

Perspective Article

Open Access

When Does Mixing Models Pay? OpenFusion, a Bounded Mixture-of-Agents Runtime, and Registered Evidence from Local Coding Benchmarks

John Cheung

Hong Kong SAR

ABSTRACT

Mixture-of-Agents (MoA) methods combine the outputs of several large language models (LLMs) at inference time and have reported gains on conversational benchmarks. Whether those gains survive deployment constraints — fixed call budgets, local hardware, objective grading and fair same-model baselines — is less clear. I present OpenFusion, an open-source, OpenAI-compatible runtime that implements fourteen bounded multi-model strategies, from fallback and Self-MoA to semantic voting, rank-then-fuse, layered refinement and uncertainty cascades, under a hard per-request call ceiling. It ships with an evaluation laboratory that reports Wilson intervals, paired tests, order-balanced judging and panel-complementarity diagnostics.

Using the project's published artifacts, I report a series of local studies on two NVIDIA GB10 workstations with 4B-parameter open models. In a registered prospective study on 500 previously unused MBPP tasks, a fixed test-guided mixed-model workflow passed 78.1% of all-test evaluations versus 72.5% for repeated same-model repair (+5.6 points, 95% CI [3.6, 7.7]), with 17.2% lower mean latency and 29.2% fewer tokens. A confidence-gated escalation policy raised held-out accuracy on a 70-item programmatic benchmark from 75.7% to 91.4% while calling a stronger model on 40% of items.

Five later controls, however, did not show that model diversity itself causes the gains. Against fresh sampling from the same model, mixed panels tied or underperformed: -0.57 points on 1,500 tasks, and -5.0 points after a 32-policy development search. I conclude that, in this setting, the measurable benefit of MoA-style systems comes from verification, bounded retries and selective escalation rather than from mixing similarly sized models. I close with a five-step selection rule and a reporting checklist for practitioners.

*Corresponding author

John Cheung, Hong Kong SAR. ORCID: orcid.org/0009-0007-9777-4393.

Received: October 17, 2026; **Accepted:** October 21, 2026; **Published:** October 30, 2026

Keywords: Large Language Models, Mixture of Agents, Model Fusion, Inference-Time Compute, LLM Routing, Evaluation Methodology, Negative Results

Introduction

Combining several large language models at inference time is an appealing alternative to training a larger one. The Mixture-of-Agents (MoA) architecture arranges models in layers: proposers answer independently, and later agents read those answers and produce refined or aggregated responses [1]. MoA reported strong results on LLM-judged conversational benchmarks, and managed products now sell multi-model fusion as a feature.

The evidence for *why* such systems help is less settled. Self-MoA found that repeated samples from the single strongest model often beat panels that add weaker models [2]. Work on sampling and voting shows that more attempts alone can raise accuracy [3,4]. Routing and cascading research shows that much of the value lies in deciding *when* to call an expensive model [5-6]. Recent preprints argue that correlated failures cap what any member-selecting ensemble can achieve, and that equal-weight aggregation can drag a strong model toward weaker neighbours [7,8].

These results point to a confound in many MoA comparisons. A mixed panel usually differs from its baseline in at least four ways at once: more attempts, different model families, a selection or synthesis step, and sometimes a stronger aggregator. A gain over a single call does not say which of these produced it. Deployment adds a fifth axis — the latency, tokens, memory and failure handling that the extra calls cost.

Practitioners who serve open models locally, through Ollama, vLLM or other OpenAI-compatible servers, face this question directly. They need a runtime that bounds cost, and an evaluation harness that makes negative results as visible as positive ones. OpenFusion was built for that purpose. This paper describes its design and uses its published benchmark series to answer four research questions:

- **RQ1:** Can MoA-style workflows be offered as a bounded, auditable runtime that existing LLM clients can call without modification?
- **RQ2:** Under a registered protocol, does a fixed test-guided mixed-model workflow beat a strong same-model alternative on both accuracy and resources?
- **RQ3:** Does model diversity — as opposed to more attempts,

verification or a stronger model — explain the observed gains?

- **RQ4:** Where does the benefit come from, and how should a practitioner decide whether to deploy a mixed panel?

The paper makes four contributions:

- **A Bounded Orchestration Runtime:** OpenFusion exposes fourteen strategies behind `/v1/chat/completions`, enforces a hard administrator call ceiling before any task is scheduled, constrains model-generated plans to built-in strategies, and returns a public trace without hidden reasoning (Section 3).
- **An Evaluation Methodology in Code:** OpenFusion Lab implements a baseline ladder, Wilson intervals, paired tests, order-balanced judging with explicit abstention, panel-complementarity diagnostics, and cost totals that stay null unless complete (Section 3.4).
- **Registered and Negative Evidence:** I report one registered 500-task study with a positive result, a gated-escalation case study, and five controls that do not support a diversity effect, all with published artifacts and checksums (Section 4).
- **A Decomposition and a Decision Rule:** I separate MoA gains into candidate coverage and candidate recognition, and derive a five-step rule for practitioners (Sections 2.4 and 5).

Section 2 reviews MoA and related work. Section 3 describes OpenFusion. Section 4 reports the experiments. Section 5 discusses use cases, Section 6 limitations, and Section 7 concludes.

Background and Related Work

The Mixture-of-Agents Concept

MoA treats several LLMs as agents in a layered pipeline. In the first layer, *proposers* answer a prompt independently [1]. Each later layer receives the previous layer's responses together with an instruction to aggregate and improve them. A final *aggregator* writes the answer returned to the user. The design rests on an observation the authors call collaborativeness: a model often writes a better answer when shown other models' answers, even weaker ones.

Three properties make MoA attractive for practitioners. It needs no training or weight access, so it works across vendors through ordinary chat APIs. It can be assembled from open models that individually trail a frontier system. And its structure — independent generation, then aggregation — maps onto familiar ensemble ideas. Its costs are equally structural: two layers of three agents followed by a final aggregator spend seven calls per request, and aggregation prompts grow with every candidate.

OpenFusion's layered_refinement and parallel_synthesis strategies implement bounded versions of this information flow. They are not the original Together AI implementation, and I make no claim to reproduce its reported scores.

Same-model sampling, ranking and fusion

Self-consistency samples several reasoning paths from one model and takes a majority vote over final answers [3]. More Agents Is All You Need reports that accuracy scales with the number of instantiated agents under sampling-and-voting [4]. LLM-Blender separates ensembling into a trained pairwise ranker and a generative fuser; OpenFusion's pairwise_rank_fuse uses a prompted ranker instead, so the two are not equivalent [4].

Self-MoA is the closest precedent for our question [2]. It compared mixed panels with repeated samples from the strongest model and found that the latter often wins, because adding weaker

models lowers average candidate quality more than it adds useful diversity. A recent preprint measures a co-failure ceiling across 67 models: when every member fails on the same items, no member-selecting method can recover them [8]. Another describes *anchor corruption*, where naive aggregation pulls a reliable model toward weaker neighbours [9]. A third shows that correlated failures break the independence assumption behind majority voting [10]. These 2026 preprints are unreviewed; I cite them as hypotheses that our data can test.

Routing, Cascades and Test-Time Compute

Routing asks which single model should answer a query; cascading asks when to escalate from a cheap model to an expensive one. Hybrid LLM routes on predicted difficulty and reported fewer large-model calls at matched quality [5]. RouteLLM learns a strong-versus-weak router from preference data [6]. Dekoninck et al [7]. unify routing and cascading and identify per-query quality estimation as the critical component. Benchmark studies report that sophisticated routers often fail to beat simple baselines and that larger model pools show diminishing returns [9,10]. Snell et al. show that the best use of extra inference compute depends on problem difficulty, which argues for adaptive budgets rather than a fixed number of calls [11].

Two measurement problems recur across this literature. First, LLM judges are biased by position, verbosity and self-preference, so a judge-graded MoA gain may partly reflect the judge [12-14]. Second, verbalized confidence is often overconfident, so a cascade that trusts self-reported confidence needs held-out calibration [15]. OpenFusion's evaluation design responds to both (Section 3.4).

A Coverage–Recognition View of MoA Gains

I use a simple decomposition throughout. Consider tasks $t = 1 \dots T$ and a panel of k candidate generators, where $y_j(t) \in \{0, 1\}$ records whether candidate j is correct on task t . Let A^* be the accuracy of the best single member, and let the selection oracle O be the fraction of tasks on which at least one candidate is correct; $1 - O$ is the all-member co-failure rate. A workflow W that returns one member's answer satisfies $A_W \leq O$, and its gain over the best member splits into two terms:

$$A_W - A^* = \underbrace{\left(O - A^*\right)}_{\text{coverage headroom}} - \underbrace{\left(O - A_W\right)}_{\text{recognition loss}}$$

Coverage headroom is what the panel *could* add; recognition loss is what the selector fails to capture. Model diversity is useful only if it raises O by more than fresh samples from the strongest model would, *and* the selector can still recognize the extra correct answers. A generative synthesizer can in principle exceed O , so for synthesis O is a diagnostic reference rather than a ceiling. The decomposition makes clear that a mixed panel and a same-model panel must be compared at equal attempts and with the same selector before diversity can be credited.

OpenFusion: Design and Implementation

OpenFusion (v0.6.1, MIT licence) is an inference-time orchestration runtime [16]. It does not merge weights and does not train an orchestration policy. It sits between an OpenAI-compatible client and one or more OpenAI-compatible model servers, and turns a single chat request into a bounded multi-call workflow.

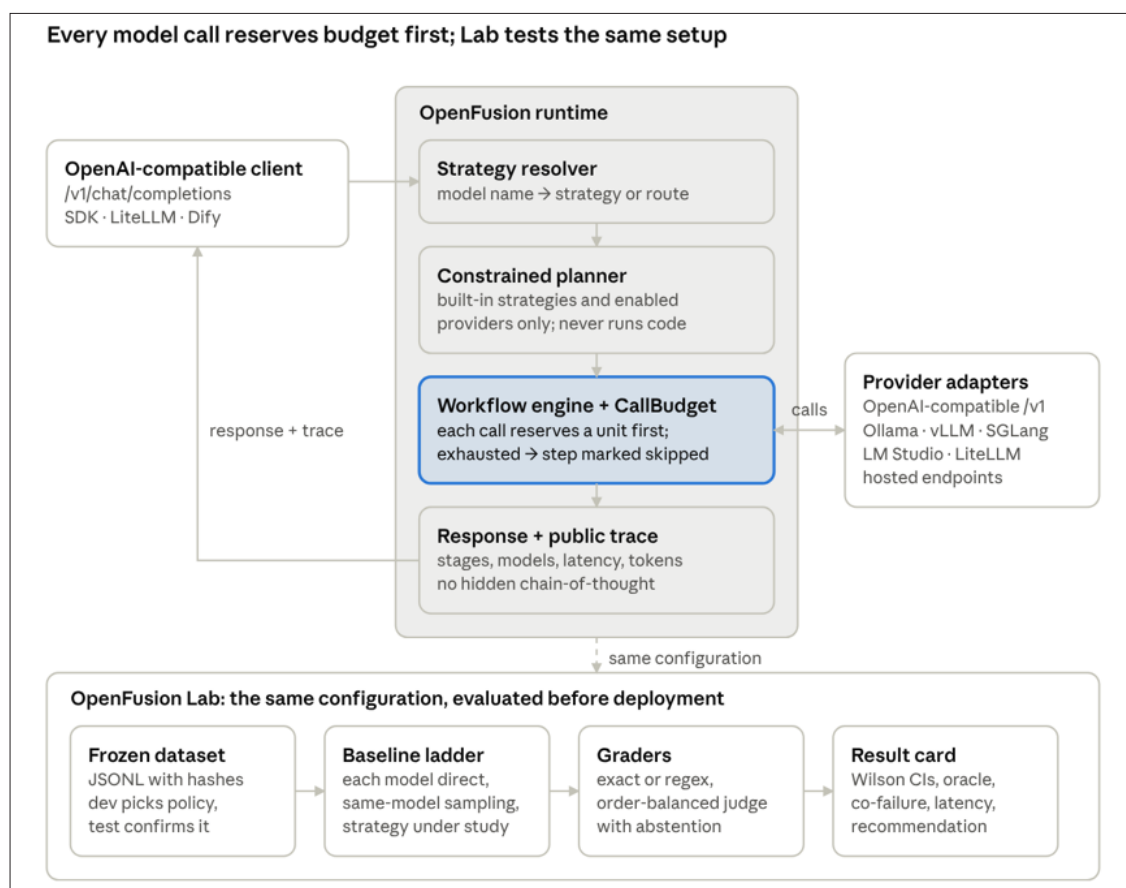
Design Goals

- **G1 Drop-in Compatibility:** Client's call `/v1/chat/completions` and `/v1/models` unchanged. A strategy is selected by model name (for example `openfusion/decision-select`), and direct

- provider/{name}/{model} routes bypass orchestration.
- **G2 Bounded Cost:** Every multi-call workflow obeys an administrator-set max_total_calls. Requests may lower the ceiling but never raise it.
- **G3 Auditability:** Each response carries a public trace of stages, models, call status, latency, tokens and optional estimated cost. Hidden chain-of-thought is never requested or exposed.
- **G4 Containment:** Candidate text is treated as untrusted input in every selector, critic and ranker prompt. Model-generated plans may only name built-in strategies and enabled providers; planner output is never executed as code, shell commands, URLs or tools.
- **G5 Falsifiability:** The same configuration that serves traffic can be evaluated against direct-model baselines, so a regression is visible before deployment.

Architecture and Request Path

Figure 1 shows the request path. A resolver maps the requested model name to a strategy, a planner produces a bounded plan, and the engine executes its stages against provider adapters. The OpenAICompatibleProvider adapter sends non-streaming requests to any /v1 base URL, reads credentials from environment variables, and pools HTTP clients by timeout. Local servers (Ollama, vLLM, LM Studio, SGLang), hosted endpoints and a LiteLLM proxy all appear as providers. When a workflow needs several candidates, providers are called independently with a neutral sampling instruction, so agents do not see peer answers before aggregation.



OpenFusion v0.6.1 Architecture · Request Path, Call Budget and Lab Loop

Figure 1: OpenFusion architecture. A request flows through the resolver and constrained planner to the workflow engine, where every provider call must reserve budget before it runs; Lab evaluates the same configuration against direct-model baselines before deployment

Hard Call Budget

Each orchestrated request receives a CallBudget object. A provider call must reserve one unit before it executes; when the reservation fails, the step is recorded as *skipped* rather than silently incurring cost. Candidate generation and pairwise comparisons are bounded before coroutines are created, including lazily generated comparison pairs. When the budget runs out before a selector or synthesizer call, Self-MoA strategies fall back to the best usable candidate by deterministic local scoring, and an adaptive plan is shrunk to fit the remaining calls. The budget limits calls, not tokens; token, rate and financial limits remain the responsibility of an upstream gateway.

Strategy Catalogue

Table 1 lists the fourteen strategies. They span the design space from Section 2: selection among independent answers, voting, same-model sampling, generative synthesis, MoA-style layering, critique and escalation.

Table 1: Strategies in OpenFusion v0.6.1. Every Call Counts toward max_total_calls

Strategy	Call shape	Aggregation	Intended use
fallback	Provider A, then B on failure	None	Operational baseline, failure recovery
best_of_n	n candidates + 1 evaluator	Strict JSON winner, returned unchanged	Selection among long answers
majority_vote, weighted_vote	n candidates	Normalized textual agreement	Concise or regex-extractable answers
semantic_vote	n candidates + bounded equivalence checks	Rule-only or LLM-judged equivalence groups	Short answers with varied wording
self_moa	n samples of one provider + selector or synthesizer	Select or synthesize	Strong single model
self_moa_seq	Batched samples with a carried running best	Sequential selection or fusion	Many or long samples
parallel_synthesis	Panel drafts + synthesizer	Generative synthesis, optional structured sections	Open-ended synthesis
pairwise_rank_fuse	Candidates + bounded pairwise or score ranking + fuser	Rank, then fuse top k	Long answers where voting is brittle
critique_revision	Drafts, critic, reviser	Separate critique and revision roles	Draft-review workflows
layered_refinement	Independent layer, refinement layers, synthesizer	MoA-style layered information flow	MoA experiments
uncertainty_cascade	Bounded provider sequence	Escalate on low confidence, disagreement, invalid format or failure	Latency- or cost-sensitive serving
decision_select	Panel + one decision-model call (System One API, e.g. Kev [28])	Probability over choices, answer returned unchanged	Choosing among plausible answers
adaptive	Heuristic or model planner, then one workflow above	Constrained JSON plan, heuristic fallback	Exploration

Two features support MoA experiments specifically. `panel_roles` gives independent panel calls different public instructions (for example a factual checker and an edge-case reviewer). With `structured_synthesis` enabled, a synthesizer returns parseable consensus points, contradictions, unique insights, missing information and a final answer; invalid structure degrades to plain text.

Evaluation Layer: OpenFusion Lab

The runtime is paired with an evaluation path (`openfusion evaluate`) and an experiment runner (`openfusion lab`). Both are designed to make a negative result as easy to read as a positive one.

- **Baseline Ladder:** Lab runs every provider directly, reports the best-observed single model, fallback, same-model sampling and the strategy under study on the same ordered examples, and reports deltas against each.
- **Grader Hierarchy:** Exact and regex graders are preferred. The optional `llm_pairwise_swap` grader judges a pair in both orders; a win, tie or loss is accepted only when both verdicts agree after remapping. Disagreement is recorded as *inconsistent* and malformed output as *abstain*, never silently as a tie. Records disclose whether the judge generated either candidate.
- **Uncertainty:** Accuracies carry two-sided 95% Wilson intervals with small-sample warnings [15]. The documentation requires paired tests such as McNemar's or a paired bootstrap for any comparison [16].
- **Panel Complementarity:** From the direct-model correctness matrix, Lab derives selection-oracle accuracy, all-member co-failure, pairwise disagreement and each provider's marginal oracle contribution - the quantities in Section 2.4.
- **Resources:** Latency is end-to-end wall time, not summed provider time. Reports include calls, prompt and completion tokens, p50/p95/p99 latency, failures kept in the denominator, and correct answers per call and per 1,000 tokens. An aggregate cost estimate is reported only when every executed call has usage and both configured prices; otherwise, it is null.
- **Result Cards:** Lab emits schema-versioned cards with dataset and configuration hashes and an objective-specific recommendation, including the explicit verdict that fusion is not recommended.

Implementation

The package is about 7,500 lines of Python (≥ 3.10) built on FastAPI, httpx and Pydantic, with roughly 4,000 lines of offline tests that never call external APIs. Continuous integration compiles, lints, tests, builds the package and validates the documentation in strict mode. Deployment guides cover Docker, LiteLLM and Dify. Experiment controllers for the studies in Section 4 live under `examples/` and use only the Python standard library plus a Docker sandbox.

Experimental Evaluation

This section reports nine studies run with OpenFusion and its experiment controllers between late September and early October 2026 [17]. Each result concerns its own tasks, models and workflow; scores from different studies must not be pooled. Some early task cohorts appear in more than one report, and I mark those comparisons as retrospective rather than counting them as replications.

Setting and Common Protocol

- **Hardware:** All studies ran on two Dell Pro Max workstations with the NVIDIA GB10 superchip (the "DGX Spark" class, unified memory), linked by ConnectX-7 RoCE. In the main studies a production SGLang tensor-parallel service stayed resident on both machines, and experiments used the remaining memory. Host drivers and kernels differed between the two nodes, and this difference is part of the tested deployment rather than a controlled variable.
- **Models:** The core panel was Qwen3-4B, Phi-4-mini (3.8B) and Gemma 3 4B, served by Ollama. The Lab case study added Llama 3.2 3B and 1B. A larger model served on SGLang acted as the "stronger endpoint"; its local alias is recorded in the artifacts, but its vendor release name was not verified. The memory-constrained study used Q4_K_M quantizations of a 27.3B Qwen-architecture model, Granite 8B, Gemma 12B and Qwen 8B.
- **Tasks and Verification:** Coding studies used MBPP, HumanEval and, according to the task identifiers in the published export, APPS; one early study used GSM8K [18-20]. Each coding prompt includes the function signature and one published test (the *visible* check). Workflows may run only that check; the remaining tests stay hidden until generation is complete and decide correctness. In the registered study every check ran in a fresh Python 3.12 container with no network, a read-only root, 256 MiB of memory, one CPU and an eight-second limit.
- **Statistics:** All comparisons are paired by task. Single-seed studies use exact McNemar tests [16]. Two-seed studies average the seeds within each task, report 95% task-cluster bootstrap intervals that keep both seeds together, and use two-sided whole-task sign-flip tests with 100,000 resamples. Registered families of contrasts use Holm correction [21]. Policies, thresholds and model orders were fixed on development tasks before any confirmation outcome was seen.

Table 2: Studies Reported in this Section

Study	Tasks	Seeds	Primary comparison	Evidence status
A. Lab case study	94 OFBench items (24 dev, 70 test)	1	Gated escalation vs best small model	Held-out after dev fitting
B. Verified MoA and Kev	60 MBPP; earlier 100 MBPP and 100 GSM8K	1	Verified MoA vs Qwen repair; selectors on identical candidates	Fresh; earlier cohorts retrospective
C. Diversity ablation	40 HumanEval; earlier 60 MBPP	1	Mixed vs repeated Qwen at three proposals	Fresh; MBPP retrospective
D. Registered deployment	500 MBPP	2	Fixed MoA vs Qwen repair	Registered, prospective
E1. Fresh-Qwen control	1,500 APPS-derived	2	Mixed vs fresh Qwen attempts	Registered gates
E2. No execution feedback	149 (58 dev)	2	Mixed vs Qwen-only synthesis	Dev-selected, held-out
E3. Wide policy search	120 (24 dev)	2	Selected mixture vs Qwen-only proposals	Dev-selected, held-out
E4. Similar-size 4B	70	2	Qwen + Gemma vs repeated Qwen or Gemma	Registered contrasts
E5. 24 GB memory proxy	160	2	Selected small pair vs same-model controls	Dev-selected, held-out

Study A: Naive Fusion Regresses, Gated Escalation Helps

The first study used OpenFusion Lab on OFBench, 94 deterministically generated items with programmatic answers across eleven subjects, under a ten-call budget. On the development split, equal-weight voting across all five small models scored 62.5%, against 79.2% for the best single model (Qwen3-4B); Lab reported that fusion was not recommended. Its complementarity report showed an oracle of 87.5%, with marginal oracle contribution concentrated in Qwen3-4B (+16.7 points) and Llama 3.2 1B (+8.3 points); the other three models added none.

Five-fold cross-validation confirmed the diagnosis. Equal-weight voting scored 67.0% against 76.6% for the best single model (2 wins, 11 losses; exact McNemar $p = 0.023$). Reliability weighting (75.5%) and subject-expertise weighting (76.6%) closed the regression but did not beat the anchor, while the oracle stood at 89.4%. Five-sample self-consistency changed the anchor's majority answer on zero items, because small-model errors on these programmatic items were systematic rather than random.

The productive move was escalation, not harder fusion. Policy $G(\tau)$ queries three local models, computes a reliability-weighted consensus, and calls the stronger endpoint once when consensus falls below τ . Fitting on the development split chose $\tau = 0.70$.

Table 3: Study A, Held-Out Test Split (n = 70)

Policy	Test accuracy	Stronger-model calls per item
Best single small model	75.7%	0.00
Equal five-model vote	62.9%	0.00
Gated escalation $G(0.70)$	91.4%	0.40
Stronger model always	98.6%	1.00

Gated escalation beat the best small model (11 wins, 0 losses; $p = 0.0010$). Against the stronger model always, the difference was not significant (1 win, 6 losses; $p = 0.125$), but equivalence was not established. The gate fixed 11 anchor errors, spent 17 escalations on items the anchor been already right, and missed 6. In the terms of Section 2.4, the small panel had 12.8 points of coverage headroom that no selector captured; escalation instead imported correct answers from outside the panel.

Study B: Verification, Not Semantic Selection, Closes The Gap

Study B froze 60 previously unused MBPP tasks and generated one deterministic proposal from each small model. Qwen alone solved 43, Phi 40 and Gemma 36. Choosing the first proposal that passed the visible check solved 51 — exactly the hindsight oracle, since at least one proposal was correct on 51 tasks. Selectors that saw no test output fell short: a Qwen LLM judge and the Kev decision model [28] each solved 45. Filtering by the visible check and then applying Kev matched rule selection at 51 (0 wins, 0 losses).

The complete workflow — proposals in order, first visible pass, else one Qwen repair — solved 51/60 against 45/60 for Qwen with up to three repairs (6 wins, 0 losses; $p = 0.031$) and 43/60 for Qwen alone (8–0; $p = 0.0078$). On an earlier, previously examined 100-task MBPP cohort the same workflow scored 82 against 78 for repair, a difference that was not significant (4–0; $p = 0.125$). On 100 GSM8K items, critique-revision scored 96 against Qwen's 95, giving no mathematics advantage.

Recognition loss was zero for the test-guided selector and six tasks for the semantic selectors. Kev's one useful intervention on the earlier cohort illustrates its niche: all three proposals passed the visible check for a trapezium median, but Qwen used integer division; Kev picked a candidate that passed the withheld

fractional case. An ordinary LLM judge did the same, so this does not establish that Kev is better than a conventional judge.

Study C: Diversity at an Equal Proposal Budget

Study C asked whether Study B's gain came from model diversity or simply from three attempts. On 40 fresh HumanEval tasks, it held the first answer, prompt, three-proposal budget and visible-check selector fixed, and replaced two Qwen samples with Phi and Gemma samples. Mixed proposals with Qwen first solved 37/40 against 36/40 for three Qwen answers (1–0; $p = 1$). With Phi first the scores were 34 against 30 (5–1; $p = 0.22$). The sampled mixed pool contained a correct answer on 38 tasks and the Qwen-only pool on 36; the selector recovered 37. Over all six fixed model orders, accuracy ranged from 32 to 36, showing that ordering matters whenever several answers pass an incomplete check.

The stronger SGLang endpoint scored 30/40 in one call; it hit its 768-token allowance, partly spent on reasoning, on 10 tasks. A separately declared length-triggered retry with 4,096 tokens raised it to 35/40. On the previously examined 60-task MBPP cohort, mixed proposals beat three Qwen answers 51 to 45 (6–0; $p = 0.031$), but that cohort was not fresh.

Study D: A Registered Prospective Deployment Test

Study D tested one workflow, chosen in advance, against the strongest same-model alternative identified earlier. The registration commit fixed the executable controller, analysis code, model manifests, task identities, protocol and decision rule before confirmation inference [27]. The cohort comprised 500 MBPP tasks never used for generation in the project, selected by fixed shuffle from 801 eligible candidates after excluding 168 previously used identities and exact prompt or syntax-tree matches. Each small-model workflow ran twice per task with different seeds.

Table 4: Study D workflows. All small-model workflows stop at the first answer that passes the visible check and allow 2,048 output tokens per call

Workflow	Fixed behaviour	Call ceiling
Single Qwen	One Qwen answer	1
Qwen repair	Qwen answer, then up to three repairs using visible-check feedback	4
Fixed MoA	Qwen answer, then independent Phi and Gemma attempts, then one Qwen repair of its first answer	4
Stronger endpoint	One answer on a preselected 40-task subset, reasoning disabled	1

A deployment recommendation required all five registered criteria: a positive lower 95% bound, sign-flip $p < 0.05$, an observed gain of at least five points, and mean latency and mean total tokens each at most 1.5 times Qwen repair. A sample-size calculation assuming paired discordance of 0.15 gave about 83% power to detect a five-point gain.

Table 5: Study D results on 500 tasks $\times$ 2 seeds (1,000 runs per arm). Latency includes generation, model loading, transport and visible checking

Workflow	Accuracy	Mean calls	Mean tokens	Mean s	Median s	p95 s
Single Qwen	69.0% (690)	1.000	250.8	2.21	1.25	4.85
Qwen repair	72.5% (725)	1.616	621.9	4.24	1.43	17.42
Fixed MoA	78.1% (781)	1.538	440.6	3.51	1.48	12.35

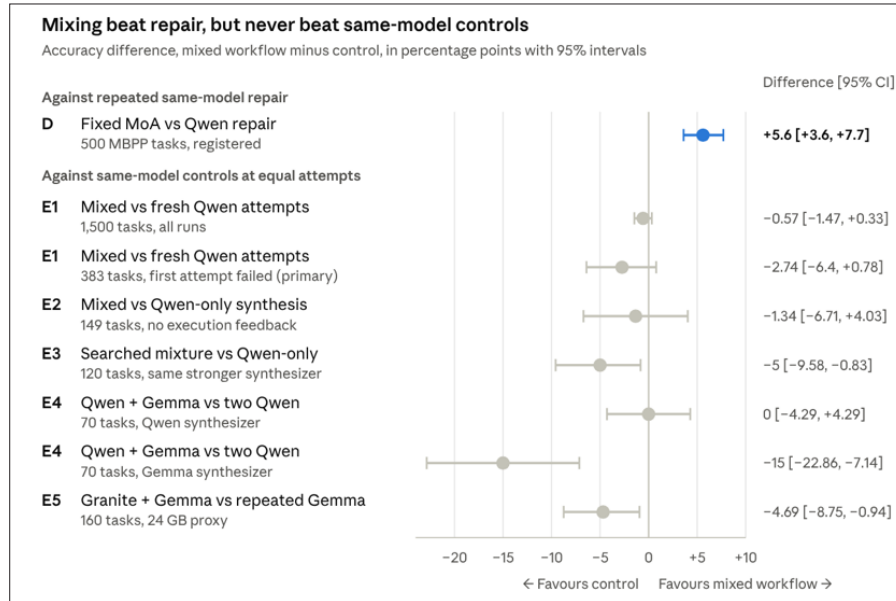
The fixed MoA workflow improved accuracy over Qwen repair by 5.6 points (95% CI [3.6, 7.7]; sign-flip $p = 0.00001$). Its mean latency ratio was 0.828 (CI [0.762, 0.905]) and its token ratio 0.708 (CI [0.654, 0.770]). All five criteria were met. The gain held at both seeds (389 vs 366 and 392 vs 359 of 500). At task level, 44 tasks favoured MoA, 8 favoured repair, and 448 tied. On the 40-task subset, the fixed workflow solved 35 against 33 for the stronger endpoint, a descriptive comparison too small to establish superiority.

Both multi-call arms start from a Qwen answer to the same prompt, so their differences arise in the roughly one run in five where that answer fails its visible check. The MoA workflow stopped at its first Qwen answer in 783 runs, at Phi in 30, at Gemma in 53, and reached the final repair in 134. Fresh attempts do not carry a failed program in the prompt, which is consistent with the lower token

use, though the study does not causally separate model diversity from prompt length or repair strategy. A structural audit verified all 3,040 generation and grading records for model order, budgets, early stopping, seeds, cohort membership and checksums.

Study E: Does Diversity Explain the Gain?

Study D's comparator was repeated repair, not fresh sampling from the same model. Five follow-up studies therefore compared mixed panels with same-model controls at equal attempt counts. Figure 2 summarizes their registered contrasts alongside Study D.



OpenFusion benchmark exports [21]. 8 registered contrasts, Studies D and E1–E5: task-cluster bootstrap intervals

Figure 2: Registered accuracy contrasts. Only the comparison against repeated repair (Study D) has an interval above zero; none of the seven equal-attempt diversity contrasts has a positive point estimate, and three lie wholly below zero

E1. Fresh-Qwen Control: On 1,500 tasks with two seeds, a workflow with Study D's structure (initial Qwen answer, two alternative attempts, final repair) was compared with the same workflow drawing its alternative attempts from fresh Qwen samples. Mixed selection scored 60.43%, fresh-Qwen selection 61.00% and single Qwen 56.37%. The unconditional difference was -0.57 points (95% CI [-1.47, 0.33]; $p = 0.24$). The registered primary analysis, restricted to 753 task-seed pairs across 383 tasks where the first attempt failed, gave 16.58% against 19.32% (-2.74 points, CI [-6.40, 0.78]; $p = 0.16$). A recovery from aborted requests left resource usage unknown, so latency and token claims were suppressed. The export also shows how often a visible pass hid a failure: 23.5% of visible passes failed hidden tests for single Qwen, 25.3% for fresh Qwen and 28.4% for the mixed workflow.

E2. Synthesis without Execution Feedback: A synthesis policy was chosen on 58 development tasks from selection, synthesis and critique-revision, then confirmed on 149 tasks with two seeds. Mixed proposals scored 58.05% against 59.40% for Qwen-only proposals (-1.34 points, CI [-6.71, 4.03]; $p = 0.71$); direct Qwen scored 56.38%.

E3. Wide Policy Search: Thirty-two configurations, crossing proposal style, aggregator model and aggregation method, were searched on 24 development tasks. The winner — independent proposals with stronger-model synthesis — led Qwen-only proposals by 10.4 points on development data. On 120 held-out tasks with two seeds it scored 60.0% against 65.0% for Qwen-only proposals with the same synthesizer (-5.0 points, CI [-9.58, -0.83]; $p = 0.043$). It beat direct small Qwen (45.4%) but not the stronger model used directly (61.7%), and failed the registered rule.

E4. Similar-Size 4B Comparison: Seventy tasks with two seeds crossed proposal pairs with two synthesizers, without execution feedback.

Table 6: Study E4 accuracy by proposal pair and synthesizer. Direct Qwen: 33.57%; direct Gemma: 23.57%

Synthesizer	Two Qwen proposals	Qwen + Gemma proposals	Two Gemma proposals
Qwen	40.71%	40.71%	34.29%
Gemma	39.29%	24.29%	17.14%

With Qwen synthesis, mixing made no difference (0 points, CI [-4.29, 4.29]; Holm $p = 1$). With Gemma synthesis, the mixed pair lost 15 points to two Qwen proposals (CI [-22.86, -7.14]; Holm $p = 0.0025$). Twenty-two Gemma failures with unknown usage, a runtime-version mismatch across workers and four extra attempts limit interpretation; worst-case retry bounds do not change the sign of either contrast.

E5. 24 GB Memory Proxy: This study asked which workflow a user with a 24 GB budget should run. Policies were frozen on development data, then 160 held-out tasks were run with two seeds: 1,920 workflows and 5,760 calls under a 600-second ceiling per workflow, with no retries.

Table 7: Study E5 Accuracy on 320 Runs Per Workflow. Failures Stay in the Denominator

Workflow	Accuracy	Median wall s	Generation failures
27B Qwen, two proposals + synthesis	71.25%	196.1	0
27B Qwen direct	63.13%	69.3	0
Gemma 12B, three proposals + synthesis	33.44%	48.1	11
Gemma 12B, two proposals + synthesis	33.13%	37.8	12
Qwen 8B + Granite 8B + Gemma 12B, Gemma synthesis	32.81%	46.7	2
Granite 8B + Gemma 12B, Gemma synthesis (selected)	28.44%	32.3	3

The selected heterogeneous pair lost to repeated Gemma by 4.69 points (CI $[-8.75, -0.94]$; Holm $p = 0.035$), and to the 27B model direct and repeated by 34.7 and 42.8 points. Every contrast favoured the control at both seeds. The machines use unified memory, so the study is a feasibility proxy: sampled device-global memory peaked at 22.06 GB, system swap rose in 84 workflows, and strict no-swap compliance was not established.

Summary of Findings

The studies answer RQ2 and RQ3 differently. For RQ2, the registered workflow met every accuracy and resource criterion against repeated repair on short, test-guided Python tasks. For RQ3, no study showed that model diversity caused a gain. Across the seven equal-attempt diversity contrasts from five studies in Figure 2, no point estimate is positive and three intervals lie wholly below zero. The only positive diversity contrasts came from Study C: a one-task difference on fresh HumanEval and a six-task difference on a previously examined MBPP cohort.

Read through the coverage–recognition lens, four mechanisms account for the observed gains and losses:

- **Verification Converts Attempts into Selections:** An executable visible check drove recognition loss to zero in Study B, while semantic selectors lost six of 60 tasks.
- **Fresh Bounded Attempts Beat Accumulated Repair:** In Study D, restarting from a clean prompt used 29% fewer tokens than repair and solved more tasks; in E1, fresh Qwen attempts scored slightly, though not significantly, higher than attempts from other models.
- **Capability Can be Imported, Not Averaged:** Escalation (Study A) and a stronger synthesizer (E3) raised accuracy by adding a better model, not by mixing peers.
- **Equal-Weight Aggregation of Unequal Models Hurts:** Equal voting cost 10–17 points in Study A, and a weaker synthesizer erased a strong proposer's contribution in E4 — consistent with the anchor-corruption hypothesis [10].

Visible checks are also imperfect. In E1 roughly a quarter of visible passes failed hidden tests, and the share was highest for the mixed workflow, which suggests that more diverse candidates may more often satisfy an incomplete check without being correct.

Use Cases and Deployment Guidance

The evidence supports some uses of OpenFusion, motivates others, and leaves several untested. Table 8 maps common workloads to a starting configuration and the evidence behind it.

Table 8: Workload Guidance Derived from Section 4

Workload	Start with	Then compare against	Evidence status
Short code with executable checks	Strong single model plus fresh same-model attempts, stopping at the first pass	The Study D mixed workflow	Supported against repair (D); diversity not established (E1)
Objective short answers on a local panel	Best single model, chosen on development data	Gated escalation to a stronger model	Promising (A); equal voting regressed
Code without executable checks	Strong single model	Same-model proposals with synthesis	Mixed panels tied or lost (E2–E4)
Fixed memory budget, e.g. 24 GB	Largest model that fits	Repeated sampling of that model	Large model dominated small mixtures (E5)
Open-ended writing, review, specialist teams	Strong single model	parallel_synthesis or critique_revision, with audited judging	Not evaluated

Granite 8B + Gemma 12B, Gemma synthesis (selected)	28.44%	32.3	3
---	--------	------	---

Test-Guided Programming Tasks

The strongest supported setting has a known function interface, a meaningful supplied check, bounded alternative attempts and early stopping. Code assistants that generate small functions against unit tests, data-transformation snippets with example rows, and teaching tools with autograders fit this shape. In Study D such a workflow beat repeated repair while using less time and fewer tokens. Two cautions apply. First, E1 suggests that fresh attempts from the same strong model capture most of the benefit, so a single-model deployment may be simpler. Second, OpenFusion's runtime strategies deliberately do not execute candidate code; checks should run in an isolated sandbox owned by the application, as they did in the experiment controllers.

Local-First Serving with Selective Escalation

Organizations that keep most traffic on local hardware for cost, latency or data-residency reasons can use small models by default and call a stronger model only when local agreement is weak. In Study A this raised held-out accuracy from 75.7% to 91.4% with stronger-model calls on 40% of items. The trade-off counts three local calls per item in addition to the escalations, and the policy was not compared with a simpler difficulty router. `uncertainty_cascade` provides the runtime pattern; its threshold must be calibrated on held-out data because self-reported confidence is often overconfident [15].

Falsifying a Panel Before Deployment

The most dependable use of OpenFusion may be as an evaluation harness. Lab runs the baseline ladder on the user's own data, reports Wilson intervals and complementarity, and states when fusion is not recommended. In Study A this report located the problem — complementary members but a poor aggregator — and redirected the work toward escalation. Result cards with dataset and configuration hashes make such decisions auditable and repeatable when models or prices change.

Drop-in Orchestration in Existing Stacks

Because OpenFusion speaks the OpenAI chat API, it can sit behind existing SDKs, a LiteLLM proxy or a Dify application without client changes. Even without fusion, fallback provides failover across providers, the hard call ceiling acts as a cost guardrail, and the public trace supports audit. These operational benefits do not depend on any accuracy claim.

Unvalidated Uses

Research writing, brainstorming, security review, specialist role teams and multilingual tasks are plausible MoA applications that this benchmark series did not test. Runtime support for a strategy is not evidence of quality. Such workloads lack objective graders, so evaluations should use blinded human review or order-balanced LLM judging with abstention and self-judge disclosure.

A Practical Selection Rule

The evidence supports a conservative procedure:

- Select the strongest suitable single model on development data.
- Measure fresh repeated sampling of that model with the same verification and stopping rules.
- Add another model only when a held-out comparison shows a useful improvement over step 2.
- Count loading, synthesis, failures and every model call in

the resource accounting.

- Keep a hard call cap, and report uncertainty and protocol deviations.

A hindsight pool that contains a correct answer shows only potential. A deployable selector must recognize that answer without hidden labels, so the complete workflow — not the panel — is the unit to evaluate.

Limitations and Threats to Validity

The positive results in this paper are narrow, and several limitations bear directly on how far they travel. I group them by source.

Limitations of the Software

- **The Main Positive Workflow Lives Outside The Runtime:** Study D's gain depended on executing a visible check and repairing on failure. OpenFusion's runtime strategies deliberately do not execute candidate code, so that workflow ran in a separate controller. Users must supply their own sandboxed verifier to reproduce it in production.
- **Calls, Not Compute:** `max_total_calls` bounds calls, not tokens, latency or money. Two workflows with equal ceilings can differ widely in realized cost, as Study D's token ratio shows.
- **No Staged Streaming:** Streaming responses are emitted after the workflow completes, so time to first token equals full workflow latency.
- **Simple Aggregation Primitives:** Majority and weighted votes compare normalized text and suit only short answers. `semantic_vote` is bounded by a pair budget. The adaptive planner uses readable heuristics or a constrained JSON plan and is not a learned or online orchestrator.
- **Partial Cost and Confidence Accounting:** The cost estimator does not price cached, reasoning or image tokens. `uncertainty_cascade` parses verbalized confidence, which needs held-out calibration before it can be trusted.

Internal Validity

Serving caches were not reset between jobs, so later requests may have benefited from earlier ones; randomized job order reduces but does not remove this bias. The two hosts ran different drivers and kernels beside a resident production service. Candidate order was fixed, and Study C shows order alone moved accuracy by four tasks out of 40. Several follow-ups had protocol deviations: E1 lost resource accounting for aborted requests, E4 had worker failures, a runtime-version mismatch and extra attempts, and E5 could not enforce a no-swap condition. Early cohorts were reused across reports, so their positive results are retrospective. Finally, E3 shows a winner's curse: the configuration that led by 10.4 points on 24 development tasks trailed by 5.0 points on held-out data.

Construct Validity

Correctness means passing a bounded set of hidden tests, not full program correctness; some studies used at most eight tests per task. In E1 about a quarter of visible passes failed hidden tests. Token counts come from different tokenizers, and latencies are provider-reported or end-to-end wall times, not GPU cycles, energy or invoiced cost. The stronger endpoint's vendor identity was not verified, and memory measurements on unified-memory hardware are proxies for discrete-GPU VRAM.

Most importantly, "MoA" in Studies B–E means test-guided candidate generation, selection, repair or synthesis among a small panel. I did not replicate the original layered MoA architecture [1] on LLM-judged conversational benchmarks, and our results do not refute its reported gains there.

External Validity

The panels used 1B–27B open models, mostly 4B-class; frontier hosted models were used only as an escalation target or not at all. The tasks were mostly short Python functions from public datasets that may appear in pretraining data, evaluated on custom held-out splits rather than official leaderboards. OFBench is synthetic and small. All studies ran in one laboratory on one hardware class, with specific model versions and quantizations. Results may change for longer programs, other languages, open-ended tasks, larger or more diverse panels, or hosted APIs.

Statistical Conclusion Validity

Studies A–C used 40–70 tasks and had low power; their secondary p-values are unadjusted for multiplicity. Two seeds per task are robustness checks, not independent replications, and bootstrap intervals are marginal. A non-significant diversity contrast is not evidence of equivalence: E1's primary interval, $[-6.40, 0.78]$ points, still admits a small benefit. Study D's five-point threshold applies to the point estimate and does not establish that the true gain exceeds five points.

Independence and Reproducibility

The software, studies and analyses were produced by the OpenFusion maintainer. Registration commits, checksums, structural audits and the publication of negative results reduce the risk of selective reporting but do not replace independent replication. The registered study releases raw responses, grades and analysis scripts; the follow-up studies release aggregate analysis exports with source hashes but not all raw responses. Reading the released answers turns those tasks into development data, so any new confirmation needs an untouched cohort.

Conclusion and Future Work

Mixing models paid off in this study only when something other than diversity did the work. OpenFusion shows that MoA-style workflows can be offered as a bounded, auditable runtime behind the standard chat API (RQ1). Under a registered protocol, a fixed test-guided mixed workflow beat repeated same-model repair by 5.6 points on 500 unseen tasks, with lower latency and fewer tokens (RQ2). Yet across five controls, mixed panels never beat fresh sampling or same-model proposals at equal attempts, and three controls favoured the same-model control (RQ3). The gains I can attribute come from executable verification, fresh bounded attempts and selective escalation to a stronger model, while equal-weight aggregation of unequal models consistently hurt (RQ4).

For practitioners, the implication is to treat a mixed panel as a hypothesis to falsify against a strong single model and its own repeated samples, not as a default upgrade. For researchers, it is to report MoA results against equal-attempt same-model baselines, with paired uncertainty and complete resource accounting.

Future work follows from the limitations:

- **Runtime-Native Verification:** A sandboxed check stage inside the runtime would let test-guided workflows run under the same call budget and public trace.
- **Richer Budgets:** Token-, latency- and cost-denominated limits alongside the call ceiling, and difficulty-adaptive

allocation in the spirit of compute-optimal test-time scaling [13].

- **Calibrated Escalation:** Learned or calibrated gates compared head-to-head with simple difficulty routers [6,7,12].
- **Generated Checks:** Automatically generated tests, evaluated for their own accuracy, cost and false rejections.
- **Beyond Code:** Grounded question answering, extraction and open-ended synthesis with blinded human evaluation, including a replication of the original layered MoA setting [1].
- **Independent Replication:** Multi-site studies on untouched cohorts, with frontier hosted models and other hardware.

Data and Code Availability

OpenFusion v0.6.1 is available under the MIT licence at github.com/johncheungmk/openfusion, with documentation at johncheungmk.github.io/openfusion. The registered study (Study D) is fixed by registration commit `ca2c9df64d656616c8bd6bade979e21694248d99`; its artifacts include public task inputs, grading tests, every model response, selection and timing records, per-test grades, the analysis, the structural audit and SHA-256 checksums. The published analysis can be reproduced without model access:

```
python examples/prospective_analysis.py results.json --output analysis.json
```

```
python examples/prospective_audit.py results.json --output audit.json
```

Studies B and C publish per-task proposals, decisions and test outcomes with replay scripts (data SHA-256 `b6fec879...` and `f5b158a7...`). Follow-up studies E1–E5 publish aggregate analysis exports with source and published-byte hashes in a manifest. Study A's configuration, dataset splits and result card are under `examples/` and `results/`. Private endpoints and credentials are omitted from all artifacts.

References

1. Wang J, Wang B, Athiwaratkun C, Zhang J, Zou J, (2025) Mixture-of-Agents Enhances Large Language Model Capabilities. ICLR.
2. Li W, Lin Y, Xia M, Jin C, (2025) Rethinking Mixture-of-Agents: Is Mixing Different Large Language Models Beneficial? arXiv preprint arXiv:2502.00674.
3. Jiang D, Ren X, Lin Y, (2023) LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. ACL.
4. Wang X, Wei J, Schuurmans D, Le Q, Chi E, Narang S, Chowdhery A, Zhou D, (2023) Self-Consistency Improves Chain of Thought Reasoning in Language Models. ICLR.
5. Li J, Zhang Q, Yu Y, Fu Q, Ye D, (2024) More Agents Is All You Need. Transactions on Machine Learning Research.
6. Ding D, Mallick A, Wang C, Sim R, Mukherjee S, Rühle V, Lakshmanan LVS, Awadallah AH, (2024) Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. ICLR.
7. Ong I, Almahairi A, Wu V, Chiang WL, Wu T, et al. (2025) RouteLLM: Learning to Route LLMs with Preference Data. ICLR.
8. Dekoninck J, Baader M, Vechev M, (2025) A Unified Approach to Routing and Cascading for LLMs. ICML.
9. Hu QJ, Bieker J, Li X, Jiang N, Keigwin B, et al. (2024) RouterBench: A Benchmark for Multi-LLM Routing System. arXiv preprint arXiv:2403.12031.
10. (2026) LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing. Findings of the Association

- for Computational Linguistics (ACL).
11. Snell C, Lee J, Xu K, Kumar A, (2025) Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Model Parameters. ICLR.
12. Zheng L, Chiang WL, Sheng Y, et al. (2023) Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. NeurIPS Datasets and Benchmarks.
13. Panickssery A, Bowman SR, Feng S, (2024) LLM Evaluators Recognize and Favor Their Own Generations. NeurIPS.
14. Shi L, Ma C, Liang W, Diao X, Ma W, et al. (2025) Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge. IJCNLP-AAACL.
15. Xiong M, Hu Z, Lu X, Li Y, Fu J, et al. (2024) Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. ICLR.
16. (2026) When Does Combining Language Models Help? A Co-Failure Ceiling on Routing, Voting, and Mixture-of-Agents Across 67 Frontier Models. arXiv preprint arXiv:2606.27288.
17. (2026) From Talking Words to Sharing Thoughts: Scalable Multi-LLM Aggregation via Structured Message Passing. arXiv preprint arXiv:2606.00405.
18. (2026) The Consensus Trap: Rescuing Multi-Agent LLMs from Adversarial Majorities via Token-Level Collaboration. arXiv preprint arXiv:2604.17139.
19. Austin J, Odena A, Nye M, et al. (2021) Program Synthesis with Large Language Models. arXiv preprint arXiv:2108.07732.
20. Chen M, Tworek J, Jun H, et al. (2021) Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374.
21. Hendrycks D, Basart S, Kadavath S, et al. (2021) Measuring Coding Challenge Competence with APPS. NeurIPS Datasets and Benchmarks.
22. Cobbe K, Kosaraju V, Bavarian M, et al. (2021) Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168.
23. Wilson EB, (1927) Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association* 22(158):209-212.
24. McNemar Q, (1947) Note on the Sampling Error of the Difference between Correlated Proportions or Percentages. *Psychometrika* 12(2):153-157.
25. Holm S, (1979) A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6: 6570.
26. Cheung J, (2026) OpenFusion: Open-Source Multi-Model Orchestration and Fusion Runtime. Software Repository, version 0.6.1, released 29 September 2026.
27. Cheung J, (2026) OpenFusion Benchmark Reports. Software Benchmark Reports.
28. Palmer J, (2026) Kev: A Decision Model for Selecting among Candidate Answers. Software Repository, checkpoint jaredpalmer/kev-4b.
29. Efron B, (1979) Bootstrap Methods: Another Look at the Jackknife. *Annals of Statistics*, 7: 1-26.

Copyright: ©2026 John Cheung. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.