once on held-out test data. When multiple router policies are tested, confidence intervals should account for selection. Oracle results should be labeled as an opportunity diagnostic rather than a realizable deployment result, consistent with recent critiques [15].

8 Systems Design for Open-Model Deployment

8.1 One Endpoint, Mutable Intelligence

The application-facing contract should remain stable:

```
model = "colomboai/mc-1"
```

while the underlying model pool changes over time. This inversion is operationally significant. Instead of requiring every application to track model releases, benchmark results, provider pricing, and deprecations, those concerns move into a centralized inference-control layer.

8.2 Provider Abstraction

Candidate models may be local processes, Kubernetes services, GPU clusters, serverless endpoints, or external APIs. MC-1 normalizes these behind a capability-and-execution interface containing at least: model identity, version, modality, context window, tool support, cost function, latency telemetry, health, region, and policy labels.

8.3 Fallback and Availability

Routing must account for failure. The highest-utility model is not useful if its provider is unhealthy or rate-limited. Availability is therefore a dynamic input, and each route should include a

fallback chain. Fallbacks should preserve hard constraints; a local-only policy must never silently fail over to an external API.

8.4 Caching and Reuse

Semantic and exact-match caches can reduce both cost and latency. However, caching interacts with routing: cached answers should be treated as another candidate action with estimated freshness and risk. When a cache hit is valid, the cheapest model call is no model call.

8.5 Observability

Every request should produce a trace containing query classification, eligible models, predicted utilities, chosen route, cost, latency, evaluator outcome, fallback events, and policy decisions. This supports debugging, auditability, benchmark reconstruction, and router learning.

9 Security, Privacy, and Governance

A cross-model router expands the attack and governance surface because a request may cross model, provider, or regional boundaries. MC-1 therefore requires policy evaluation before model invocation. Relevant controls include:

- data-classification-aware provider eligibility;
- signed model/provider identities and version pinning;
- request redaction or transformation before external routing;
- tool and agent permission propagation;
- audit logging of routing decisions;
- fail-closed behavior for policy uncertainty;
- explicit separation of quality optimization from safety constraints.

Safety should not be represented only as a soft negative term in utility. Certain constraints must remain hard and non-tradable: a cheaper or more capable model cannot compensate for violating a legal or organizational prohibition.

10 Limitations

Router error. The central limitation is selection uncertainty. Complementarity does not guarantee realizability: a router can only exploit differences it can predict from pre-answer signals. Recent benchmarking shows large gaps between oracle opportunity and practical router performance [8, 15].

Evaluation quality. Closed-loop learning is only as reliable as its evaluators. Automated judges may inherit biases, reward verbosity, or fail on specialized domains. Ground-truth outcomes should be preferred when available.

Non-stationarity. Model providers can alter checkpoints, system prompts, pricing, rate limits, or serving infrastructure. Capability profiles therefore expire and require continuous validation.

Overhead. Query classification, routing, verification, and telemetry consume compute. The router must remain much cheaper and faster than the inference it seeks to optimize.

Pool complexity. Larger pools increase operational burden and can create diminishing returns. Pool construction should optimize complementarity, not cardinality alone.

Benchmark contamination. Public model benchmarks may be contaminated or saturated. Production-like held-out workloads and task-specific success criteria are essential.

11 Broader Implications

If the model ecosystem continues to diversify, the dominant application abstraction may shift from *model APIs* to *intelligence APIs*. Under this view, an application specifies the required result and policy constraints while an inference layer chooses the appropriate compute substrate.

This has several implications. First, open-source models become easier to adopt because users no longer need to choose a single winner. Second, small and specialized models gain economic value: they need not replace large models globally; they only need to dominate a useful region of the workload. Third, model innovation can be integrated continuously without forcing application migrations. Fourth, organizations can optimize sovereignty and privacy alongside quality and cost rather than treating them as separate infrastructures.

This architecture also changes benchmarking incentives. A model's value is no longer only its average score; it includes its *marginal complementary coverage* relative to an existing pool. A specialist that solves a narrow but economically important failure cluster may be more valuable to a MoM system than a generalist with a slightly higher global average.

12 Research Agenda

We identify eight open research directions for MC-1 and MoM systems:

1. **Selection-valid learning:** train routers to recover more oracle opportunity without leaking outcome information.
2. **Pool optimization:** choose compact complementary model sets under serving constraints.
3. **Step-level routing:** route reasoning stages or tool calls rather than whole prompts when beneficial.
4. **Cross-modal routing:** jointly route text, image, audio, video, and structured sensor workloads.
5. **Sovereign routing:** optimize across local, edge, private-cloud, and public endpoints while preserving hard governance boundaries.
6. **Continual capability estimation:** detect drift and update capability vectors online with calibrated uncertainty.
7. **Energy-aware inference:** add energy, carbon intensity, and accelerator utilization to the utility function.
8. **Agentic routing:** choose models dynamically across planning, coding, browsing, tool use, verification, and long-running autonomous tasks.

13 Conclusion

Mixture-of-Experts established a powerful principle: large capacity does not require activating every component for every input. MC-1 applies the same conditional-computation intuition at the model ecosystem level. Instead of forcing every request through one general-purpose model, MC-1 treats independently trained models as a dynamic set of experts and selects among them according to task requirements, measured capability, economics, latency, confidence, policy, and availability.

The value proposition is not merely lower inference cost, and it is not merely higher benchmark scores. It is a change in abstraction. Developers should not have to continuously decide which model the application *is*. They should specify what the application needs, and an intelligence layer should determine which model—or sequence of models—can satisfy that requirement most efficiently and safely.

The research literature now provides substantial evidence for model complementarity and for the feasibility of learned routing, while also warning that oracle opportunity is much larger than the capability of current routers. That gap defines the research problem. MC-1 is our attempt to treat it as a full-stack intelligence systems problem: measurable, adaptive, policy-aware, economical, and compatible with a continuously evolving open-model ecosystem.

Reproducibility and Claim Boundary

This manuscript is an architecture and methods paper. Numerical results attributed to prior work are reported only with citations and should be interpreted within the original authors' experimental settings. The conceptual Pareto diagram in Figure 3 is illustrative. We intentionally do not fabricate unpublished MC-1 benchmark values. A reproducible empirical release should publish exact model versions, prompts, inference settings, routing traces, cost tables, evaluation code, and held-out results using the protocol defined in this paper.

Acknowledgments

The authors thank the ColomboAI engineering and research teams for their work on model orchestration, inference infrastructure, evaluation, and the broader Cairo intelligence stack.

Author Contributions

Wilfried Kouadio conceived the Mixture-of-Models vision for ColomboAI-MC-1, productized the system direction, and led the architecture and research framing. Andrew Li contributed to machine-learning architecture, model intelligence, evaluation design, and technical development. Both authors contributed to the manuscript and research agenda.

Conflict of Interest

The authors are affiliated with ColomboAI, which develops ColomboAI-MC-1 and related AI infrastructure.

References

[1] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". In: *arXiv preprint arXiv:1701.06538* (2017).

[2] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding". In: *arXiv preprint arXiv:2006.16668* (2020).

[3] William Fedus, Barret Zoph, and Noam Shazeer. "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39.

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

[5] 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". In: *International Conference on Learning Representations* (2025).

[6] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. "Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models". In: *Proceedings of NAACL-HLT* (2024), pp. 1964–1974.

[7] Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. "MixLLM: Dynamic Routing in Mixed Large Language Models". In: *Proceedings of NAACL-HLT* (2025), pp. 10912–10922.

[8] Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. "LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing". In: *arXiv preprint arXiv:2601.07206* (2026).

[9] Francesco Massa and Marco Cristofanilli. "Brick: Spatial Capability Routing for the Mixture-of-Models (MoM) Paradigm". In: *arXiv preprint arXiv:2606.13241* (2026).

[10] Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. "AutoMix: Automatically Mixing Language Models". In: *arXiv preprint arXiv:2310.12963* (2023).

[11] Yasmin Moslem and John D. Kelleher. "Dynamic Model Routing and Cascading for Efficient LLM Inference: A Survey". In: *arXiv preprint arXiv:2603.04445* (2026).

[12] Xunzhuo Liu, Huamin Chen, Samzong Lu, Yossi Ovadia, Guohong Wen, Zhengda Tan, Jintao Zhang, Senan Zedan, Yehudit Kerido, Liav Weiss, et al. "vLLM Semantic Router: Signal Driven Decision Routing for Mixture-of-Modality Models". In: *arXiv preprint arXiv:2603.04444* (2026).

[13] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. "LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion". In: *arXiv preprint arXiv:2306.02561* (2023).

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

[15] Ibne Farabi Shihab, Abu Sa-Adat Mohamed Moon-Im Ahsan, and Md Najmus Swaqeeb. "Opportunity Is Not Realizability: Selection-Valid Diagnostics for Multi-LLM Routing". In: *arXiv preprint arXiv:2608.08265* (2026).

[16] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. "MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark". In: *arXiv preprint arXiv:2406.01574* (2024).

[17] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" In: *arXiv preprint arXiv:2310.06770* (2023).

A Reference Capability Schema

A production MIL entry can be represented by the following conceptual schema:

```
model_id
version / checkpoint
provider / region / endpoint
modality[]
context_window
capability_vector {
  reasoning, math, coding,
  long_context, vision,
  multilingual, tool_use,
  structured_output, factuality
}
uncertainty_vector
cost_model { input, output, fixed }
latency { ttft_p50, ttft_p95,
          decode_tps, e2e_p95 }
reliability { timeout, error,
              success_rate }
policy_labels[]
hardware / quantization
last_evaluated_at
benchmark_provenance[]
production_sample_count
```

B Example Enterprise Policies

B.1 Local-First Policy

```
require: data_residency = local
prefer:  local_open_weight = true
max_cost_per_request = 0.002
min_quality_confidence = 0.90
allow_external_escalation = false
```

B.2 Hybrid Sovereign Policy

```
require: pii_external = false
prefer:  local_model = true
allow:   approved_external_models
external_escalation_if:
  predicted_quality_gain >= 0.15
  and request_class != restricted
```

B.3 Latency-Critical Agent Policy

```
max_ttft_ms = 500
max_e2e_ms = 3000
require_tool_calling = true
prefer_small_model_if:
  predicted_success >= 0.95
fallback_on_timeout = true
```

C Recommended Benchmark Report Card

A public MC-1 evaluation should disclose a compact report card for each workload and aggregate:

Metric	Value
Task quality	–
Best fixed model quality	–
Outcome oracle quality	–
Oracle recovery ρ	–
Mean cost / request	–
Cost / correct answer	–
p50 / p95 latency	–
Escalation rate	–
Route entropy	–
Policy violations	0 required

Table 2: Template for an empirical MC-1 report. Dashes intentionally indicate measurements to be populated by reproducible experiments rather than invented values.

D Suggested Empirical Release Sequence

A rigorous companion evaluation can be released in three stages. First, publish a fixed static candidate pool and compare MC-1 to best-fixed, random, domain, cost-only, learned-router, and cascade baselines. Second, add a time-split experiment in which new models are introduced after router training to test adaptability and transfer. Third, run an online shadow deployment on real application traffic, recording realized latency, cost, human or task-level success, and calibration drift without exposing users to experimental routes until safety and policy gates are satisfied.