UAVBench 的决策模型与边缘部署的可行性

MatthewMichelson GoranMuric AdrienBibal Steven Minto n

InferLink Corporation，Marina del Rey，CA

arXiv:2611.28331v1 [cs.AI] 20 Nov 2026

摘要

小型语言模型常被用于无人机上的推理。然而，良好的基准测试得分并不能说明模型是否能够在边缘计算机上实际运行并发挥作用。本文使用 UAV决策基准测试 UAVBench，研究三个相关问题。第一，小型结构化决策模型是否适合执行这些任务？第二，在将候选模型部署到物理设备上测试之前，普通本地机器（Mac）能否对其进行筛选？第三，哪些软件和资源限制决定了部署可行性？我们的主要 Mac 评估涵盖了 24 种模型配置，使用的是 UAVBench 中相同的 2,000 道经过验证的选择题。这些配置包括 11 个生成式模型的 20 种规模与精度配置，以及四个未接受额外针对UAVBench的训练的开放权重结构化决策模型。在我们观察到的结果中，一个结构化决策模型 H2O Lightning 4B 取得了最高准确率，在 Mac 上达到 77.65%。这表明，这些模型非常适合此类任务。然而，当我们尝试部署这些决策模型时，在所测试的边缘设备上遇到了转换和运行时限制，本文将对此进行详细说明。为测试本地筛选效果，我们在 Mac 和 Qualcomm RB3开发板上分别对两个生成式模型进行了匹配评估。两者的准确率并不完全一致，但排名一致。LFM2 2.6B 在两个平台上的排名都高于 Qwen2.5 1.5B。这支持使用本地机器对候选模型进行初步筛选。尽管如此，不同平台之间的具体答案仍存在差异，因此在实际硬件上进行测试仍然是必要的。

关键词：UAVBench；边缘计算；结构化决策模型；小型语言模型；模型部署。

1 引言

无人机可能需要根据对当前状态的描述，在少数几种任务动作中作出选择。例如，它可能需要决定是继续执行任务、返回基地，还是请求协助。需要明确的是，在这种应用场景中，决策模型运行在与飞行控制器不同的计算机上。它不负责稳定控制或加速度控制等低级活动，而是决定执行哪些高级命令。实际问题是：哪些小型模型能够足够准确、足够迅速地作出这些选择，同时还能运行在足够小巧、可由无人机携带的计算机上。

作为一个具体示例，请看图1中的 UAVBench 问题，该问题取自基准测试的已验证发布版本 [1,2]。一架固定翼无人机正在机场附近的搜索救援网格中飞行，侧风会随高度增加而增强，同时卫星定位（全球导航卫星系统）受到干扰，电池储备为 30%。问题询问：在七个动作中，哪一个既能保持搜索效率，又能确保安全。有些选项明显违反规则。例如，爬升至 180 米超过了 150 米的上限高度，而下降至 25 米则低于 30 米的下限高度。另一些选项看似合理，实际却不正确，例如立即返航。正确答案是 D：下降至 60 米并调整网格间距。我们将以这个问题作为贯穿全文的示例。

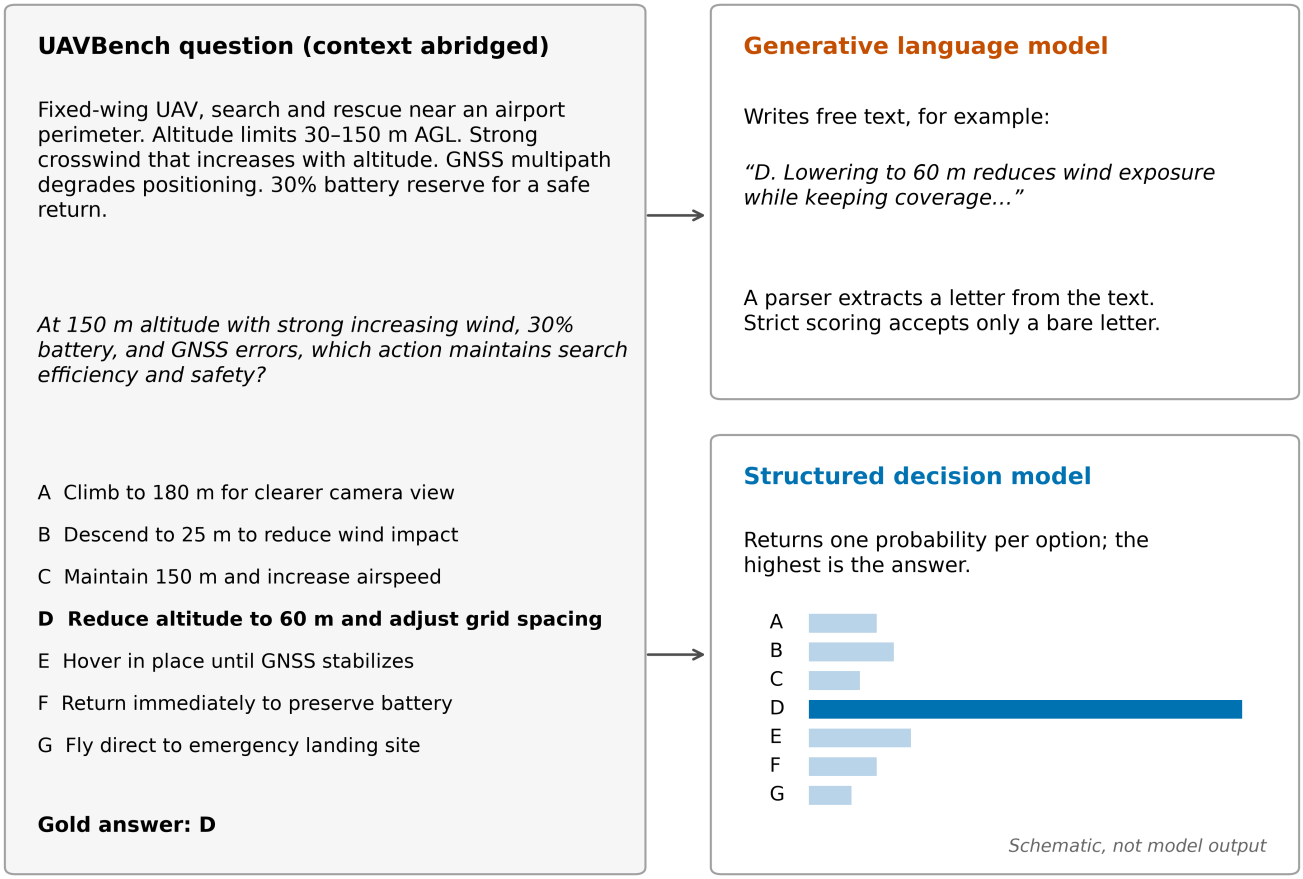


图1： 一个 UAVBench 问题（样式 10，混合综合推理）以及模型回答该问题的两种方式。生成式模型会撰写文本，解析器则从中提取一个字母。结构化决策模型会为每个选项返回一个概率。生成的文本和柱形高度仅为示意，并非经过测试的模型输出。

图1还展示了模型作答的两种方式。生成式语言模型会生成文本，随后解析器必须从中找出所选字母。结构化决策模型（有时也称为类型化决策模型）则省去了这一步。给定一个状态、一个问题和列出的选项，它会为每个选项返回一个概率，并以概率最高的选项作为答案。TypeSafe AI 通过其 Jev 模型在商业上提供这种接口，并将其称为 System One 模型 [3]。近来，开放权重发布版本让人们可以轻松地在本地机器上测试这类模型 [4–7]。像 UAVBench 这样的有限选项基准测试很适合用于测试这类模型，因为所需输出本来就是从多个选项中做出选择。

本文探讨三个问题。第一，小型结构化决策模型是否适用于无人机任务决策？第二，在真实边缘硬件上进行测试既缓慢又昂贵，那么能否先用普通台式机（本例中为 Mac）对候选模型进行排序？第三，尝试将最佳候选模型部署到实体机器人开发板时，会遇到哪些障碍？为回答这些问题，我们在 Mac 上评估了 24 种模型配置，在 Qualcomm RB3 Gen 2 开发板上重复进行匹配实验，以检验模型准确率的排序是否保持不变，并进一步尝试将每个决策模型移植到 RB3。

简言之，模型选择与模型部署是相互关联的问题，而本文的主要贡献正是将二者联系起来。总而言之，本文有三项贡献：

我们证明，四种开放权重结构化决策模型无需任何任务特定训练，就能在 UAVBench 上表现出色。四种模型中有三种的得分高于我们测试的全部 20 种生成式配置，而表现最佳的模型得分为 77.65%。

我们表明，在边缘硬件上进行测试前，本地机器可以提供有用的初步排名。不过，

配对实验还表明，相近的整体准确率可能掩盖单个答案中可复现的变化，因此仍需进行硬件验证。

我们描述了将这些模型部署到边缘硬件上时所面临的实际障碍，而这些障碍并不只是模型能否装入内存。其中包括不支持的算子、转换过程中产生的数值漂移，以及对单次内存分配的限制。

本文其余部分的结构如下。第2节介绍相关工作，第3节描述我们的实验。第4、5和6节依次回答三个问题。第7节讨论这些结果的含义，第8节汇总局限性和后续实验。

2 背景与相关工作

2.1 用于无人机决策的语言模型

Ferrag、Lakas和Debbah推出了UAVBench，用于研究语言模型对无人机场景进行推理的能力 [1]。他们围绕十种推理风格生成了50,000道选择题，并发布了其中2,000道题组成的经验证的子集，这也是我们所使用的数据集（第3.1节）。随后，他们在该子集上评估了近40个模型，涵盖Llama 3.2 3B等小型开放模型，以及GPT-5 Chat等大型托管模型。论文分别报告了每种推理风格下的准确率，而不是单一的总体得分。不过，作者还发布了逐题结果 [2], ，这些结果与论文中的各风格准确率完全一致。因此，我们根据这些结果计算了每个模型的总体得分。
他们评估中表现最佳的模型是 Qwen3-235B-A22B，总体得分为 83.55%。其他大型托管模型紧随其后，例如 ChatGPT-4o（80.35%）、GPT-5 Chat（80.15%）和 Gemini 2.5 Flash（76.75%）。小型模型的得分较低。例如，Gemma 3n E4B 得分为 73.25%，LFM2 2.6B 得分为 69.75%，Llama 3.2 3B 得分为 62.00%。在分析中，作者观察到，较小或轻量级模型的得分往往集中在 70% 左右，并据此得出结论：复杂的无人机推理取决于模型规模和专业化程度 [1]。

我们的工作从三个方面在他们的研究基础上进行了拓展。首先，我们评估了一类不同的模型，即他们未曾测试过的结构化决策模型。正如我们在第4节中所展示的，这些模型对“无人机在 UAVBench 上的准确率仅仅随模型大小变化”这一观点提出了挑战。其次，作者通过托管服务评估模型，并关注每个模型答案的质量。而我们自行运行了每个模型，测量其速度和内存使用情况，并采用更严格的规则对其答案进行评分（第3.1节）。最后，我们将最有前景的模型部署到真实的边缘硬件上，并报告了它们在这些硬件上的表现，以及阻碍其运行的因素。
更广泛而言，语言模型正被用于探索无人机自主性。例如，Vemprala 等人使用 ChatGPT 将高级指令转换为机器人代码，包括空中导航任务 [8]。类似地，TypeFly 使用语言模型编写简短程序，根据用户的任务描述操控无人机飞行 [9]。Javaid、Saeed 和 He 对这一不断发展的领域进行了综述 [10]。

与我们的研究设定更接近的是，SayCan 将语言模型与机器人的可用技能相结合 [11]。它会对每项技能进行评分，将其作为候选的下一步，并综合考虑语言模型对该技能是否有帮助的判断，以及用于判断该技能能否成功的价值函数，而不是生成自由文本。换句话说，它将决策视为在一组固定选项中进行选择，这与 UAVBench 问题的形式相同。我们在此研究方向的基础上，进一步开展了受控局部测量、推理系统间的迁移，并纳入了近期的开放权重决策模型。

2.2 结构化决策模型

结构化决策模型是一个新兴的研究课题，这也不是首个对此类模型开展的研究。Deußer、Sparrenberg 和 Sifa 在 37 个公开数据集上对托管的 Jev模型进行零样本评估，使用其选择题、是非题和评分细则评分题型 [12]。Qu、Chen 和 Labi 将 Jev 与语言世界模型结合，用于模拟驾驶和重放交通场景 [13]。Delaunay 研究了开放权重决策模型的置信度、对扰动的敏感性及其解释 [14]。我们的新增工作是在 UAVBench 上开展评估，同时进行本地筛选、在 RB3 上进行物理测量，并记录我们遇到的部署障碍。

在本文中，结构化决策模型是指一种针对声明的答案返回概率分布的系统。该术语描述的是接口，而非特定架构。它并不意味着 GLiDE、Liquid d1 和 H2O 复现了 Jev 未公开的实现或训练过程，我们也不评估商业版 Jev模型本身。这些公开发布版本之间也存在相当大的差异。GLiDE 提供了一个小型编码器模型（0.4B）以及一个混合模型，后者混合了线性注意力层和全注意力层（0.8B） [4,5]。Liquid d1 提供了其自身的模型、决策计算和校准[6]。H2O 提供了一个模型，以及已发布的读出组件和服务实现 [7]。我们保留所有这些特定于模型的组件。因此，尽管我们通过同一个七选一任务调用每个模型，但它们的提示词、精度和内部计算并不完全相同。同样，我们使用“开放权重”而非“开源”这一说法，因为并非每个发布版本都拥有不受限制的开源许可证。

2.3 使用语言模型在选项中进行选择

生成式模型也可以用于在答案选项中进行选择，而执行这一选择的机制会影响最终结果。Robinson 和 Wingate 发现，将问题和选项一起呈现，并要求模型回答选项字母，比单独为每个答案评分效果更好，尤其是对于能够很好地将选项与其字母对应起来的模型而言 [15]。Zheng 等人发现，模型往往倾向于选择特定的选项 ID（例如“选项 A”）作为答案，他们将这种现象称为选择偏差 [16]。这对 UAVBench 很重要，因为出现频率最高的标准答案字母约占全部问题的四分之一。

最后，只有经过校准，概率才对决策有用。直观地说，被赋予 0.7 概率的选项，应该有大约 70% 的概率是正确的。现代神经网络的校准效果通常较差 [17], ，因此不应盲目相信决策模型返回的概率。

2.4 在边缘硬件上运行语言模型

在手机、机器人及其他边缘设备上运行语言模型是一个活跃的研究领域，Xu 等人对此进行了综述 [18]。大多数部署都依赖权重量化来满足内存预算。例如，GPTQ 会将权重压缩至 3 或 4 位，而准确率几乎不会损失 [19], ； AWQ 则针对相同的低位宽、仅权重量化场景，适用于设备端使用 [20]。在实践中，我们使用的量化格式由 llama.cpp [21] 和 Apple 的 MLX [22] 等本地运行时提供。

然而，量化模型只是问题的一半。目标设备上的运行时还必须支持模型中的每个算子，并产生相同的数值。Yuan 等人指出，GPU 类型、GPU 数量或批大小的变化都可能改变模型的输出，因为微小的舍入差异会逐渐累积 [23]。当同一个模型文件分别在 Mac 和 RB3 上运行时，我们也观察到了相同的现象。

3 实验设置

我们的实验遵循第 1 节提出的三个问题，并分两个阶段进行。第一阶段，我们在 Mac 上使用完整的 2,000 个 UAVBench 问题集，对所有模型进行了评估。其中包括 20 种生成式模型配置，以及四种结构化决策模型，并回答了我们的

第一个问题。第二阶段，我们将一组规模较小的模型迁移到 RB3开发板上。对于其中两种生成式模型，我们在两个平台上都运行了完整基准测试，以检查 Mac 的结果是否能在开发板上复现（即我们的第二个问题）。随后，我们尝试将全部四种决策模型迁移到开发板上（即我们的第三个问题）。本节其余部分将介绍任务及其评分方式（第 3.1 节）、我们的硬件（第 3.2 节）、模型（第 3.3 节）、将模型迁移到 RB3开发板的方法（第 3.4 节）以及统计检验（第3.5节）。最后，第 3.6 节用一个表格总结了所有实验。

3.1 任务与评分

我们使用了经过验证的 UAVBench 版本，其中包含 2,000 道选择题。我们将其固定在源代码修订版 4f2baa78a50eea3bb63dffb83565b8f9fb7a7a77 [2], ，以确保每个模型看到的题目完全相同。该版本针对基准测试的十种推理风格各包含 200 道题目，并且每道题都有七个答案选项（A 至 G）。需要注意的是，作者还发布了一个规模大得多的 50,000 道题目集。我们选择经过验证的 2,000 道题目子集有两个原因。首先，作者自己的所有模型结果都使用了这一子集，因此我们的数据可以直接与其结果进行比较（第 2.1 节）。其次，这使实验更具可行性。按照开发板每道题大约需要 4 至 17 秒的速度，完整的 50,000 道题目集对每个模型而言都需要数天时间。在运行任何模型之前，我们对该版本进行了审查，未发现重复题目。我们还确认，标准标签和全部 14,000 个答案选项都与固定的源代码相匹配。

在推理过程中，模型只能看到场景、问题和答案选项。标准答案会单独保存，仅在之后用于评分。我们也没有针对 UAVBench 做任何适配。例如，我们没有进行训练，没有基于准确率进行提示调优，没有进行分类器拟合，也没有重新校准概率。尽管如此，UAVBench 是公开的，因此我们无法排除其中一些内容曾出现在模型的原始训练数据中。因此，我们所说的“没有任务特定训练”，是指我们没有自行进行任何训练。

我们的准确率始终以所有预定问题为分母，确保不会悄悄遗漏任何问题。如果问题长度超过模型的令牌预算，我们就将其计为错误，而不会缩短或删除该问题。运行失败和无效输出也同样处理。为了便于理解这些数据，若模型均匀随机地选择一个选项，其得分为 14.29%。有趣的是，我们发现最常见的标准答案字母是 C，因此，即使完全不进行推理，始终回答 C 的模型也能得到 25.30%，这构成了朴素基准。

生成式模型和决策模型会产生不同类型的输出，因此我们采用不同的评分方式。我们要求生成式模型只用一个字母作答，严格评分仅接受完全符合这一格式的答案。这意味着严格评分衡量的不仅是模型是否选对答案，也包括它是否遵循格式要求。模型也可能在较长的句子中给出正确答案（例如：“最佳选项是 D，因为……”）。对于这类答案，我们还报告了基准测试自带的宽松提取器所计算的分数，该提取器会在文本中查找字母。对决策模型评分则更简单：它会为每个选项返回一个概率，我们只需将概率最高的选项作为答案（如果两个选项的概率完全相同，则选择字母靠前的选项）。我们根据返回的概率自行计算分数，而没有依赖某些推理服务 API 为方便起见返回的“choice”字段。

3.2 硬件

我们在配备 32 吉比字节统一内存的 Apple M2 Pro Mac 上进行了本地实验。对于边缘平台，我们使用了 Qualcomm RB3 Gen 2开发板。该开发板搭载 QCS6490 处理器和 Adreno 643 GPU，其运行时可用的随机存取存储器约为 5.25 吉比字节。我们通过 Qualcomm Device Cloud 的免费层远程访问该开发板。远程连接仅用于发送

命令和收集结果，所有推理运行均在开发板本身上完成。因此，我们从报告的计时中排除了网络耗时。

需要明确的是，我们将 Mac 用作开发机器，而不是 RB3 的模拟器。我们确实对 Mac 设置了内存限制，以便让实验保持可控，并在一定程度上反映实际情况。不过，这些限制无法重现实体开发板的内存带宽、加速器、功耗预算或热设计。因此，Mac 可以帮助我们判断哪些模型看起来有前景，却无法告诉我们它们在开发板上的实际表现。这正是我们在第5节中描述的差距。

3.3 模型

我们测试了两组模型。第一组是一系列生成式模型，作为基线；第二组则是四个结构化决策模型。

生成式模型。这组模型涵盖了来自六个系列的 11 个检查点，分别是通义千问 Qwen2.5（参数量为 0.5B、1.5B、3B 和 7B）、SmolLM2（135M、360M 和 1.7B）、LFM2 2.6B、Granite 4 H Micro、Llama 3.2 3B，以及仅含文本的 Gemma 3n E4B 版本。每个检查点都以两种精度运行：MLX FP16 和 MLX 4 位，只要通过资源检查即可，因此我们获得了 20 种可用配置。所有生成式模型的运行均采用相同设置：每次处理一个请求、使用贪心解码、每个请求使用全新的缓存、输入上限为 2,048 个 token，输出上限为 16 个 token（足以生成一个字母的答案）。最后，我们首次对 Gemma 进行的 4 位转换产生了 NaN逻辑值。我们将其排除，改用 BF16 权重进行修正后的转换（我们会在第5.1节讨论这一失败）。

结构化决策模型。我们测试了四种决策模型的固定版本，分别是 GLiDE 0.4B、GLiDE 0.8B [4,5], Liquid d1 2B [6] 和 H2O Lightning 4B [7], 。为便于说明，后文分别简称为 Liquid d1 和 H2O。GLiDE 模型和 Liquid d1 通过 Apple MPS 以 BF16 运行。H2O 则以 8 位 MLX 转换版本运行，该版本链接自 H2O 自己的模型卡 [24]。我们通过 H2O 的官方适配层提供服务，采用推荐的双向注意力和默认配置档。所有情况下，我们都保留了各模型发布时的提示词、标签头和校准设置，因为这些都是发布方实际交付内容的一部分。

在正式进行完整测试之前，我们为每个模型进行了简短的试运行，测试十个问题（每种推理风格各一个）；这些问题是在未查看任何模型输出的情况下选定的。试运行只检查模型能否运行，以及对同一问题询问两次时是否给出相同答案，不要求达到任何最低准确率。四个模型均通过了试运行。随后，每个模型都对全部 2,000 个问题进行了测试，输入限制为 4,096 个 token，且不进行截断。参考文献列出了我们所用每个模型的确切修订版本。表 1 总结了本文通篇使用的精度标签。

表 1： 本文使用的精度和文件格式标签。

标签	含义	使用场景
MLX FP16	16 位浮点权重，使用 Apple 运行 MLX	Mac 生成式模型
MLX 4 位	MLX 仿射 4 位量化，由……转换而来 FP16 权重（Gemma 使用 BF16）	Mac 生成式模型；MLX 资源重复
BF16	16 位 bfloat 权重，通过 Apple MPS 运行	Mac 决策研究（GLiDE、Liquid d1）
MLX q8	H2O Lightning 4B 的 MLX 8 位转换	Mac 决策研究（H2O）
Q4_0 GGUF	4 位 GGUF 文件，两个平台上完全相同	匹配的 Mac–RB3 传输（通义千问2.5 1.5B, LFM2）
Q4_K_M GGUF	4 位 k-量化 GGUF 文件，两个平台上完全相同	H2O Mac–RB3 诊断

3.4 从 Mac 转向 RB3

我们的第二个和第三个问题都需要在 RB3 上运行模型，而不只是 Mac。因此，我们设计了四组实验，尽可能直接地比较这两个平台。

匹配迁移。对于 Qwen2.5 1.5B 和 LFM2 2.6B，我们在两个平台上运行了相同的官方 Q4_0 GGUF 文件，并使用匹配的提示词和生成设置。在 Mac 上，推理使用 Metal；在 RB3 上，则使用带 GPU OpenCL 的 GenieX。我们确认两个平台上的提示哈希一致，因此二者接收到的输入完全相同。需要注意的是，两者的运行时和内核构建仍然不同。因此，这项实验比较的是两套完整的部署系统，而不是单独隔离硬件本身的影响。

重复计时。单次计时可能存在噪声，因此我们还重复进行了计时测量。我们选取了一个包含100个问题的共享题集，并在每个平台上针对每个模型运行六轮。每轮都会启动一个全新的进程，并进行各自的预热；同时，我们交替安排哪个模型先运行。（需要说明的是，在 RB3 上，我们还会在每轮开始前等待 GPU 冷却至 50 C 或更低，作为“冷却检查”。）随后，我们取每轮中的位于中间位置的时间，并报告这六轮中位数的中位数。计时范围包括分词和生成，但不包括模型加载、预热或网络耗时。我们还全程记录了 GPU 温度，因为高温可能会降低开发板的速度。

数值诊断。我们还想了解为什么某些答案在不同平台上会发生变化。因此，对于每个模型，我们选取了 40 个问题，其中 20 个问题的答案在两个平台上相同，另外 20 个不同，并对每个问题分别运行了三次。除此之外，我们还向两个平台输入了完全相同的已保存 token ID，并直接比较答案选项对应的 logits。这样一来，我们观察到的任何差异都不可能是由分词或提示词处理造成的。

RB3 上的决策模型。我们尝试将全部四个决策模型迁移到 RB3，但只有其中一个——H2O——成功在开发板上回答问题。另外三个模型分别在迁移过程的不同阶段失败，我们将在第 6 节逐一说明这些失败。每次尝试时，我们都会先将模型转换为开发板运行时支持的格式。只有在转换忠实保留原始模型的情况下，我们才会认可该转换。也就是说，转换必须保留模型的注意力、分词、池化、标签头和校准，并且必须能在一组固定的测试输入上复现原始模型的概率。如果转换未通过这些检查，我们会将其报告为转换失败，而不是准确率低，因为转换出错无法说明模型本身的决策能力如何。对于 H2O，我们在 Mac 和 RB3 上运行了同一个 GGUF Q4_K_M 文件 [25]，该文件由同一运行时源代码构建。不过，就连 H2O 的首次概率检查也未能通过，因此我们在固定的 100个问题测试集上对其进行了测试，而非完整基准测试（见第 5.5 节）。

3.5 统计分析

我们报告的大多数差异都是描述性的。不过，对于支撑主要结论的比较，我们进行了统计检验，并根据数据的配对方式选择相应的检验方法。

对于 LFM2 2.6B 与通义千问 Qwen2.5 7B 的比较，两个模型回答了相同的 2,000 个问题，因此我们采用了配对自助法。具体来说，我们对这些问题进行 2,000 次重采样，每次从两个模型中抽取相同数量的问题，并重新计算准确率差异。由此，我们得到了该差异的区间。

对于每个决策模型与最佳生成式配置的比较，我们采用了更简单、更谨慎的方法。我们计算了差异的 95%区间，并将两次运行视为相互独立。实际上，两个模型往往会答对相同的问题，而配对分析会利用这一点来得到更窄的区间。因此，忽略配对关系后，我们得到的区间比实际所需的更宽，这种做法更为谨慎。

对于 Mac 与 RB3 的比较，两个平台回答了相同的问题，因此我们使用了精确麦克内马尔检验 [26,27]。直观来说，如果两个平台都答对（或都答错）某个问题，这并不能告诉我们

哪个平台更好。因此，该检验只考察恰好只有一个平台答对的问题，并判断这些情况是否均匀分布在两个平台之间。
需要特别指出的是，这些测试都没有考虑到我们是在众多候选模型中进行选择这一事实，也没有考虑 UAVBench 与模型训练数据之间可能存在重叠。

3.6 实验总结

表 2 总结了我们的五项主要实验。请注意，这些实验并未涵盖完全相同的模型或问题。两项 Mac 研究覆盖了全部 2,000 个问题。在 RB3 上，通义千问Qwen2.5 1.5B 和 LFM2 2.6B 也完成了全量测试，并进行了重复计时测试以测量其延迟。H2O 进行的是固定的 100 题诊断测试，而非全量测试，因为它首次概率检查未通过，且开发板运行较慢（第 3.4 节）。最后，GLiDE 和 Liquid d1 为部署提供了相关证据，但无法在 RB3 上测得准确率，因为我们无法让它们在该开发板上运行（第 6 节）。我们还在 Mac 上进行了规模较小的速度和内存重复研究，并将在第 4.3 节中进行介绍。

表 2： 五项主要研究、其涵盖范围及主要控制变量。

研究	模型与涵盖范围	主要控制变量
Mac 生成式模型-els	11 个检查点；20 个可用条件；每种各 2,000 个问题	MLX FP16 或 MLX 4 位；2,048 个 token 的输入限制；分配器额度为 8 吉比字节
Mac 决策模型	GLiDE 0.4B 和 0.8B、Liquid d1 2B，H2O Lightning 4B；每种各 2,000 个	原生决策代码；BF16 或 MLX q8；4,096-令牌输入限制
匹配的完整传输	通义千问 1.5B 和 LFM2 2.6B；每个模型和平台	完全相同的 Q4_0 GGUF 文件和提示词；Metal 与 OpenCL 对比
重复传输计时	两个模型；每轮 100个问题，共六轮每个平台上	使用全新进程；单独预热；交替进行模型顺序
H2O 传输诊断-nostic	Mac 上相同的固定100个问题以及 RB3	完全相同的 Q4_K_M GGUF 文件和运行时源代码；1,024个令牌上下文

4 小型决策模型作为候选方案

4.1 决策模型在未经任务特定训练的情况下得分最高

表3给出了完整的 Mac 结果，图 2 绘制了准确率与速度的关系。最值得关注的是，决策模型是很有竞争力的候选者。即使是最小的 GLiDE 0.4B，得分也达到 66.30%，与通义千问 Qwen2.5 7B 采用 MLX 4 位时的 66.15% 相近，而后者是大得多的生成式模型。另外三个决策模型的得分则都超过了表现最佳的生成式配置——采用 MLX 4 位的 Gemma 3n E4B，其得分为 70.20%。

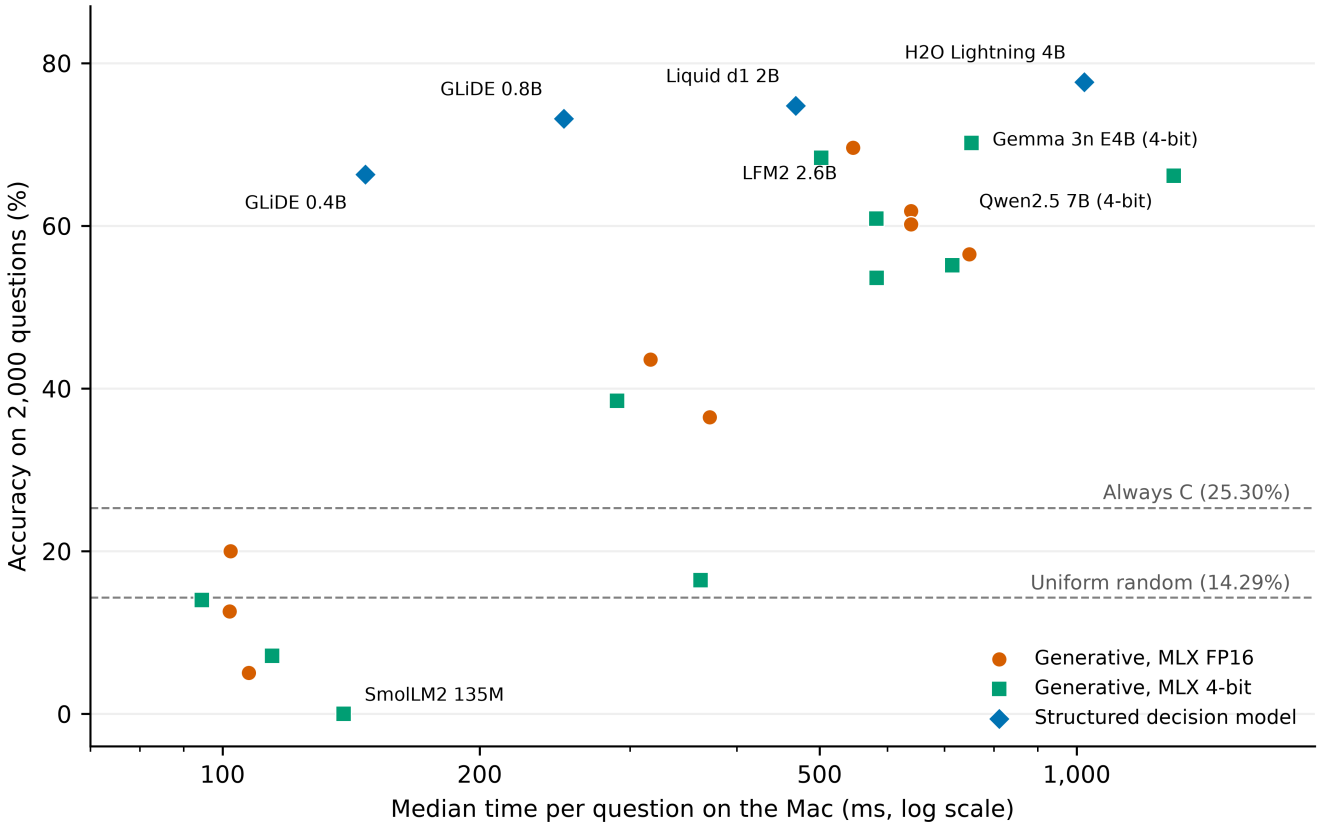


图 2：表3中全部 24 种配置在 Mac 上的准确率与每题中位数用时。决策模型的用时包括提示构造和分词，因此与生成式模型的用时相比，若有偏差也只会高估。虚线表示均匀随机选择和始终回答 C。

具体来说，H2O 比 Gemma 高出 7.45 个百分点，这意味着它在 2,000 道题中多答对了 149 道。Liquid d1 高出 4.55 个百分点（多答对 91 道），GLiDE 0.8B 高出 2.95 个百分点（多答对 59 道）。当然，差距的一部分可能只是运气使然，取决于基准测试中恰好出现了哪些题目。因此，我们还针对每个差距计算了一个 95%区间，给出与数据相符的差异范围（第3.5节）。直观来说，如果这个范围包含零，我们就无法排除两个模型表现相当的可能性。H2O 的范围为 4.74 到 10.16 个百分点，Liquid d1 的范围为 1.79 到 7.31 个百分点，因此两者都明显优于 Gemma。GLiDE 0.8B 的范围为 0.16 到 5.74 个百分点，因此也领先，但优势很小，因为其下限仅略高于零。最后请注意，我们构建这些区间时采取了谨慎的做法。配对分析很可能会得出更窄的范围，这只会让决策模型的领先优势更加明显。

表3： Mac 上全部 24 种配置的完整 UAVBench 结果。每种配置均根据全部 2,000 道问题进行评分。

模型	精度	正确数 / 2,000	准确率（%）	宽松评分（%）	中位数时间（毫秒）	峰值分配（吉比字节）
SmolLM2 135M	MLX FP16	101	5.05	13.50	107.3	0.625
SmolLM2 135M	MLX 4 位	0	0.00	12.50	138.6	0.343
SmolLM2 360M	MLX FP16	252	12.60	14.55	101.9	1.285
SmolLM2 360M	MLX 4 位	143	7.15	12.50	114.3	0.638
SmolLM2 1.7B	MLX FP16	729	36.45	36.85	371.7	3.653
SmolLM2 1.7B	MLX 4 位	329	16.45	16.55	362.8	1.624
通义千问2.5 0.5B	MLX FP16	400	20.00	20.00	102.2	1.439
Qwen2.5 0.5B	MLX 4 位	280	14.00	14.00	94.5	0.673
Qwen2.5 1.5B	MLX FP16	871	43.55	43.55	316.8	3.235
Qwen2.5 1.5B	MLX 4 位	770	38.50	38.50	289.7	1.546
Qwen2.5 3B	MLX FP16	1,236	61.80	61.80	639.5	6.055
Qwen2.5 3B	MLX 4 位	1,072	53.60	53.60	583.3	2.328
Qwen2.5 7B	MLX 4 位	1,323	66.15	66.15	1,298.3	4.568
LFM2 2.6B	MLX FP16	1,392	69.60	69.60	547.1	5.090
LFM2 2.6B	MLX 4 位	1,367	68.35	68.35	502.1	2.037
Granite 4 H Micro	MLX FP16	1,130	56.50	56.50	748.3	7.888
Granite 4 H Micro	MLX 4 位	1,103	55.15	55.15	714.9	3.915
Llama 3.2 3B	MLX FP16	1,204	60.20	60.20	639.6	6.313
Llama 3.2 3B	MLX 4 位	1,218	60.90	60.90	582.6	2.330
Gemma 3nE4B 文本	MLX 4 位	1,404	70.20	70.20	752.9	4.733
GLiDE 0.4B	BF16	1,326	66.30	–	147.0	–
GLiDE 0.8B	BF16	1,463	73.15	–	250.9	–
Liquid d1 2B	BF16	1,495	74.75	–	468.7	–
H2O Lightning 4B	MLX q8	1,553	77.65	–	1,020.6	–

评分。前 20 行是生成式模型，按照严格单字母规则评分。宽松列显示基准测试提取器的得分。最后四行是决策模型，按概率最高的选项评分。溢出和失败均计为错误，数值无效的 Gemma 转换不计入。

精度。标签遵循表 1。Gemma 4 位行由 BF16 权重转换而来，其他 4 位行则由 FP16 转换而来。峰值分配指 MLX 峰值，而非总 RAM，且仅报告生成式模型的峰值分配。

计时。时间为已执行请求的单次运行中位数，不包括加载和预热。对于生成式模型，计时涵盖预填充和生成；对于决策模型，还涵盖提示构造、分词和概率读取组件，H2O 还包括一次本地 HTTP 调用。由于计时范围有所不同，这两组模型的速度并不能严格比较。

这些结果也让决策模型有了更广阔的参照背景。回顾第 2.1 节，UAVBench 作者发现，较小的模型得分往往集中在 70% 左右；他们测试的最佳大型模型 Qwen3-235B-A22B 得分为 83.55%。在 Mac 上运行的 4B 模型 H2O 得分为 77.65%，与他们评估中的最佳模型相差约 6 个百分点，且略高于 Gemini 2.5 Flash（76.75%）等大型托管模型。他们的数据采用了该基准测试中较宽松的提取器，并使用各模型的托管版本，因此这种比较只能作为参考，并非完全精确。即便如此，这仍表明，在这项基准测试中，配备合适接口的小模型也能接近大得多的模型。

尽管结构化模型产生了有趣的结果，但我们确实希望提醒读者注意该比较所展示的内容。首先，我们比较的是完整的软件包，而不是仅单独考察模型，而且除决策模型设计本身之外，两组之间的许多方面

也可能存在差异。例如，GLiDE 0.8B 以完整的 BF16 精度运行，而 Gemma 则以 4 位精度运行；此外，两者使用了不同的提示词和不同的推理软件。其中任何一项差异都可能造成部分性能差距。我们的结果可以帮助从业者针对其任务，在各种已发布的模型中进行选择；不过，这并未明确表明决策模型设计本身更优。我们将在第 7.1 节介绍一项能够区分这些影响的实验。

4.2 决策模型在准确率和速度之间提供了不同选择

决策模型同样在准确率和速度之间提供了一系列选择。H2O 的准确率最高，达到 77.65%，其中位调用时间为 1.021s；GLiDE 0.8B 的准确率为 73.15%，耗时 0.251s，速度约快四倍。Liquid d1 介于两者之间，准确率为 74.75%，耗时 0.469s。最后，GLiDE 0.4B 是这组模型中速度最快的，耗时 0.147s，但得分最低，为 66.30%。不过需要指出，这些结果来自采用不同精度和实现方式的单次运行，因此我们建议将其用于排序，而不要过分依赖具体数值。

4.3 生成式模型的表现

虽然生成式模型主要用作基线，但它们仍然让我们了解到选择小型模型时的两个要点。首先，在同一模型系列中，较大的模型表现更好。例如，在 MLX 4 位类别中，Qwen2.5 模型的严格准确率从 0.5B 时的 14.00%，提升至 1.5B 时的 38.50% 、 3B 时的 53.60% 和 7B 时的 66.15%。SmolLM2 模型也会随着规模增大而表现更好。因此，尽管最小的模型所需内存很少，但它们并不适合这项任务。不过需要注意，同一系列的不同版本在训练方式和规模上可能有所不同。因此，这一趋势能帮助我们选择合适的版本，但无法单独衡量参数量本身的影响。
第二个启示是，来自不同系列的较小模型可以用低得多的成本达到较大模型的水平。例如，采用 MLX 4 位的 LFM2 2.6B 得分为 68.35%，略高于通义千问 Qwen2.5 7B 的 66.15%；后者的参数量近乎是前者的三倍。2.20 个百分点的差距本身并无实际意义，因为其配对自助法区间为 -0.05 至 $+4.40$ 个百分点，其中包含零。换言之，最稳妥的解读是，这两个模型的准确率大致相当。然而，成本差异很大。为了可靠地衡量这一差异，我们在 Mac 上开展了一项单独的研究，每个模型进行了六轮测试，每轮包含 100个问题（表 4）。LFM2 的速度约为通义千问 Qwen2.5 7B 的 2.57 倍，峰值内存用量则约少 55%。表现最佳的生成式模型 Gemma 3n E4B 比 LFM2 略准确，但每道题的耗时约为其 1.5 倍，所需内存也超过两倍。最后，为了检验重复运行本身是否可靠，全部 1,800 次计时请求的答案都与原始完整运行的答案一致。

表 4： 重复进行的 MLX 资源研究，每个模型进行六轮、每轮 100个问题。时间取六轮中位数的中位数。分配是 MLX 分配峰值，而非总 RAM。

4 位模型	完整准确率	中位数时间	轮次范围	分配峰值
LFM2 2.6B	68.35%	505.5 毫秒	503.9–507.4 毫秒	2.035 吉比字节
Gemma 3n E4B 文本	70.20%	756.1 毫秒	753.2–761.4 毫秒	4.726 吉比字节
Qwen2.5 7B	66.15%	1,301.7 毫秒	1,298.7–1,304.4 毫秒	4.562–4.563 吉比字节

作为合理性检查，我们还在同样的 2,000 个问题上，将我们的结果与原始 UAVBench 结果进行了比较。作者发布的逐题结果显示，Qwen2.5 7B 的总体得分为 66.05%，LFM2 2.6B 为 69.75%，Granite 4 H Micro 为 57.80%，Llama 3.2 3B 为 62.00%，Gemma 3n E4B 为 73.25%[2]。（论文自身仅按推理风格分组报告了这些模型 [1]。）这五个模型的得分均与我们在表3中的得分相差约三个百分点以内，这表明我们的设置运行正常。不过，

我们的数值并非精确复现。作者所提供服务的模型修订和精度与我们的不同，而且他们没有提供原始生成结果，因此我们无法使用严格规则对其重新评分。

5 在 Mac 上进行筛选，然后在开发板上测试

我们的第二个问题是：能否使用本地机器（例如 Mac）在开发板上进行测试之前筛选出有前景的候选模型？简而言之，答案是可以，但需要注意一些问题。首先，在信任任何本地得分之前，我们必须检查该得分是否有效（第5.1节）。确认有效后，Mac 能够很好地对候选模型进行排名（第5.2节）。然而，它无法预测模型会答对哪些具体问题（第5.3节）、模型在开发板上的运行速度（第5.4节），也无法预测对于决策模型而言，模型返回的确切概率（第 5.5 节）。

5.1 在信任本地得分之前检查其有效性

在使用本地得分对模型进行排名之前，我们需要确保它衡量的确实是我们认为它所衡量的内容。在实验中，我们发现得分可能通过三种方式产生误导。
第一种是量化。量化使用更少的位来存储模型的权重（此处为 4 位，而不是 16 位），从而减少模型的内存占用。理想情况下，这只会带来很小的准确率损失，而对于某些模型确实如此。例如，从 MLX FP16 切换到 MLX 4 位时，LFM2 的准确率只下降了 1.25 个百分点（从 69.60% 降至 68.35%），Granite 也只下降了 1.35 个百分点。其他模型受到的影响要大得多。通义千问2.5 3B 的准确率下降了 8.20 个百分点，SmolLM2 1.7B 则下降了 20.00 个百分点。Llama 3.2 3B 的准确率实际上略有提升，从 60.20% 升至 60.90%，但如此微小的差距并不能证明量化有帮助。我们得到的启示是，必须针对每个模型分别检查量化效果，因为节省内存的设置也可能大幅削弱模型的实用性。

第二个因素是我们如何解读模型的回答。如第3.1节所述，我们的严格规则只接受单个字母，而基准测试的宽松提取器则会在模型的文本中搜索字母。大多数时候，两者的结果是一致的。事实上，除了 SmolLM2 之外，每个模型系列的两种得分都完全相同（表3）。体量非常小的模型则有所不同。例如，采用 MLX FP16 的 SmolLM2 135M 使用宽松提取器时得分为 13.50%，但按照严格规则只有 5.05%；其 4 位版本的得分分别为 12.50% 和 0.00%。当模型写出一个句子而不是单个字母时，宽松提取器仍然可以从中找到一个字母，而这个字母有时恰好是正确答案。因此，严格得分衡量的不仅是模型是否选出了正确答案，也衡量了它是否遵循了要求的格式。实际上，这一选择主要会改变我们对最弱模型的解读。需要注意的是，我们没有原始的 UAVBench 输出，因此无法使用我们的严格规则重新评估这些模型。consequently，我们无法声称它们已发布的得分会如何变化。

第三种情况是数值故障。我们首次将 Gemma 的 FP16 权重进行 4 位转换时，在抽样的六个诊断请求中，所有请求都产生了 NaN（“非数值”）logits。换句话说，模型内部的某项计算出了问题，因此其输出不再反映任何真实的推理。不过，这种运行故障仍可能返回看起来像答案的内容。因此，如果不进行检查，它看起来就只像是一个能力较弱的模型。改为使用 BF16 权重转换 Gemma 后，诊断请求和全部 1,999 个已执行请求的 logits 均保持有限，模型的得分为 70.20%，这是最佳生成结果。对于其他配置，我们只抽样检查了部分请求的 logits 是否有限。因此，这些检查只能说明抽样请求的情况，并不能保证每个请求都有效。

5.2 Macaccuracy 为板上测试筛选候选模型

由于在开发板上测试既缓慢又昂贵，Mac 最有用的作用就是告诉我们哪些模型值得在开发板上测试。为了检验它是否能做到这一点，我们在两个平台上运行了 Qwen2.5 1.5B 和 LFM2 2.6B 的完整基准测试

，并在每个平台上使用相同的 Q4_0 file（表5）。总体准确率非常接近。Qwen2.5 1.5B 在 Mac 上的得分为 41.10%，在 RB3 上为 41.05%，仅相差一道题。LFM2 在 Mac 上的得分为 69.30%，在 RB3 上为 68.30%，这意味着 RB3 少答对了 20 道题。
随后，我们使用第3.5节所述的麦克尼马尔检验，检验这些小幅下降是否真实存在。两者均不显著（Qwen2.5 1.5B 的 $p = 1.0$，LFM2 的 $p = 0.11$）。换言之，这种幅度的差距很容易由随机因素造成，因此我们没有证据表明迁移到开发板后，任一模型的准确率有所下降。

对于筛选而言，更重要的是，排名保持不变。LFM2 在 Mac 上远远领先于 Qwen2.5 1.5B，在 RB3 上也依然遥遥领先。这支持我们先使用 Mac 淘汰较弱的候选模型并找出有潜力的模型，再投入时间在开发板上进行测试。当然，两个模型的样本量很小。此外，像 LFM2 这样下降一个百分点的情况，可能会导致两个得分接近的候选模型互换排名，因此，与其在实力相近的模型之间做出选择，Mac 更适合用于区分强模型和弱模型。

表5： 完整基准测试中的匹配 Q4_0 transfer。准确率基于全部 2,000 道问题计算。答案一致性基于已执行的 1,999 道问题计算，因为有一道过长的问题在两个平台上都计为错误。

模型	Mac 准确率	RB3 准确率	答案完全相同	FullRB3 中位数
通义千问2.5 1.5B	41.10%	41.05%	1,943 / 1,999	3.942 秒
LFM2 2.6B	69.30%	68.30%	1,775 / 1,999	6.306 秒

5.3 相近的准确率可能掩盖不同的答案

总体得分接近，并不意味着两个平台给出了相同的答案。事实上，在执行的 1,999 道题中，Qwen2.5 1.5B 在 Mac 和 RB3 上有 56 道题的答案不同，而 LFM2 有 224 道，约为每九道题就有一道。对于 Qwen2.5 1.5B，在这些题中，Mac 答对而 RB3 答错的有 13 道，反过来的有 12 道。对于 LFM2，这两个数字分别是 79 和 59。在其余答案不同的题目中，两个平台都答错了，只是答案不同。最值得注意的是，得失大体相互抵消，因此总体得分仍然接近，尽管许多单独题目的答案发生了变化。

一个自然的问题是，这些变化是否只是随机现象，也就是说，在同一平台上再次运行同一道题，是否会得到不同的答案。为此，我们针对每个模型，在每个平台上将 20 道答案不同的题目各重复运行三次（第 3.4 节）。结果表明，这些变化并非随机现象。每次重复运行时，20 处答案不一致都再次出现；并且对于每个模型，在我们测试的全部 40 道题中，同一平台各答案选项的得分在每次重复运行中都完全相同。随后，我们向两个平台输入完全相同的 token ID，从而排除了提示词分词方式或处理方式存在差异的可能。两个平台仍然选择了不同的答案。最后，我们进行了核验：开发板根据这些 token ID 给出的答案，在全部 40 道题中都与其正常生成的答案一致。

换句话说，Mac 和 RB3 分别使用同一个模型文件计算出一致的结果，但两者的结果并不相同。这与 Yuan 等人在更改 GPU 配置[23], 时发现的情况一致：数字舍入方式上的微小差异累积起来，会导致不同的输出。需要明确的是，我们并未将差异追溯到某个特定的内核或舍入步骤。

实践中得到的教训是，即便使用相同的模型文件、提示词和随机种子，在不同计算机上也无法保证做出相同的决策。Mac 可以告诉我们某个模型总体上很有前景。不过，在实际硬件上测试时，仍需关注哪些答案发生了变化，而不只是总体得分。这对于高后果决策尤其重要。例如，在图1所示的情境中，开发板上运行 LFM2 的无人机可能会做出与 Mac 版本不同的判断。

5.4 Mac 延迟无法预测开发板延迟

正如预期，开发板的速度远慢于 Mac（表6）。Qwen2.5 1.5B 在 Mac 上每个问题的中位耗时约为 0.3 秒，而在 RB3 上约为 4.0 秒，长约 12.7 倍。LFM2 在 Mac 上耗时约为 0.5 秒，在 RB3 上约为 6.3 秒，同样长约 12.1 倍。这些测量结果也非常稳定。在六轮测试中，开发板上的中位耗时彼此相差仅几毫秒，而且两个模型在每一轮中对全部 100个问题都给出了相同的答案。

由于两个模型的速度下降幅度相近，因此很容易将“慢约 12 倍”视为把 Mac 测得的时间换算为开发板测得时间的规则，但我们不建议这样做。速度下降幅度取决于设备、运行时和模型。例如，在同一块开发板上，H2O 的速度下降了约 17.3 倍（第 5.5 节）。因此，Mac 上的计时可以告诉我们哪些模型更快，但只能进行相对比较。模型在开发板上的实际速度必须在开发板上测量。

表6： 重复匹配计时，单位为毫秒。每个单元格汇总了 100个问题的六轮测试结果。范围涵盖六轮测试的中位数。

模型	Mac 中位数	Mac 范围	RB3 中位数	RB3 范围
Qwen2.5 1.5B	313.7	312.5–314.0	3,979.2	3,977.2–3,980.9
LFM2 2.6B	525.8	525.7–526.4	6,338.4	6,333.4–6,339.2

我们还记录了 GPU 温度，因为开发板过热时可能会自行降速以便冷却（这称为降频）。该开发板有两个 GPU 传感器。在计时轮次期间，Qwen2.5 1.5B 的读数分别为 55.4–69.8 C 和 58.1–79.1 C，LFM2 的读数分别为 55.8–74.0 C 和 56.6–83.7 C。完整的 2,000 个问题运行过程温度更高，Qwen2.5 1.5B 的峰值分别达到 75.2 C 和 84.9 C，LFM2 的峰值分别达到 80.6 C 和 91.1 C。不过，完整运行和计时轮次在持续时间和初始温度方面有所不同，因此我们无法根据这些读数判断哪个模型运行时温度更高。此外，温度较高本身并不能证明开发板发生了降频。我们在计时轮次中没有发现降频迹象，因为其中位数时间在各轮次之间几乎保持不变。

5.5 H2O 能迁移准确率，但无法迁移概率或速度

H2O 是我们唯一成功在 RB3 上运行的决策模型（第 3.4 节）。由于首次概率检查未通过，而且开发板运行速度较慢，我们没有使用完整基准测试，而是在固定的 100 个问题集上对它进行了测试（见下方表 7）。在这些问题上，答案迁移效果良好。两个平台都答对了 100 个问题中的 80 个，并且在 97 个问题上给出了相同的答案。在剩余的三个问题中，每个平台各有一个问题答对而另一个平台答错；对于第三个问题，它们给出了不同的错误答案。

概率的迁移效果则没那么好。这对于决策模型来说很重要，因为概率是其主要优势之一。例如，系统可能只会在模型给出的某个选项的概率高于特定阈值时才采取行动。我们发现，在 100 个问题中，有 32 个问题至少有一个选项的概率在两个平台之间相差超过 0.01。最大差异为 0.1718，足以轻易使某个选项的概率跨过 0.5 这样的阈值。H2O 的发布方提供了自己的探测问题，用于检查转换是否成功，这些问题都通过了。不过，发布方的探测问题允许差异最高达到 0.05，容差比我们的 0.01 更宽松。因此，我们在首次检查中发现的概率不匹配问题仍未解决。

速度方面的差距最为严重。H2O 在 Mac 上每道题的耗时中位数为 0.967 秒，而在 RB3 上为 16.762 秒，约长 17.3 倍。最慢的题目耗时更久：在开发板上，每 20 道题就有一道耗时近 20 秒（第 95 百分位数为 19.809 秒）。粗略估算，按照开发板的平均耗时，完整运行 2,000 道题约需 9.35 小时。请注意，我们并未测量

表 7： H2O 在两个平台上使用相同的 Q4_K_M GGUF 文件，针对相同的固定 100 个问题进行测试。这些是诊断结果，并非完整基准测试准确率。

测量	Mac	RB3
正确答案	80 / 100	80 / 100
响应时间中位数	0.967 秒	16.762 秒
平均响应时间	0.980 秒	16.827 秒
第 95 百分位数	1.150 秒	19.809 秒
TimedGPU 传感器 0 范围	未测量	47.3–62.4°C
TimedGPU 传感器 1 范围	未测量	46.9–70.9°C

这一点。我们的测试集输入最长也只有 571 个词元，因此更长的问题尚未经过测试。同样，我们也不声称测试集的 80% 准确率能够在完整基准测试中保持。

6 软件和资源限制决定部署

我们的第三个问题是，是什么阻碍了这些模型的实际部署。简而言之，Mac 结果告诉我们哪些模型值得部署，却没有告诉我们哪些模型能够部署。在我们的四个决策模型中，有三个从未在开发板上回答过任何问题，而第四个 H2O 则需要采用多种变通方法才能做到。本节将介绍每个模型出了什么问题，因为每次失败都揭示了不同的障碍。

6.1 GLiDE 0.4B：运行时无法完成计算图定型

GLiDE 0.4B 是该开发板最有希望的候选模型，因为它是 Mac 上速度最快的决策模型，同时准确率也相当不错。其导出结果也通过了我们所有的本地检查。然而，它在开发板本身上运行失败。在运行模型之前，开发板的运行时（此处为其 QNN GPU 路径）必须将导出的模型编译成 GPU 可以执行的形式，这一步称为计算图定型。GLiDE 0.4B 在这一步失败了。我们尝试了两种改变模型索引和掩码表示方式的改造方案（一种使用整数索引，另一种使用布尔掩码）。在我们的试点问题上，两种方案都保持了模型的概率不变，但都没有解决定型失败的问题。为了确认 GPU 路径本身能够正常工作，我们还在关闭 CPU 回退的情况下，在开发板上运行了一次小型矩阵乘法，结果成功。这表明 GPU 后端可以处理小型操作，但并不能证明该后端支持完整模型。最终，我们无法在 RB3 上完成 GLiDE 0.4B 的任何一次基准测试运行。

6.2 GLiDE 0.8B：一个不受支持的算子

GLiDE 0.8B 比 GLiDE 0.4B 更早失败，因为它根本没能部署到开发板上。要在开发板上运行模型，我们首先必须导出模型，也就是将模型的每项数学运算（称为算子）转换为开发板运行时能够识别的格式。如果导出工具不支持某个算子，导出就会停止。GLiDE 0.8B 的循环注意力层使用了一种称为三角求解的算子，而我们选择的 ONNX 导出路径不支持该算子。我们本可以将 GLiDE 0.8B 的混合注意力替换为普通的因果解码器，导出器支持这种解码器。不过，这会改变模型，因此我们测试的就不再是 GLiDE 0.8B 了。所以我们停止了尝试。需要说明的是，这是我们尝试的导出路径的限制，并不能证明 GLiDE 0.8B 无法在任何边缘加速器上运行。

6.3 Liquid d1：容差检查未通过

Liquid d1 也遇到了导出问题，不过这个问题有办法绕过。它的注意力层使用了导出器不支持的分组查询注意力调用（enable_gqa=True）。因此，我们将该调用改写成导出器能够处理的形式。为确认改写没有改变模型，我们在十个试点问题上将改写后的版本与原始版本进行了比较。改写后的模型和导出版本在全部十个问题上的选择都与原始版本相同。不过，概率并不完全一致。有两个问题中，导出版本的概率与原始版本的差异超过了我们预先设定的 0.001 容差，最大差异为 0.00207。因此，我们将其判定为检查未通过，没有在开发板上运行 Liquid d1。这些差异很小，而且选择确实一致。不过，仅凭十个问题上的选择一致，并不足以成为悄然放宽我们事先根据概率设定的规则的理由

6.4 H2O：单次分配的上限

H2O 最终在开发板上运行，但前后进行了三项修正。首先，我们确保导出的模型保留了 H2O 的注意力行为。其次，模型文件声明了一个 H2O 不使用的额外数据块，而这项声明导致运行时无法加载模型。我们移除了这项声明，只修改了文件的元数据，并确认模型权重的每个字节都未改变。第三，也是最重要的一点，RB3 运行时在单次 GPU 内存分配中最多只能分配 256 MiB。H2O 有一个约 497 MiB 的张量。该张量存放词嵌入，模型的输入层和输出层共用这些词嵌入。由于无法在单次 GPU 内存分配中容纳它，该张量只能留在 CPU 上。因此，H2O 的部分计算在 CPU 上执行，其余部分则通过 OpenCL 在 GPU 上执行，所以这并不是纯 GPU 的运行结果。

请注意，这个 256 MiB 的限制与开发板的总内存（约 5.25 吉比字节）并不相同，也不同于 GPU 驱动程序报告的内存数值。事实上，H2O 在 GPU 上的缓冲区总计约为 2,004 MiB，远超 256 MiB，因为这些缓冲区被拆分成许多较小的分配。打个比方，停车场的总车位可能很多，但没有任何一个车位大到足以停下一辆巴士。因此，对于“能否装下？”这个问题，答案各不相同，取决于我们问的是哪一种限制。

6.5 RB5：驱动程序和运行时问题

我们还尝试了一块 RB5 开发板，以查看相同的软件能否在其上运行。运行时检测到了开发板的 Adreno GPU，但在查询 GPU 的子组支持时遇到了接口问题，随后又无法编译其 GPU 内核。因此，没有任何一个基准测试问题能够完成。这说明的是该开发板上安装的驱动程序与运行时的组合情况，并不能说明 RB5 的准确率，也不能说明其硬件能否通过不同的软件运行这些模型。

6.6 什么样的模型适合作为部署候选项

综上，这些案例表明，对部署候选项需要采用更广义的定义。当然，候选项必须在任务上达到足够好的表现。不过，它还必须跨过其他几道门槛，而我们每个失败的模型都分别卡在了不同的门槛上。它必须符合开发板的内存限制，包括对任何单次分配的限制，而这正是 H2O 遇到的问题。它必须在转换过程中保持其计算完整性，而这正是 Liquid d1 遇到的问题。最后，它只能使用目标运行时支持的算子，而这正是 GLiDE 0.8B 遇到的问题。模型的参数量和文件大小只能反映这一整体情况的一部分。我们报告这些失败的尝试，是因为在实践中，正是这些因素决定了我们那些有前景的本地模型是否能够在开发板上进行测试。

7 讨论

7.1 决策模型结果的额外价值

我们最明确的新发现是，小型开放权重决策模型是处理基于文本的无人机决策的有力候选项。例如，GLiDE 0.8B 在 Mac 上以约四分之一秒的调用时间达到了 73.15% 的准确率，而 H2O 达到了 77.65%。然而，我们的部署结果带来了一个重要的转折，即纸面上最好的候选项未必最容易部署。GLiDE 0.8B 在 Mac 上看起来很有前景，但我们无法通过现有的导出路径将其部署到 RB3 上。H2O 确实能在 RB3 上运行，但只有部分模型运行在 CPU 上，响应时间大幅延长，而且其概率与 Mac 上的概率出现了偏差。因此，模型能否在目标设备上实际运行，以及能否忠实运行，必须成为模型选择的一部分，而不能等到事后再检查。

我们也尚未厘清决策模型得分如此之高的原因。正如我们在第4.1节所指出的，这两组模型在训练、提示词、精度、注意力和推理软件方面各不相同。不过，有一点差异格外突出：生成式基线会写出自由文本，而决策模型则必须从七个选项中选出一个。后续研究可以让生成式模型也享有同样的优势。也就是说，我们不让生成式模型直接写出答案，而是读取它分配给字母 A 到 G 各自的概率，并按照 Robinson 和 Wingate[15] 的思路，将概率最高的字母作为答案。例如，对于图1中的问题，Gemma 就不再会写一句关于风和高度的话，而只会像决策模型那样给出七个概率。如果经过这一调整，决策模型仍然领先，那么优势就来自模型本身。如果差距缩小，那么其中很大一部分优势就来自接口。

7.2 实用的本地筛选流程

我们的结果表明，在部署模型之前，可以采用分阶段流程在本地进行筛选。首先，在比较任何模型之前，先固定问题、标签、令牌数量限制和评分规则，确保以相同方式测量每个模型。其次，在查看其准确率之前，检查每次运行在数值上是否有效且是否符合资源预算，因为正如第5.1节所示，一次出错的运行可能会让一个模型看起来很弱。第三，利用本地结果筛选出少数几个准确率和速度符合任务需求的候选模型。最后，在目标设备上为每个候选模型构建忠实版本，在固定的输入集上重复数值检查，然后再收集准确率和计时结果。

在目标硬件上进行验证时，我们还建议报告不同平台上的答案有多大比例一致，而不仅仅是总体准确率。正如我们的Qwen2.5 1.5B和LFM2结果所显示的，即使许多单独的答案发生变化，总体得分仍可能非常接近。对于返回概率的模型，还应比较概率，而不仅是选择结果。对于计时，应在一组共享的问题上重复测量，明确说明计时器包含哪些环节，并记录温度。最后，如果目标设备上的版本在精度或运行位置上不同于本地版本（例如，模型的一部分运行在CPU上），就应将其视为一种新条件，并为其构建匹配的本地参考版本，正如我们对H2O所做的那样。

7.3 anonboardclaim 需要什么

对于无人机而言，响应时间、温度和功耗都很重要，但它们回答的是不同的问题，而我们只测量了前两项。功耗之所以重要，是因为无人机的电池限制了其飞行时长。我们保留了 RB3 的原始功耗读数，但尚未核实其单位或它们具体测量的内容。因此，我们不提供能量数据，也不对这些模型会如何影响飞行时间作出任何断言。

时间是另一个差距。我们设想的用例是偶尔进行高层次的任务决策，而不是飞行控制。即便如此，RB3 上速度最快的生成式模型每次决策也耗时约四秒，

而 H2O 约需十七秒。这是否足够快，取决于任务需要多久进行一次决策，以及无人机在等待期间会做什么。例如，在图1所示的场景中，无人机若等待十七秒才能得到答复，就会在强风和电池电量有限的情况下继续执行当前计划。因此，真正的部署研究应测量模型在规定的截止期限和持续使用条件下，以及开发板同时忙于感知或导航等其他工作时，能多频繁地及时选择正确的动作。

8 局限性与后续实验

Rat与其在全文中反复提及各种注意事项，我们在此汇总主要事项 e.
基准测试。UAVBench 用文本描述每种情境，并直接给出无人机的状态（例如高度和电池电量）。在真实无人机中，这些状态需要由传感器提供，而传感器可能会产生噪声或发生故障。该基准测试也不会检验模型采取行动后的情况，也不会检验无人机的决策如何反馈到下一个情境中，进而影响其飞行（闭环飞行）。同样，基准测试中的标签只表明作者将哪个答案标记为正确，并不能证明该答案在真实飞行中是安全的。例如，图1中问题的标准答案是 D，但仅凭标签无法说明在那个特定空域中下降到 60 米是否安全。最后，十种推理风格的准确率差异很大。例如，经过校正的 Gemma 模型在推理风格 1、3 和 6 上的得分分别为 47.0%、51.0% 和 95.0%。因此，整体得分良好也可能掩盖特定推理类型中的严重不足。

比较。如第 4.1 节和第 7.1 节所述，生成模型和决策模型在提示词、精度、运行时间以及计时器涵盖的内容上各不相同。因此，我们的比较都无法单独衡量模型设计或模型大小的影响。我们的候选模型范围也很广，但并不全面。此外，在研究的不同阶段，模型修订和精度有所变化；而且，我们是在 Gemma 失效后才加入部分数值检查的。当我们因模型超出内存预算而将其排除时，这意味着它在我们设定的限额下无法运行，并不意味着它永远无法在 Mac 上运行。最后，我们尚未测量校准情况。在完成测量之前，不应将决策模型给出的概率视为置信度的真实衡量标准。也就是说，模型给某个选项分配 0.9 的概率，并不意味着它有 90% 的时候都是正确的。

测量结果。我们的重复计时数据来自一台 Mac 和一块 RB3 上的单次会话，而且我们没有充分控制这些设备的室温或其他后台任务。我们确实发现，Mac 已经在使用一些交换内存（即内存不足时用作溢出空间的磁盘空间）。不过，我们没有记录模型本身是否在换入和换出，因此无法排除这对计时结果造成影响的可能性。在 RB3 上，我们在预热请求之前进行了散热检查，因此各轮计时开始时的温度并不完全相同。H2O 的结果来自一个仅包含 100个问题的探索性测试集。最后，通过我们尝试的途径，GLiDE 和 Liquid d1 均未能在开发板上产出结果。这并不意味着这些模型的效率如何。

后续实验。我们的首要任务是让 GLiDE 0.8B 或 Liquid d1 在边缘设备上忠实运行。尝试第二种硬件（例如 NVIDIA Jetson）也有助于判断，我们观察到的故障是否特定于所使用的 Qualcomm 软件。除了部署之外，我们还计划按照第 7.1 节所述，根据生成式模型为每个选项字母给出的概率对其进行评分。我们还计划在未用于调优的问题上测量校准和弃答情况，其中，弃答是指模型在不确定时拒绝回答。最后，在对能量使用或实际部署做出任何结论之前，我们需要在开发板的电源输入处采集经过验证的功耗测量结果，进行更长时间的热测试，并开展这样的测试：开发板同时运行真实无人机所需的工作，例如感知和导航。

9 结论

在本文中，我们围绕在无人机的计算机上运行小型语言模型提出了三个问题。首先，我们询问小型结构化决策模型是否适合用于无人机任务决策。我们的发现是：适合。我们测试的四个开放权重发布版本在 UAVBench 上表现良好，且未经过我们自己的任何训练；其中三个版本的表现超过了我们尝试的所有生成式配置，最佳的 H2O 达到了 77.65%。其次，我们询问 Mac 是否能够在上开发板测试之前筛选候选模型。可以将其作为第一轮筛选，因为它对两个匹配模型的排名与 RB3 相同。然而，同一个模型文件迁移到开发板后，部分回答仍然发生了变化，而且运行速度大约慢了 12 至 17 倍。第三，我们询问部署面临哪些障碍。当我们尝试移植这些决策模型时，四个模型中有三个因三种不同的原因而始终无法在开发板上回答问题，而这些原因都无法通过模型大小预测。

综上所述，我们建议先在本地筛选模型，然后在目标计算机上验证实际模型文件，并按照其在生产环境中的运行方式进行测试。最后，这些结果说明了哪些模型是合适的候选，以及哪些因素限制了它们的部署。但它们并不能证明这些模型中的任何一个能确保自主飞行安全，也不能证明其能节省能量。

参考文献

[1] M. A. Ferrag、A. Lakas 和 M. Debbah。《UAVBench：通过大语言模型生成的飞行场景，为自主与智能体式人工智能无人机系统构建的开放基准数据集》。IEEE《开放式车辆技术期刊》，7:2542–2558，2026年。doi:10.1109/OJVT.2026.3721777。预印本：arXiv:2511.11252。[2]UAVBench 作者。已发布的基准测试源代码和评估数据。修订版 4f2baa78a50eea3bb63dffb83565b8f9fb7a7a77。
`https://github.com/maferrag/UAVBench/tree/4f2baa78a50eea3bb63dffb83565b8f9fb7a7a77`[3] TypeSafe AI。System One 模型与类型化决策。产品网站和文档。访问日期：2026年11月12日。
`https://typesafe.ai/` 以及 `https://docs.typesafe.ai/introduction`[4] Fastino。GLiDE-0.4B 模型卡、已发布的权重和实现。附录 B 中固定的修订版。`https://huggingface.co/fastino/GLiDE-0.4B`。修订版 6bda7ab91aeb36dcad6d27828a8871e548c7fc71。[5] Fastino。GLiDE-0.8B 模型卡、已发布的权重和实现。附录 B 中固定的修订版。`https://huggingface.co/fastino/GLiDE-0.8B` 修订版 9eacdeec79fa48ec52a8d6f54b331376b4bbfaf0。[6] Liquid AI。d1-2B 模型卡、已发布的权重和实现。附录 B 中固定的修订版。 `https://huggingface.co/LiquidAI/d1-2B` 修订版 652f920e544ea37f81e4b0ce89b81d4ed0da90b1。[7] H2O.ai。H2O-Lightning-4B 模型卡及已发布的实现。
`https://huggingface.co/h2oai/h2o-lightning-4b`MLX 实现修订版 420ddaf0c87b08c67801b047567b073b182ee665。[8] S. Vemprala、R. Bonatti、A. Bucker 和 A. Kapoor。《ChatGPT for Robotics：设计原则与模型能力》。arXiv:2306.17582，2023年。
`https://arxiv.org/abs/2306.17582`[9] G. Chen、X. Yu、N. Ling 和 L. Zhong。《TypeFly：使用大语言模型驾驶无人机》。arXiv:2312.14950，2023年。 `https://arxiv.org/abs/2312.14950`[10] S. Javaid、N. Saeed 和 B. He。《无人机大语言模型：现状与未来发展路径》。arXiv:2405.01745，2024 年。
`https://arxiv.org/abs/2405.01745`[11] B. Ichter、A. Brohan、Y. Chebotar 等。《照我所能做，而非照我所说：将语言建立在机器人可供性之上》。机器人学习会议（CoRL），PMLR 205:287–318，2022年。另见 arXiv:2204.01691（该版本列 M. Ahn 为第一作者）。 `https://arxiv.org/abs/2204.01691`[12]T. Deußer、L. Sparrenberg 和 R. Sifa。《对 System One 模型 Jev 的评估与基准测试》。arXiv:2609.37647，2026年。
`https://arxiv.org/abs/2609.37647`

[13] Y. Qu、J. Chen 和 S. Labi。《Jev with Foresight：借助语言世界模型实现安全的语言引导自动驾驶》。arXiv:2611.05448v1，2026年。 https://arxiv.org/abs/2611.05448v1[14] J. Delaunay。《System 1 模型的置信度与决策是否可信？对 Jev 风格决策模型的可解释性与扰动审计》。arXiv:2611.12521v1，2026年。 https://arxiv.org/abs/2611.12521v1[15] J. Robinson 和 D. Wingate。《利用大语言模型进行选择题问答》。国际学习表征会议（ICLR），2023年。arXiv:2210.12353。 https://arxiv.org/abs/2210.12353[16] C. Zheng、H. Zhou、F. Meng、J. Zhou 和 M. Huang。《大语言模型并非稳健的选择题选择器》。国际学习表征会议（ICLR），2024年。arXiv:2309.03882。https://arxiv.org/abs/2309.03882[17] C. Guo、G. Pleiss、Y. Sun 和 K. Q. Weinberger。《现代神经网络的校准》。国际机器学习会议（ICML），PMLR 70:1321–1330，2017年。[18] J. Xu、Z. Li、W. Chen、Q. Wang、X. Gao、Q. Cai 和 Z. Ling。《端侧语言模型：综合综述》。arXiv:2409.00088，2024年。 https://arxiv.org/abs/2409.00088[19] E. Frantar、S. Ashkboos、T. Hoefler 和 D. Alistarh。《GPTQ：生成式预训练变换器的精准训练后量化》。国际学习表征会议（ICLR），2023年。arXiv:2210.17323。[20] J. Lin、J. Tang、H. Tang、S. Yang、W.-M. Chen、W.-C. Wang、G. Xiao、X. Dang、C. Gan 和 S. Han。《AWQ：用于端侧大语言模型压缩与加速的激活感知权重量化》。机器学习与系统会议论文集 6（MLSys），87–100，2024年。arXiv:2306.00978。[21] ggml-org。llama.cpp：使用 C/C++ 进行大语言模型推理。软件代码库。 https://github.com/ggml-org/llama.cpp[22] A. Hannun、J. Digani、A. Katharopoulos 和 R. Collobert。《MLX：在 Apple 芯片上实现高效且灵活的机器学习》。软件代码库，2023年。 https://github.com/ml-explore/mlx[23] J. Yuan、H. Li、X. Ding、W. Xie、Y.-J. Li、W. Zhao、K. Wan、J. Shi、X. Hu 和 Z. Liu。《理解并缓解大语言模型推理中非确定性的数值来源》。神经信息处理系统进展（NeurIPS），2025年。arXiv:2506.09501。 https://arxiv.org/abs/2506.09501[24] NetProjects（Internet Projects Ltd）。H2O-Lightning-4B MLX q8 转换，由 H2O-Lightning-4B 模型卡提供链接；与 H2O.ai 无关联。附录 B 中固定的修订版。 https://huggingface.co/netprojects/h2o-lightning-4b-mlx-q8 修订版 228e6e97c3c5f6bff1a29f1f6786cab350dd0383。[25] H2O.ai。H2O-Lightning-4B GGUF 发布版。 https://huggingface.co/h2oai/h2o-lightning-4b-gguf修订版 61f8d8b7a8c2d84426d0d143ca187f65d235934b。[26] Q. McNemar。《相关比例或百分比之差的抽样误差说明》。Psychometrika 12(2):153–157，1947年。doi:10.1007/BF02295996[27] T. G. Dietterich。《比较监督分类学习算法的近似统计检验》。Neural Computation 10(7):1895–1923，1998年。doi:10.1162/089976698300017197