不是更大的大脑，而是更好的调度：Fugu 想把多模型协作变成一台机器

基于 Sakana AI 的《Sakana Fugu Technical Report》整理与研究。

如果把今天最强的语言模型比作“天才选手”，那 Sakana Fugu 想做的事情很不一样：它不是再训练一个更大的天才，而是训练一个会派活、会换人、会接力的“总调度”。遇到写代码，它不一定永远叫同一个模型；遇到数学、科学、长上下文或者多轮工具调用，它会动态决定谁先上、谁来兜底、谁负责最后收口。**Fugu 真正想卖的，不是单个模型更聪明，而是“组织模型”的能力也可以被学会。**

这听起来像多模型路由器，甚至像简单的投票系统，但报告的野心更大。Sakana AI 试图证明：如果 frontier 模型之间已经出现了明显分工，那么下一步的规模化不只来自参数更多、算力更大，还来自把“怎么组合这些专长”变成一个可训练对象。换句话说，模型本身不再是唯一的生产单位，协作结构也开始成为能力的一部分。**这是一条把“能力”从单体属性改写为系统属性的路线。**

一、先回答一个很朴素的问题：为什么不能只用一个最强模型

在很多工程任务里，问题并不在于“有没有一个模型能做”，而在于“哪个模型更像这个任务要的那种脑子”。有的模型擅长写可运行代码，有的更擅长规划和拆解，有的对科学问答、视觉图表、工具调用或者长上下文保持更稳。今天的大模型生态越来越像一个分工明确的工厂，而不是一个人人都能胜任的天才班。**Fugu 的出发点就是承认这种分工已经存在，并把它当成设计前提，而不是瑕疵。**

最容易理解的类比不是“多模型”，而是“分诊台”。医院里，分诊护士不是替医生完成治疗，而是判断该把病人送到哪一科、谁先看、谁需要会诊。Fugu 做的事情很像这个动作，只不过它处理的不是病人，而是问题；不是科室，而是模型；不是一次性分

流，而是根据任务动态设计一条多步骤的工作流。类比成立的地方在于“先判断由谁处理”，失效的地方在于 Fugu 还要处理工具、记忆、反馈和多轮交互，远比一个静态分诊表复杂得多。

如果把问题再简化一层，它其实接近我们熟悉的“转接电话”或“客服分层”。普通客服先判断是技术问题、账单问题还是投诉，再把你转给专员。普通路由器只做一次转接，好的调度系统则会在过程中不断修正判断：先给谁答、答到哪一步再换人、是否需要让第二个专家复核。Fugu 的新意在于，它不是人工写死这套规则，而是让一个语言模型自己学会这套规则。**被学习的对象不是答案本身，而是答案是怎么被组织出来的。**

二、Fugu 到底是什么：一个会选人的模型，而不是一个会长篇输出的模型

报告里有两个版本。Fugu 追求低延迟，面向日常交互；Fugu-Ultra 追求极致质量，愿意为了更高答案质量付出更多时间。两者都属于“learned orchestrator”，可以理解为“学出来的编排器”。它们面对用户时看起来像单个模型，内部却会接入一个由多个 frontier worker 组成的池子。用户提问后，系统先判断该让谁出手，再决定要不要接力、复核、合成。

这里最关键的一点，是 Fugu 没有把“协作”理解成固定模板。很多多代理系统要么先开会再投票，要么按固定流程轮流讨论，要么永远由同一个模型做总编。Fugu 想做的，是让编排本身适应问题。一个问题也许只需要一次路由，另一个问题则需要先用 GPT 型模型搭框架，再叫 Opus 型模型查漏洞，最后由 Gemini 型模型做聚合。**它试图学习的不是“谁最强”，而是“在什么情境下谁最合适”。**

从架构上看，Fugu 的路由器非常轻。报告说它在 backbone 后面接了一个轻量 head，直接读隐藏状态 h，输出每个 worker 的 logits，然后选择一个模型。它不先生成冗长的中间文本再让文本去决定路由，而是尽量在内部状态上直接做决策。这个选择很重要，因为它把 orchestration 的成本压低了：如果路由器自己也得先说一大段话，那编排系统会变成一个更慢、更贵的聊天模型。**Fugu 的思路是：先把“派谁上”这个决策做快，再把“谁来完成”交给真正擅长干活的模型。**

报告还提到一种更省参数的做法：只微调部分矩阵的奇异值尺度，而不大动 backbone 的全部权重。这个细节看起来很工程，但它透露了整个设计哲学：编排器不需要把自己训练成另一个通用大模型，只需要把“判断力”磨得足够好。它像一个经纪人、导演或首席调度员，核心不是亲自上场，而是把资源调配得更精确。**这是一种把智能从“产出内容”转移到“配置流程”的写法。**

三、它怎么学会“派活”：先学评分，再学流程

Fugu 的训练分成两段，这个顺序很有意思。第一段是监督微调，第二段是进化策略；Fugu-Ultra 则再往前走一步，用强化学习把多模型流程直接学出来。这个顺序的背后，是一个很老练的判断：如果你一开始就让系统在长流程里试错，它会很慢，也很不稳定；先让它学会粗略分辨“哪个 worker 更像这类题”，再让它在复杂任务上优化工作流，效率会高得多。**先学路由，再学编排，是这套系统最稳的骨架。**

第一阶段的监督信号来自单步任务。报告做法不是简单地给每题贴一个“最佳模型”标签，而是让池子里的每个 worker 多跑几次，算出平均表现，再把这些分数变成一个 soft target distribution。换句话说，训练标签不是“某个模型永远对”，而是“这道题上，哪些模型更可能对、差距多大”。形式上可以写成：

$$p_i(j) = \frac{\exp(\bar{r}_{i,j}/\tau)}{\sum_{j'=1}^{K} \exp(\bar{r}_{i,j'}/\tau)}$$

再用 KL 散度去拟合路由器输出。这个设计的妙处在于，它把“近似同样好”的模型保留下来了，而不是粗暴地压成一个赢家。**这比硬分类更像真实世界，因为真实世界里的很多决策不是选唯一正确答案，而是选一个“差不多最合适”的专家。**

这一步还透露出另一个更深的逻辑：Fugu 学的不是知识本身，而是“知识在不同模型之间如何分布”。如果一个 worker 在数学上强、另一个在代码修复上强，路由器需要学会的是这种能力地图。它像一个掌握全院医生专长的总护士长：知道谁能看急诊，谁适合做二诊，谁更擅长处理复杂并发症。这里的核心洞见很朴素，也很关键：**专家系统的价值，不只在专家个体，更在专家之间的边界被识别出来。**

第二阶段变成了端到端任务上的进化优化。报告把真实多轮轨迹拿进来，包含编码环境、工具调用、执行反馈和最终完成情况，然后用 sep-CMA-ES 去直接优化终局奖励 $J(\theta) = \mathbb{E}[R(\tau)]$。为什么这里不用普通监督学习？因为长流程里的“对不对”往往是稀疏的、延迟的、还带噪声：你在第三轮选对了专家，不代表第七轮就一定赢。进化策略适合这种情况，因为它不要求每一步都有清晰标签，而是直接比较整条轨迹的好坏。**当局部标签不好构造时，直接优化最终结果反而更诚实。**

这也是报告最像工程系统的地方。它没有假装自己掌握了所有中间正确动作，而是接受“只看终局”的现实。对于复杂交互，尤其是软件工程、工具链和环境反馈，很多错误不是因为模型不会，而是因为它在错误时机调用了错误专家。进化策略提供了一种粗糙但有效的搜索方式，让编排器在“多轮交互”而不是“单题分类”的意义上被打磨。**这一步让 Fugu 从一个路由器，开始接近一个会自我修正的行动系统。**

四、Fugu-Ultra 为什么更像“导演”，而不是“路由器”

如果说 Fugu 是在做“选谁来答”，Fugu-Ultra 做的就是“怎么让多人一起答”。它建立在 Conductor 框架上，输出的是自然语言形式的 agentic workflow：先把任务拆成几个子任务，再决定每个子任务由哪个 worker 处理，最后指定这些子任务的结果如何流向下一步。这里的关键变化在于，系统不再只是“选一个模型”，而是在学“编排拓扑”。**它关心的是合作结构，而不是单点胜负。**

这可以把直觉进一步类比成乐团。路由器像指挥台上的接线员，负责把小提琴转给小提琴，钢琴转给钢琴；Fugu-Ultra 更像真正的指挥，知道什么时候让弦乐先开主题，什么时候让木管接一句，什么时候让低音部做支撑。它甚至允许“总编”自己也成为工作流中的一个 worker，这意味着系统可以把更高层的判断放到流程内部迭代，而不是外面硬塞一个固定总控。**这不是简单的多轮投票，而是把“谁在什么时刻说什么”变成可学习的结构。**

Fugu-Ultra 训练时用 GRPO 来直接优化工作流质量，并且有一个非常值得注意的奖励设计：如果 workflow 不能被解析，奖励就是 0；如果格式对但结果不对，则给一个中间值。这个设计让模型不仅要“想对”，还要“写对”。在多代理系统里，格式不是边角料，而是控制流本身。一个 workflow 写得再聪明，只要结构无法解析，整条链路就断了。**这提醒人们：在复杂 agent 系统里，语法就是工程，工程就是能力。**

更重要的是 Fugu-Ultra 处理了一个多代理系统里经常被忽略的难题：记忆隔离与共享记忆之间的张力。完全隔离会让后续 agent 不知道前面发生了什么，容易重复劳动；完全共享又会造成 orchestration collapse，先上场的模型把所有后续模型都带进同一条轨道，后来的专家失去独立判断空间。Fugu-Ultra 的做法是：在工作流内部隔离，在跨工作流层面共享必要记忆。**这其实是在回答一个很第二层的问题：协作系统到底该共享什么，保留什么独立性。**

五、它真的强在哪里：不是“平均更强”，而是“在不同题目上换脑子”

报告最吸引眼球的结果，是 Fugu-Ultra 在一组基准上的表现。以软件工程和代理编码为例，它在 SWE-Bench Pro 上达到 73.7，在 Terminal Bench 2.1 上达到 82.1；Fugu 则分别是 59.0 和 80.2。对比几家 frontier 模型，报告认为这已经到了“整代改进”的级别。更重要的是，这些分数不是单纯堆次数或堆 token，而是来自更细粒度的调度：同一个任务里，模型会在关键节点切换 worker。**优势不是来自某个模型无所不能，而是来自它知道何时让谁上场。**

在科学和图表理解上，Fugu 也给出了强结果。它在 GPQA Diamond 上做到 95.5，在 CharXiv Reasoning 上 86.6；在 LiveCodeBench 上 93.2，在 LiveCodeBench Pro 上 90.8。报告特别强调，Fugu 会把 GPT 型模型放到数学和物理推理里，把 Gemini 型模型更多用于某些科学和多模态任务，把 Opus 型模型更常用于调试和代码质量检查。这个分工听上去像经验规则，但 Fugu 的关键在于它把经验规则变成了可以随题目自适应变化的策略。**它不是一份固定的专家名单，而是一种会随任务重排的专家图谱。**

报告里有一个很能说明问题的细节：在 Terminal Bench 2.1 的某些长链路任务里，Fugu 先让 GPT 负责搭建，再在关键调试点换成 Opus；也有相反情形，先用 Opus 找漏洞，再让 GPT 从头重审问题，把看似服务器侧的症状重新定位成客户端并发 bug。这个模式的价值不在于“哪个模型更强”，而在于“强点在不同阶段如何接上”。**复杂问题常常不是缺一个天才，而是缺一个会安排天才的过程。**

更有意思的是它的一些非常规任务。比如自动研究优化 AutoResearch，Fugu-Ultra 在 123 次自主实验后把验证 BPB 做到 0.9774 的平均最好值，优于多个单模型基线。比如古典日文书信的阅读顺序恢复，Fugu-Ultra 在自建的小数据集上用同样的 beam search 取得最好 NED。再比如 CAD 机械光圈、盲fold chess、在线股票交易，这些任务都不只是“答题”，而是要持续维护状态、修正路径、在反馈中重新决策。**这些任务共同说明，Fugu 想要进入的不是测验，而是工作流。**

六、但这类结果也有明显边界：它强在组织，不等于强在通用理解

首先要承认，这是一份由作者自己发布的技术报告，不是独立第三方复现论文。报告中大量对比基线是 provider-reported，也就是供应商或平台自己报出的成绩。对于这种前沿系统，基准数字能说明方向，但不自动等于可重复的公共真相。尤其当评测 harness、工具权限、最大步数、超时策略都不完全一样时，分数的可比性会被稀释。**越靠近 frontier，数字越需要被当成“条件下的结果”，而不是“永远成立的排名”。**

第二个边界在于，Fugu 的胜利依赖一个前提：worker 池里本来就有相当强、而且彼此互补的模型。如果所有 worker 都在同一个盲点上失误，编排器再聪明也救不了。它能放大差异，能减少浪费，能把能力匹配得更精细，但它并不能凭空创造不存在的知识。这个限制很重要，因为它说明 Fugu 不是“从无到有造智能”，而是“把已有智能的缝隙缝得更紧”。**调度能放大能力，但放大不了空白。**

第三个边界是成本。Fugu-Ultra 显然更贵、更慢，也更复杂。多轮 workflow 让系统更强，但也让失败模式更多：路由可能错，专家可能互相污染，某个步骤的错误可能被后续步骤放大，甚至因为流程太长而引入新的脆弱性。Fugu 事实上也在和这种脆弱性作斗争，所以它才花很多篇幅讲 memory、isolation、topology 和 orchestration collapse。**多模型不是免费的超级兵器，它只是把复杂性从参数里搬到了系统里。**

还有一个经常被忽略的点：很多演示任务对“能否把工作流跑通”很友好，却未必能完整代表真实世界。比如盲fold chess、AutoResearch、kana 阅读顺序、CAD 机械光圈，确实都很有启发性，但它们仍然是作者精心设计的场景。这样的结果说明系统有

潜力，却不等于它在所有开放世界任务中都能稳定复制。换句话说，**Fugu 展示的是编排思维的上限，不是社会化 AI 的最终答案。**

七、它改变了什么直觉：智能也许不是一个点，而是一张图

Fugu 最值得重视的地方，也许不在于某个榜单位置，而在于它悄悄改变了我们衡量“更强模型”的方法。过去我们常常默认：更强，就是更多参数、更大数据、更长训练。Fugu 给出的另一个答案是：更强，也可能意味着更好的分工、更好的路由、更好的拓扑、更好的记忆策略。**当模型之间的专长开始分化，能力的瓶颈就不只在“谁最聪明”，而在“谁和谁配合得最好”。**

这会带来一个很有哲学意味的变化。我们通常把模型看作一个黑箱，输入问题、输出答案；但 Fugu 把黑箱拆成了两层：一层负责“思考”，一层负责“组织思考”。前者是内容能力，后者是结构能力。前者决定你能不能答出来，后者决定你能不能在有限成本下答得更稳、更快、更适配不同场景。**如果说传统模型在争“脑力”，Fugu 争的是“智力的交通系统”。**

这也解释了为什么报告会强调 privacy、compliance 和模型可替换性。参数级合并要求你拿到模型权重，行为级编排则只需要 API 可用。前者适合开源权重世界，后者适合现实中的闭源 API 世界。于是“模型融合”不再只是权重平均，也可以是行为层面的功能合成。这个转换很深，因为它把 AI 生态从“谁有最大训练预算”部分转向“谁能最会整合现成资源”。**如果这条路成立，AI 进步的一部分将来自制度化的协作，而不是单点式的巨型训练。**

当然，这种叙事也不能走过头。把 orchestration 说成新 scaling axis，并不自动意味着它会一直压过更大的模型训练。很可能真实世界里两条路都会继续前进：更大的模型提供更强的基础能力，更好的编排把这些能力更高效地用起来。Fugu 的价值，也许就在于它把原来被忽视的一层显影出来，让我们知道“系统级智能”不是附属品，而是一个独立的研究对象。**它不是宣布模型时代结束，而是在说：模型时代里，组织能力也该被当成一等公民。**

八、如果把 Fugu 的故事压缩成一句话

最简单的一句话是：**Fugu 不是让一个模型学会所有事，而是让一个系统学会在正确的时刻调用正确的模型。**这句话之所以重要，是因为它把“性能”从单点能力改写成了时序和结构问题。一个会选专家的系统，未必每一步都比最强专家更强，但它可能在长期、复杂、跨领域任务中，比任何单独模型更稳定、更适配、更可控。

这也是我读完整份报告后最强的直觉：Fugu 的真正实验对象，不是某个 benchmark，而是“协作是否可以被规模化”。如果答案是肯定的，那么下一步就不是继续问“哪个模型最强”，而是问“怎样的协作图最强”。如果答案是否定的，那 Fugu 仍然已经完成了一件重要的事：它把这道题从想象变成了可测量、可优化、可争论的工程问题。**无论结论最终偏向哪一边，真正被改变的，是我们理解智能的坐标系。**

还有一个更形象的比喻可以帮助记住它：Fugu 像空中交通管制，而不是一架更大的飞机。飞机比喻强调单体性能，塔台比喻强调协调秩序。现实中的复杂任务越来越像一片拥挤空域，单架飞机再强，也不如一套能让不同机型、不同航线、不同天气条件都安全起降的调度系统。Fugu 想做的，正是把这种“安全起降”的逻辑搬进语言模型世界。**当任务足够复杂时，最重要的未必是把一台机器造到极限，而是让整个系统学会不乱。**

参考资料

1. Sakana Fugu Technical Report PDF
2. Sakana Fugu 官方页面
3. Trinity: An Evolved LLM Coordinator
4. Learning to Orchestrate Agents in Natural Language with the Conductor
5. SWE-Bench Pro
6. Terminal-Bench
7. LiveCodeBench
8. GPQA

9. Humanity’s Last Exam
10. CharXiv: Charting Gaps in Realistic Chart Understanding in Multimodal LLMs