CREATE: 面向高效且可靠的具身人工智能系统的跨层韧性表征与优化

Tong Xie
北京大学集成电路学院 中国北京
xietong@stu.pku.edu.cn

Yijiahao Qi
北京大学电子工程与计算机科学学院 中国北京
qiyijiahao@stu.pku.edu.cn

Jinqi Wen
北京大学电子工程与计算机科学学院 中国北京
2200012451@stu.pku.edu.cn

Zishen Wan
佐治亚理工学院 美国佐治亚州亚特兰大
zishenwan@gatech.edu

Yanchi Dong
北京大学集成电路学院 中国北京
dongyanchi@pku.edu.cn

Zihao Wang
北京大学人工智能研究院 北京, 中国
zhwang@stu.pku.edu.cn

Shaofei Cai
北京大学人工智能研究院 北京, 中国
caishaofei@stu.pku.edu.cn

Yitao Liang
北京大学人工智能研究院 北京, 中国 yitaol@pku.edu.cn

Tianyu Jia
北京大学集成电路学院, 中国北京
tianyuj@pku.edu.cn

Yuan Wang
北京大学集成电路学院，中国北京
wangyuan@pku.edu.cn

Runsheng Wang
北京大学集成电路学院, 中国北京
r.wang@pku.edu.cn

Meng Li*
北京大学人工智能研究院，中国北京
meng.li@pku.edu.cn

arXiv:2601.14140v1 [cs.AR] 20 Jan 2026

摘要

近年来，具身人工智能（AI）因将人工智能与物理世界相连接而受到广泛关注。现代具身人工智能系统通常结合基于大语言模型（LLM）的规划器来进行高层任务规划，并采用基于强化学习（RL）的控制器生成低层动作，使具身智能体能够在真实世界环境中应对复杂任务。然而，具身智能体的部署仍面临挑战，因为它们需要大量计算资源，尤其是在电池供电的本地设备上。尽管降低工作电压等技术可以提高能量效率，但也可能引入位错误，导致任务失败。

在本文中，我们提出了 CREATE，这是一种通用设计原则，利用不同层级的异构韧性，实现能耗-可靠性协同优化。我们首次对现代具身人工智能系统开展了全面的错误注入研究，并观察到其固有但具有异构性的容错能力。基于这些发现，我们在电路级提出了一种异常检测与清除 机制，以消除离群错误。在模型层级，我们提出了一种权重旋转增强规划 算法，以提升基于大语言模型的规划器的容错能力。此外，我们引入了一种应用级技术——自主性自适应电压缩放，用于动态调节控制器的工作电压。我们对电压调节电路进行协同设计，以支持在线电压调节。大量实验表明，在不影响任务质量的情况下，与标称电压基线相比，CREATE 平均可节省 40.6% 的计算能耗；与现有技术相比，平均可节省 35.0%。这进一步带来了 29.5% 至 37.3% 的芯片级节能，以及约 15% 至 30% 的电池续航时间提升。

*通讯作者

本作品采用知识共享署名 4.0 国际许可协议。*ASPLOS '26*，*2026*年*3*月*22*日至*26*日，美国宾夕法尼亚州匹兹堡©2026 版权所有者/作者所有。ACM ISBN 979-8-4007-2359-9/2026/03https://doi.org/10.1145/3779212.3790147

CCS 概念： • **硬件** → **鲁棒性**； • **计算机系统组织** → **体系结构**；**嵌入式与信息物理系统**。

关键词： 具身人工智能系统；效率；可靠性；韧性表征；跨层优化

ACM 参考文献格式：

Tong Xie、Yijiahao Qi、Jinqi Wen、Zishen Wan、Yanchi Dong、Zihao Wang、Shaofei Cai、Yitao Liang、Tianyu Jia、Yuan Wang、Runsheng Wang 和 Meng Li。2026 年。CREATE: 面向高效且可靠的具身人工智能系统的跨层韧性表征与优化。载于 *第 31 届 ACM 编程语言与操作系统架构支持国际会议论文集，第 2 卷（ASPLOS '26），2026年3月22日至26日，美国宾夕法尼亚州匹兹堡。* ACM，美国纽约州纽约市，18 页。

https://doi.org/10.1145/3779212.3790147

1 引言

具身人工智能（AI）是一类能够理解、推理并与物理世界交互的智能系统 [1–3]。这些系统擅长执行复杂的长时程多目标任务，已展现出广泛的应用潜力，涵盖工业制造 [4], 物流与运输 [5], 以及搜救行动 [6]。如图1(a) 所示，为实现这些能力，具身智能体通常采用大语言模型（LLM）或多模态语言模型（MLM）作为规划器，将整个任务分解为具体的子任务，同时采用基于强化学习（RL）的控制器来确定每个子任务的低层动作，这通常需要数千个步骤 [4, 7–12]。任务性能通常通过成功率和平均完成步数来衡量 [4, 9–13]。

与由预定义规则驱动的传统机器人系统不同， [14, 15], 具身智能体利用大规模大语言模型，而每次推理往往需要执行数万亿次运算。尽管通过 API 进行云端推理能够使用 GPT-4 [16], 等大型大语言模型，但这会带来较高的通信延迟，并依赖 Wi-Fi，从而限制了许多应用所需的实时交互能力。因此，近期研究开始探索将具身智能体端到端部署在边缘硬件上，例如 GPU [8, 17, 18] 和定制加速器 [4, 7, 13, 19, 20], , 如 TPU类脉动阵列 [21, 22]。

然而，由于依赖电池，具身人工智能系统的边缘部署通常受到严格的能耗限制，池电量 [1, 19, 20, 23, 24]。面对密集的计算工作负载（例如长上下文大语言模型推理和高帧率图像处理），据大量报告显示，这些计算平台会消耗大量功耗 [4, 8, 13, 14, 25–30]。例如， [30] 报告称，四足机器人的计算平台占总功耗的 50% 以上 [31], ； [13] 则证明，在采用 NVIDIA GPU 和 Franka 机械臂的典型机器人配置中，大语言模型推理占据了总能耗的主要部分 [26, 27, 32]。因此，提高具身人工智能系统的功率效率势在必行。鉴于功耗与工作电压呈平方关系，

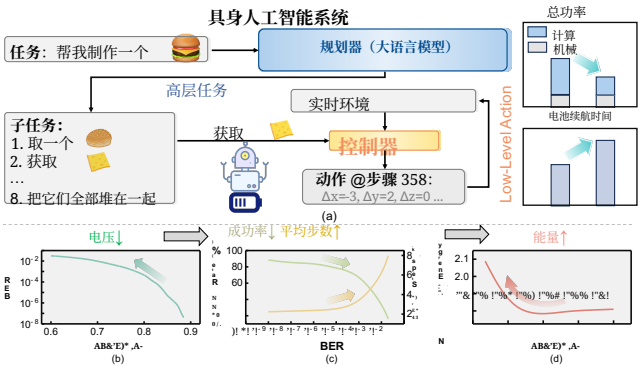

图 1. 具身人工智能系统概览。 (a) 具身人工智能系统采用分层范式来处理复杂任务：基于大语言模型的规划器将任务（通常以文本描述）分解为多个子任务；然后，低层控制器针对每个子任务逐步生成详细动作。这些系统通常由电池供电。(b) 降低工作电压会引入位错误，导致 (c) 任务性能显著下降（例如，任务成功率降低、任务执行步数增加）。(d) 这反过来会导致每项任务的能耗增加。

电压，直接降低电压可以提高处理效率。然而，这种方法可能导致时序违规 [33–36] 并提高计算位错误率（BER）1，如图所示(b)，从而严重降低任务性能，例如降低任务成功率并增加执行步骤（图 1(c)）。这些因素最终可能增加完成单个任务所需的能耗（图1(d)。

已有多种技术被提出，用于在电路、算法和系统层面平衡可靠性与效率[37–56], 但对于具身人工智能系统而言，这些技术仍然成本高昂，或不切实际。与传统机器人学不同，现代具身智能体主要基于人工智能和深度神经网络 (DNNs)，而这类网络本身具有错误弹性 [53, 57–62]。例如，如图 1(c) 所示，在低比特错误率 (石头中 [11])，任务成功率不会立即下降，这表明可以通过有策略地降低工作电压来节省能量，同时不损害任务性能。然而，具身人工智能系统的错误弹性尚未得到充分表征，因此难以确定合适的电压缩放策略。尽管先前研究考察了单个深度神经网络（例如，卷积神经网络 (CNNs) 、强化学习算法等）的韧性 [57–60] 以及传统机器人系统的韧性 [55, 56], 具身人工智能系统中不同模型之间复杂的协作，以及其中复杂的应用，可能产生本质上不同的韧性表现。

因此，为了通过电压缩放实现高效且可靠的具身人工智能系统，我们首次开展大规模错误注入实验，系统地表征其错误弹性。我们的研究揭示了系统鲁棒性固有但存在差异的特点：虽然

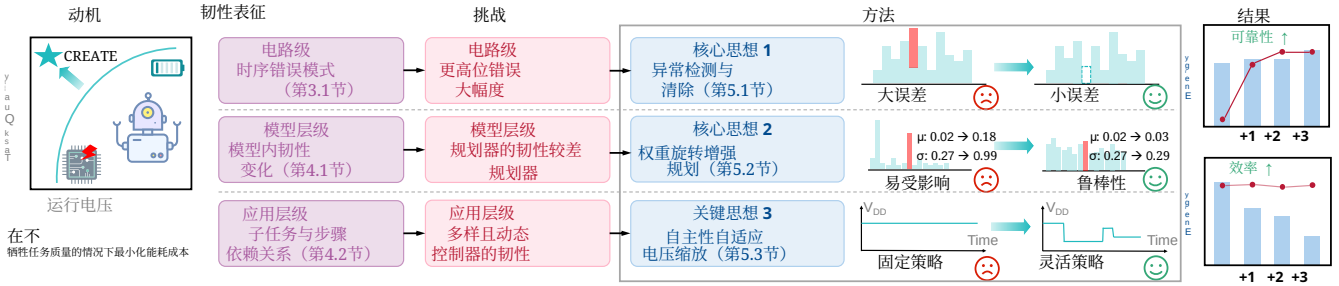

图2。CREATE概览。CREATE是一项通用设计原则，旨在实现高效且可靠的具身人工智能系统。在电路级，异常检测与清除可减轻较大的时序错误。在模型层级，权重旋转增强规划重新分配大语言模型激活值，以提升鲁棒性。在应用层级，自主性自适应电压缩放根据任务需求动态调整电压。这些技术协同提升了可靠性和效率。

规划器和控制器在低误码率下（例如 $\leq 10^{-7}$）都表现出良好的错误鲁棒性，但控制器在较高误码率下（例如从 10^{-7} 到 10^{-3}）展现出强得多的韧性。我们还分析了规划器和控制器中不同网络组件的韧性差异，以及韧性对任务和执行状态的依赖性。我们观察到，大语言模型中系统性的激活异常值与后续归一化操作相结合，是导致规划器在较高误码率下韧性较差的根本原因。与此同时，控制器在任务执行期间，不同子任务和动作步骤呈现出不同的韧性模式。

基于这些洞见，我们提出了CREATE，这是一项通用设计原则，如图2所示，通过协同整合电路、模型和应用层级的优化，使各层级逐步协同工作。在**电路级**，我们引入了异常检测与清除机制，以抑制由时序违例引起的大误差。尽管这一机制在概念上较为简单，但它为后续优化奠定了坚实基础。然而，由大语言模型规划器的激活分布产生的较小但仍然关键的错误依然存在，并会导致性能下降。为解决这一问题，我们在**模型层级**提出了权重旋转增强规划，以重新分配大语言模型规划器的激活模式并提升鲁棒性。为了进一步优化控制器的能效，我们在**应用层级**提出了自主性自适应电压缩放，该方法根据当前子任务执行状态采用灵活的电压控制策略。我们还针对脉动阵列和低压差稳压器（LDO）定制了用于动态电压缩放的电路，从而整体实现这些优化。我们的贡献总结如下：

- 我们对具有代表性的具身人工智能系统开展了全面的错误注入研究，以分析电压缩放引起的错误对任务性能的影响。分析表明，控制器的错误弹性显著优于规划器。我们还进一步研究了不同网络组件和执行步骤中的韧性行为。据我们所知，这是首个关于具身人工智能系统错误弹性的全面研究。
- 我们提出 CREATE，这是一项通用设计原则，通过跨层优化利用具身 AI 系统异构的错误韧性，在不损害其可靠性和任务性能的前提下，最大限度地降低能耗。该框架通过定制电路和低压差稳压器实现。
- 大量实验表明，CREATE 显著提升了容错能力和能量效率，并且能够适用于多种任务和电压条件。通过大幅降低工作电压，CREATE 相较于标称电压运行，计算能效平均提升 40.6%；相较于最先进（SOTA）方法，平均提升 35.0%，相当于芯片级节能 29.5% 至 37.3%，电池续航时间延长约 15% 至 30%，且任务质量保持不变。

2 背景

2.1 具身智能体概述

具身智能体是一类依托物理实体来理解、推理并与物理世界交互的智能系统 [1–3]。具身智能体利用大语言模型的泛化和推理能力来处理复杂任务，通常由两个核心组件构成：规划器和控制器 [4, 7–12]。

规划器**通常是微调后的大语言模型或掩码语言模型，负责理解任务并将其分解为一系列具体的子任务，这一过程称为高层规划。这些子任务随后会传递给控制器**，通常是用于低层控制的较小型强化学习模型。控制器会在每个步骤中根据其输出的动作逻辑值采样动作。这一范式已在近期研究中得到广泛采用 [4, 7–9, 11, 12, 77]，，如表1所示。与此同时，端到端视觉-语言-动作（VLA）模型是另一种常见方法，它直接生成动作令牌或调用 API [10, 26, 27]。此外，许多研究聚焦于用于低层运动控制的模型 [18, 64, 68, 75, 79]。

表 1. CREATE 可支持的代表性具身智能体系统

具身系统	规划器		控制器		任务		
	模型	参数量（十亿）	模型	架构	描述	示例	基准测试
SayCan [7]	PaLM [63]	540	MT-Opt [64]	CNN	导航	“移动到桌子旁”	ALFRED [65]
PaLM-E [8]	PaLM-E [8]	562	MT-Opt [64]	CNN	操作	“将水瓶直立放置”	Language-Table [66]
LLM-Planner [9]	GPT-3 [67]	175	HLSM [68]	Transformer	导航	“移至桌子旁”	ALFRED [65]
RT-2 [4]	PaLI-X [69]	55	RT-1 [18]	Transformer	操作	“将水瓶直立放置”	Language-Table [66]
Voyager [10]	GPT-4 [16]	1,760	-	-	多任务	“获取一把铁剑”	《我的世界》 [70]
OpenVLA [26]	LLaMA [71]	7	-	-	多任务	“把瓶子放在柜子上”	LIBERO [72]
RoboFlamingo [27]	OpenFlamingo [73]	3	-	-	多任务	“拉把手打开抽屉”	CALVIN [74]
Octo [75]	-	-	Octo [75]	Transformer	多任务	“给我拿 3 种不同的苏打水”	OXE [76]
LEO [77]	Vicuna [78]	7	ScanRefer [79]	Transformer	导航	“找厕所”	HM3D [80]
JARVIS-1 [11]	LLaVA [71]	8	STEVE-1 [81]	Transformer	多任务	“获得一把铁剑”	《我的世界》 [70]

尽管存在这些差异，视觉-语言-动作模型和运动控制模型在功能上分别类似于规划器和控制器，因此遵循相同的总体框架。此类系统的能力通常通过标准化的仿真基准测试集进行评估，例如[65, 66, 70, 72, 74, 76, 80]。

如表 1所示，现代具身智能体通常采用基于大语言模型的规划器与基于 Transformer 的控制器，以获得更强的多任务学习能力[4, 18, 20, 24, 68, 81]。然而，许多此类系统 [4, 7–10]依赖运行于云服务器上的大型（> 50 B 参数）规划器，使错误注入研究的成本高得令人望而却步，或难以开展。此外，我们认为，目标任务应足够复杂且多样，不能仅限于简单的抓取或导航，才能全面评估智能体的性能。因此，本文选取 JARVIS-1[11]——《我的世界》中的代表性开放世界智能体 [70]，它满足上述所有要求，作为韧性表征的主要测试平台。我们还在多个具身人工智能系统 [18, 26, 27, 75] 上评估我们的方法，以验证其广泛适用性。

Fig3 展示了 JARVIS-1 平台，其中由堆叠的 Transformer 层和各种网络组件组成的K、Down等大语言模型规划器和控制器，在《我的世界》场景中协同工作。智能体需要与动态的三维环境交互以获取指定物品，这通常需要 5~20 基本子任务，每个子任务都涉及数百个动作步骤。例如，要获得一枚附魔金苹果，规划器会将其分解为多个子任务（例如挖掘原木、制作石镐、熔炼金锭），再由控制器按顺序执行。有兴趣的读者可参阅 [11]，了解有关 JARVIS-1 平台和《我的世界》任务的详细描述。根据 [11],，如果某个子任务在 600 个动作步骤后仍未完成，规划器将重新调用并调整该子任务。总步数超过 12,000 后，高层任务即判定为失败。

本文评估了多个《我的世界》任务的任务质量 [10, 11],，这些任务涵盖基础目标（例如木镐、石镐等）以及复杂

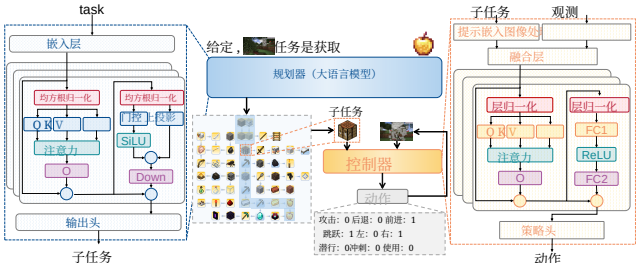

图 3. JARVIS-1 平台。JARVIS-1 能够在《我的世界》沙盒环境中稳定获取高级物品。规划器会生成一系列子任务，而控制器则结合子任务提示和视觉观测，确定每一步的动作。规划器和控制器主要由堆叠的 Transformer 模块构成，其中包含各种网络组件，例如 K 和 Down。

目标（例如铁剑、熟鸡肉等），每项任务都具有不同的轨迹和生物群系。本文将这些任务简称为单个电传打字机字体词（例如 wooden）。为减少执行过程中的随机性，每项试验至少重复 100 次。任务质量通过成功率（↑）和平均步数（↓）衡量，而系统效率则通过平均功耗和总能耗评估。

2.2 具身人工智能系统的计算平台

在具身人工智能系统中，规划器和控制器通常都采用深度神经网络实现，其核心操作涉及密集的通用矩阵乘法（GEMM）。为满足严格的能量要求和实时延迟约束，这些深度神经网络通常部署在本地 GPU 上[8, 17, 18]，或部署在专用边缘加速器上，例如 TPU 式脉动阵列 [4, 7, 13, 20–22]。在许多情况下，通用矩阵乘法会被量化为低精度格式（例如 INT8） [82–88]，以降低功耗，尤其是在必须持续与动态环境交互的电池供电系统中。在本文中，我们考虑将整个具身人工智能系统以 INT8 格式部署在基于脉动阵列的加速器上，这是一种标准化且普遍采用的配置[21, 40, 42, 58, 89, 90]。规划器和控制器以协调的流水线方式执行。

2.3 容错能力研究

错误来源。 随着 CMOS技术 向纳米尺度发展，具身人工智能系统中的存储器和逻辑电路都容易出现可靠性问题。降低工作电压会加剧这一问题：在片上SRAM中，降低的电压噪声容限会导致存储器位翻转 [53, 57, 58, 91],，而在逻辑电路中，电压欠缩放会延长路径延迟，从而导致时序违例和错误的输出 [35, 40, 47, 89]。鉴于在计算密集型工作负载中，逻辑电路通常占据片上功耗的主要部分 [22, 92, 93] ，并且具身人工智能工作负载的动态特性为计算优化创造了重要机会，本文聚焦于由电压欠缩放引起的推理过程中的瞬态计算错误。将基于时序的韧性扩展到存储器组件仍是未来研究中很有前景的方向。

模型韧性研究。 一般而言，错误韧性是指算法容忍错误的能力，通常通过比特错误率与模型性能之间的关系来量化 [58, 60]。许多研究 [89, 90, 94–97]采用随机位翻转模型，作为表征深度神经网络和自主系统固有韧性的抽象。例如， [57] 分析了深度神经网络中的错误传播，而 [58] 探讨了错误影响与数据类型之间的关系。[59]发现网络组件存在异构脆弱性，并提出了选择性保护。 [53, 60]将韧性分析扩展到基于强化学习的应用和自主系统。 [61, 62] 表征了各个大语言模型的韧性， [56] 评估了机器人运动规划中的内存错误风险。然而，由于模型的多样性及其协作方式，以及所执行任务的复杂性，具身智能体系统的韧性行为可能呈现出独特特征。

错误缓解技术。纠错码（ECC）[98] 能够有效应对存储器或片上缓冲区中的硬件故障。缓解计算数据通路中的瞬态故障更具挑战性 [47, 60, 96, 97],，可采用多个层级的保护措施。尽管这些方法在特定场景下有效，但会带来难以承受的硬件开销或运行时开销，在具身智能体场景中尤其如此。例如，在**电路级**，基于冗余的方法（如双模冗余（DMR） [37–39],）广泛应用于自动驾驶车辆中的错误检测，但需要执行多份计算。时序借用方法 [43–45] 在传统电路中集成影子触发器（FF），以检测时序错误，但难以扩展至大规模加速器。尽管有些方法面向加速器，但它们要么需要预先了解硬件规格 [41, 42]，要么会引入显著的硬件开销 [40],，从而限制了它们在在线时序错误缓解中的适用性。在**算法层面**，故障感知再训练虽然对卷积神经网络 [50–52] 和简单强化学习算法 [53, 60], 有效，但

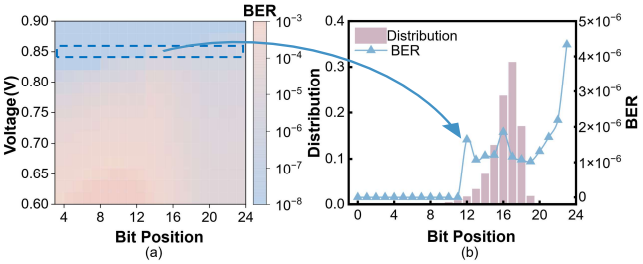

图4. 时序错误模型。（a）不同电压下的位级时序错误率。（b）0.85V下的错误模式，与正常运行时激活分布重叠。

由于重新训练成本难以接受，这种方法不适用于具身智能体。轻量级错误检测与纠正算法，例如基于算法的容错（ABFT）及其变体 [46–49], 适用于 GEMM 运算，但通常需要重新计算以恢复错误，可能违反实时约束。在**系统层面**，面向机器人学或自主系统的应用专用保护策略 [54–56] 是针对基于规则的数据流量身定制的，几乎无法无缝迁移至深度神经网络驱动的具身智能体；后者可能具有截然不同的韧性特征。

3 时序错误建模与注入

3.1 错误模型

由于纠错码可以有效缓解内存故障[59, 60, 96, 97, 99], ，我们专注于推理期间的瞬态计算错误，并假设数据已从内存中正确读取。为对时序错误建模，我们采用广泛使用的随机位翻转模型 [89, 94–96]作为抽象，并以误码率参数化错误严重程度。为捕捉不同电压下的错误模式，我们使用商用22纳米工艺设计套件，综合设计了一个8位乘法器/24位累加器脉动阵列（标称电压：0.9V），并使用 Synopsys PrimeTime 和 HSPICE 进行详细的时序分析。错误模式如图4(a)所示，与先前的芯片测量结果一致 [43, 53, 62, 89]。如图4所示，**高位会频繁出现较大的时序错误**，导致严重的性能下降；而**低位错误则出现在正常运行时数据范围内**。这与以下直觉相符：高位通常具有更长的进位链和更长的路径延迟 [35, 89]。

3.2 错误注入方法

我们在 Py- Torch 中实现了一个动态错误注入框架，将位翻转模拟为运行时张量运算，这也是以往研究中常用的方法 [89, 94–97]。按照 [82],规划器和控制器中的通用矩阵乘法及卷积层输入均量化为 INT8，并在其输出中注入错误。为进行韧性表征（第4节），我们在不同误码率下采用均匀错误模型，以将算法韧性与特定硬件的依赖性分离，从而得出更具普适性的结论。在第6节中，我们根据图4(a)建立查找表，以更准确地捕捉电压诱发的时序错误。

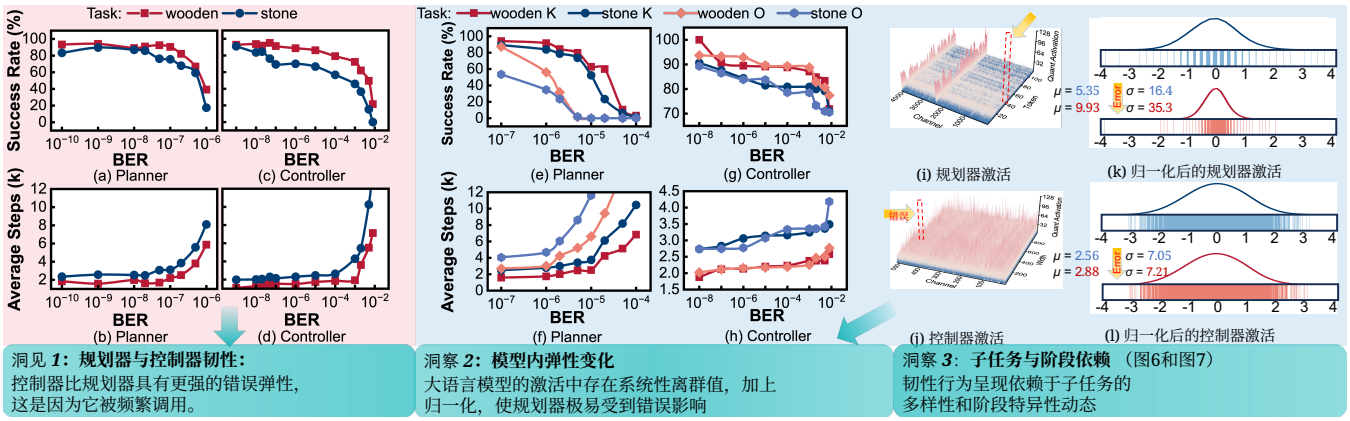

图 5. 韧性表征。(a)-(b) 规划器韧性特征。(c)-(d) 控制器韧性特征。(e)-(f) 规划器中 K 与 O 的韧性比较。(g)-(h) 控制器中 K 与 O 的韧性比较。(i)-(j) 规划器和控制器中预归一化层的激活分布。(k) 规划器的归一化结果在错误下表现出明显偏斜；(l) 控制器的归一化保持适中的方差。

4 韧性表征

本节将介绍我们对具身人工智能系统的韧性表征，并探讨以下问题：

模型级 Q1：规划器和控制器的韧性特征是什么？它们有何不同？

模型级 Q2：此外，在规划器和控制器内部，不同网络组件的韧性有何差异？

应用级 Q3：韧性表现如何随着子任务和执行步骤动态变化？

4.1 模型级韧性表征

规划器的韧性特征。我们首先向规划器注入错误，以评估其韧性。图5(a) 和 (b) 显示，对于这两项任务，成功率在比特错误率 $= 2 \times 10^{-8}$ 附近骤降，而平均步数则显著增加，表明其容错能力有限。造成这一现象的因素有两个：(i) 规划器只调用一次，却要指导智能体执行多个步骤，因此一条误导性指令就可能导致智能体长时间执行无关或错误的动作；(ii) 作为大语言模型，规划器的输出质量在比特错误率较高时会严重下降，生成无关或毫无意义的文本，从而妨碍控制器。后文将进一步分析。

控制器的韧性特征。 同样，我们通过跟踪图中不同比特错误率下的成功率和平均步数，评估控制器的韧性。 5(c)-(d)。在比特错误率达到 $= 1 \times 10^{-4}$ 附近时，成功率显著下降，平均步数相应增加，这凸显了控制器对任务性能的关键影响，同时也揭示了其显著的容错能力。这种现象也可以从两个方面解释：(i) 控制器频繁调用，将错误影响限制在单个步骤内，并允许在后续决策中进行纠正；(ii) 控制器能够容忍次优决策，因为每个步骤可能都有多个可行操作。

讨论。 比较图5 (a)-(b) 与 (c)-(d) 后，我们得出结论： **控制器的错误弹性显著高于规划器，这归功于其更频繁的调用。**

不同网络组件的韧性。本节通过向 Transformer 层中的单个网络组件（例如Q 和FC1）注入错误，探究规划器和控制器的内部韧性。在大语言模型规划器中，经过归一化处理的组件（包括O 和Down）的韧性明显低于未经归一化处理的对应组件（例如K），如图5(e)-(f) 所示。相比之下，控制器在不同组件间仅表现出轻微差异（图5(g)-(h)）。

这种差异源于不同的激活分布。据大量研究报道，参数量超过数十亿的大语言模型会产生幅值大得多的系统性离群值[82, 85, 86] （图 5 （i）），而控制器的激活值则更加均匀（图 5 （j））。因此，在规划器中，归一化参数，即 μ 和 σ，会受到这些离群值的影响；而在控制器中，这些参数在各元素之间的确定则更加均衡。当发生单个硬件故障并在大语言模型计算中引入较大的错误时，这些参数可能会发生剧烈偏移（例如， μ 在图 5 （k）中从 5.36 变为 9.93），从而显著扭曲归一化结果。相比之下，控制器中的偏移相对温和（例如， σ 在图 5 （l）中从 7.05 变为 7.21），因此对结果的影响有限。

讨论。我们得出结论：**大语言模型激活值中的离群值，加之后续的归一化，使规划器极易受到错误的影响。**

4.2 应用级韧性表征具身智能体通常通过连续执行一系列子任务来完成高层目标，整个过程往往长达数百个步骤。本小节考察控制器在多种子任务和执行步骤中的韧性。

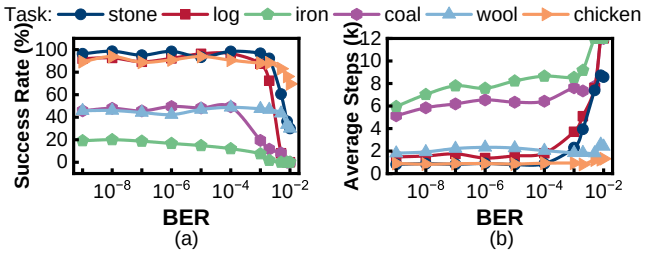

图 6. 不同子任务展现出不同的韧性。

子任务间的韧性差异。 我们通过向图中执行不同子任务的控制器注入错误，评估其在各子任务中的韧性。 6。结果表明，不同子任务的表现各异：子任务 原木 和 石头 在比特错误率超过 $= 10^{-4}$ 后，性能会骤然下降；而 鸡肉 和 羊毛 的性能则逐渐下降。我们认为，这与任务结构有关：顺序子任务（例如砍树）具有确定性的动作依赖关系，一个错误就可能打乱整个工作流程。相比之下，随机性任务（例如动物交互）本身能够容忍变化，从而减轻错误的影响。因此，错误弹性与子任务的特定要求及环境随机性相关。

执行步骤中的韧性动态。 具身智能体通常会在子任务中于关键阶段和非关键阶段之间切换。为探究错误的影响如何随执行步骤而变化，我们以子任务挖掘原木为案例研究，并在不同步骤注入位错误，如图 7所示。该子任务大致可分为两个阶段：探索（寻找树木）和执行（砍伐树木）。在探索阶段，如图7（a）所示，动作受到的约束较少，错误主要导致局部移动偏差，不会造成灾难性故障。然而，执行阶段（图 7（b））要求动作精准、按序且彼此对齐（例如敲击特定的树木方块）。此时，错误可能破坏动作间的依赖关系，引发不连贯动作（例如敲击位置偏移），最终导致任务无法完成。这表明错误严重程度因阶段而异，并随任务关键性的提高而增加。

讨论。 **韧性表现出子任务依赖的多样性和特定阶段的动态特性。**

5 CREATE框架

基于第3.1节和4节的洞见，我们提出 CREATE，这是一项协同优化设计原则，涵盖电路级、模型级和应用级技术，旨在打造高效且可靠的具身人工智能系统。首先，电路级 异常检测与清除(AD) 可减轻时序违例导致的大误差的影响。接下来，权重旋转增强规划(WR) 可增强大语言模型对剩余
检测并清除小但仍然关键的错误。最后，自主性感知电压缩放 （VS）机制会监控智能体的执行状态，并在运行时动态调整控制器的电压，以进一步节省功耗。两者结合，

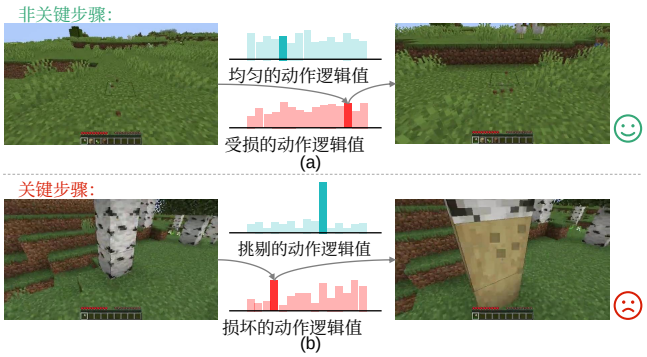

图 7. 阶段特定的弹性。(a) 非关键步骤表现出均匀的动作逻辑值，能够在不发生灾难性故障的情况下容忍损坏。(b) 关键步骤表现出挑剔的动作逻辑值，损坏可能导致故障（例如，未能砍倒树木）。

通过对规划器应用 AD+WR，并对控制器应用 AD+VS，CREATE 实现了紧密的垂直集成，带来协同效益，使其能够大幅降低电压，而无需调整工作频率。

5.1 异常检测与清除

如第 3.1 节所述，降低电压裕度通常会引发时序错误，这些错误主要会翻转更多高有效位，从而导致较大的偏差和性能下降。为缓解这一问题，我们提出了一种异常检测与清除（AD）机制，用于识别越界值并将其钳制为零。

在实际应用中，加速器上的 GEMM 运算通常采用较低精度（例如 INT8）格式，并根据离线确定的缩放因子重新量化结果 [21, 22, 82]。通过分析整个 JARVIS-1 推理流水线中的运行时 GEMM 输出（图8(a)），我们观察到两个特性：(1) 输出值很少占用高有效位（如图4(b)所示）；(2) 大多数元素都接近于零。因此，如果任何结果超出已知的有效范围（例如 INT8 中输出缩放因子的 127 倍），就会将其标记为异常并钳制为零。尽管这无法修正错误结果，却能显著减轻大幅值错误，将剩余偏差留给深度神经网络固有的容错能力来处理。我们强调，在许多深度神经网络的推理过程中，上述两个特性都有充分的文献记录 [57, 60, 84, 86]。因此，我们的策略并不局限于 JARVIS-1，而是可以推广到基于类似深度神经网络架构构建的各种具身人工智能系统。

如图所示，我们通过对脉动阵列进行轻量级改造来实现异常检测与清除（AD）。该阵列支持具身人工智能系统中的通用矩阵乘法（GEMM）运算。 8(b)。在通用矩阵乘法计算过程中，权重存储在各个处理单元（PE）中，而输入则从左向右水平传输。部分和沿列累加，并在向下垂直传递的过程中逐步汇总，最终结果在

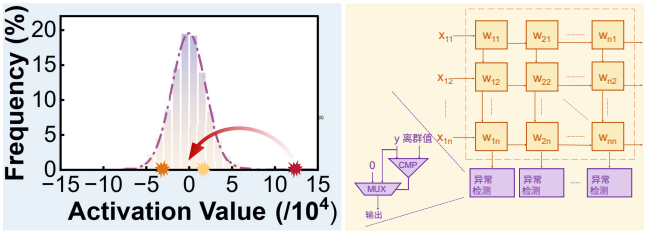

图 8. 异常检测与清除。(a) 通用矩阵乘法运行时结果分布。(b) 电路设计。

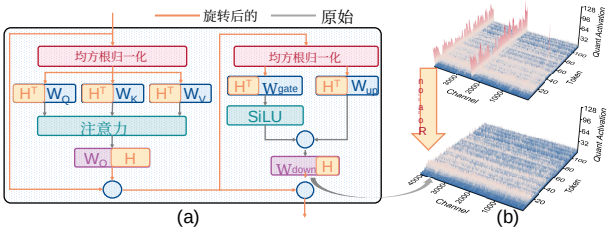

图 9. 权重旋转增强型规划。 (a) 离线将权重与哈达玛矩阵相乘。(b) 旋转前后的激活分布。

底部生成。我们在输出阶段增加一行异常检测单元。每个单元由一个比较器和一个多路复用器组成，并将最终结果与有效范围进行比较。任何超出范围的异常值都会被置为零，而范围内的值则保持不变。这种后处理检测与硬件规格无关，且开销极小，从而实现可移植、轻量级的实时错误抑制。

5.2 权重旋转增强规划

如第 4.1 节所述，规划器大语言模型中的预归一化组件极易受错误影响。第 5.1 节中的异常检测与清除虽然能够抑制超出异常界限的时序错误，但无法处理正常范围内较小、仍会降低任务质量的错误。为此，我们提出权重旋转增强规划（WR），这是一种模型级方法，通过离线权重变换重新分布激活异常值。

受量化技术 [85–88]，启发，我们通过与旋转矩阵相乘，将激活值旋转为不含离群值的分布。一个简单而有效的选择是标准哈达玛矩阵$\mathbf{H}$ [100]，，其通过克罗内克积递归定义（⊗） [101]:

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \quad \text{and} \quad \mathbf{H}_{2^k} = \mathbf{H}_2 \otimes \mathbf{H}_{2^{k-1}}$$

这种旋转将离群值分散到各个维度，从而使激活值分布更加均匀。关键的是，由于旋转矩阵能够保持 L2 范数（作为均方根归一化的分母），因此可以无缝融入大语言模型计算，而不会改变网络的整体输出。

我们的方法仅专注于在归一化之前重新分配激活值，并避免像 [85–88] 中那样进行在线旋转。图 9(a) 描绘了所需的旋转，这些旋转可以离线合并到权重（O 和 Down）中。例如，考虑输入为 X、权重为 W_O 的 O，可得 Y = XW_O。将 W_O 右乘 H 后，可得到 X(W_OH) =(XW_O)H = YH，从而在图 9(b) 中获得无离群值的激活值。为保持多层感知机（MLP）积木中 SiLU 的功能正确性，W^{gate} 和 W^{up} 左乘 H^T。类似地，为消除 Down 之后的离群值，W_{down} 右乘 H，而下一层中的 W_Q、W_K 和 W_V 左乘 H^T。这些离线过程不会引入额外的运行时开销，也不需要特定模型微调。此外，旋转后的激活值可以与异常检测与清除（AD）协同，通过进一步收紧异常界限，实现更有效的错误抑制。

5.3 自主自适应电压调节

To leverage the substantial variation in embodied behavior (Sec. 4.2), we propose an Autonomous Voltage Scaling (VS) mechanism that dynamically adapts to the workload. We first identify the entropy of action logits [102] as the primary runtime state indicator and train a lightweight network to predict it ahead of execution. We then introduce an LDO-based architecture that enables fine-grained voltage control with minimal overhead, providing a practical runtime voltage scaling solution for learning-based embodied controllers.

动作逻辑值的熵作为指标。 具身智能体通常面临动态环境，其中每个步骤的重要性都可能有显著差异。正如第 4.2 节所述，韧性行为因子任务和动态执行状态而异，因此人工设计或固定策略并不适用。

为了设计有效的电压控制策略，需要一种运行时指标来量化步骤关键性。观察图 $z \in \mathbb{R}^n$ 中的动作逻辑值（记为 $z \in \mathbb{R}^n$）7 可发现其分布随阶段而异：在非关键阶段，逻辑值接近均匀分布，表明置信度较低；而在关键阶段，逻辑值则高度集中，反映出对特定动作的高置信度。因此，我们使用熵 [102], 定义$_n$为 $H(z)$ $= -\sum z_i$对数(z_i)，以量化这种置信度并$_{i=1}$表示步骤关键性。图 10 展示了任务各步骤的熵示例，它能精确反映智能体的状态。较低的熵对应需要较大电压裕度的关键阶段，而较高的熵则表明这些非关键阶段可以采用较低电压以提高效率。该方法能够动态协调算法确定性与系统能耗成本。

熵预测器。 为实现逐步粒度电压调节，我们引入了一种执行前熵预测器。熵预测器接收子任务提示嵌入和观测图像作为输入，估计控制器在不受任何先前错误影响、不会产生失真的情况下的熵。如图 11（a）所示，预测器使用卷积神经网络进行图像处理，

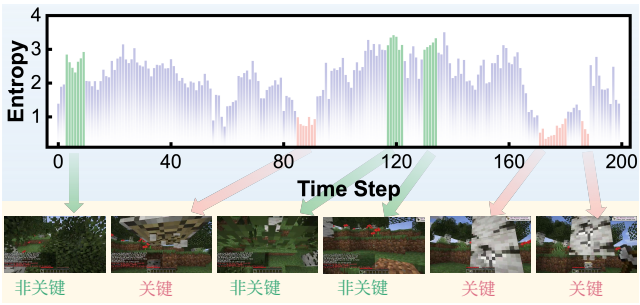

图 10. 不同时间步的熵曲线。较高的熵对应非关键执行步骤，而较低的熵则表明这些步骤至关重要。

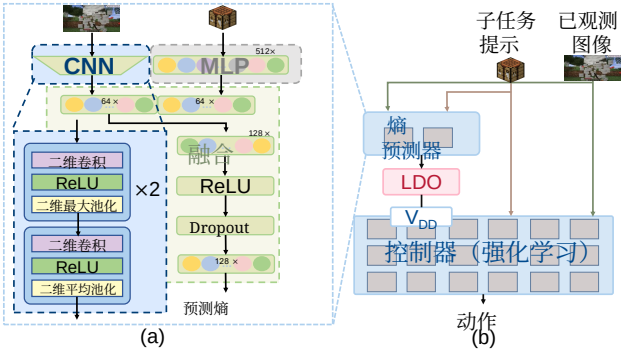

图 11. 自主性感知电压缩放。(a) 熵预测器网络架构。(b) 电压缩放系统：低压差稳压器根据实时熵预测动态调整控制器模型的供电电压。

使用多层感知机处理提示嵌入，并通过融合多层感知机输出单个标量熵估计值。模型详情见附录 A。我们构建了一个包含 25 万多帧的数据集，涵盖多种任务；每帧均包含一个提示嵌入、一幅观测图像，以及根据控制器无错误执行得出的真实熵值。预测器通过预测熵与真实熵之间的均方误差（MSE）损失进行训练。

电压缩放系统。 如图所示，我们的电压缩放系统会根据预测的熵自适应调整控制器模型的供电电压11(b)。控制器模型和预测器均运行在处理单元阵列上，而预测器以标称电压运行，以确保熵预测无误。该预测结果会传递给数字低压差稳压器[93],，由其调整控制器的工作电压。熵越高，触发的电压越低，具体的熵到电压映射关系见第6.5节。控制器的电压每 5 步更新一次——这一频率足以跟踪工作负载变化，同时更新间隔也足够长，可将切换开销降至最低（第6.5节）。

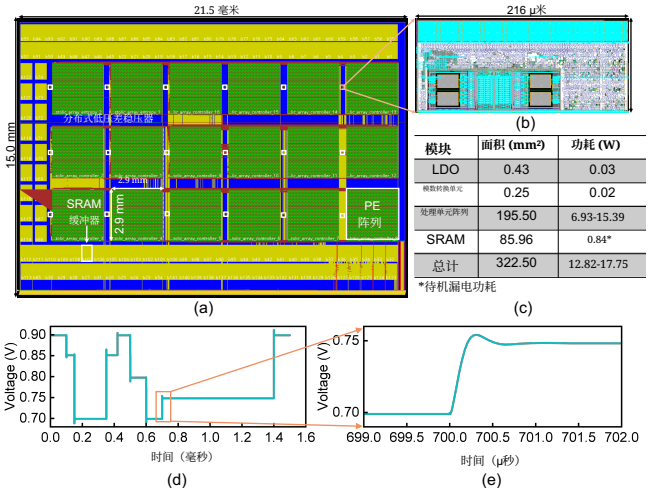

图 12. 硬件平台。(a) 完整加速器的布局布线后版图。(b) 低压差稳压器的详细版图。(c) 面积和功耗分解。(d)(e) 电压缩放波形示例。

6 实验

本节首先介绍实验设置（第6.1节）和电路级指标（第6.2节）。随后，我们分别分析 CREATE 的三项核心技术的有效性（第6.3–6.5节），并通过消融研究探讨这些技术之间的相互作用（第6.6节）。我们进一步展示 CREATE 的整体优势，验证其在各类任务和具身 AI 平台上的通用性（第6.7节），并分析其在现实世界中的影响（第6.8节）。最后，我们讨论其对若干配置的敏感性（第6.9节），并将所提出的方法与现有方法进行比较（第6.10节）。

6.1 实验设置

硬件平台。如图所示，我们将 CREATE 技术集成到一个统一的硬件系统中，专为端到端具身人工智能部署而设计。12。第5.1 节中的异常检测单元被添加到脉动阵列中，阵列包含 128 × 128个处理单元，每个处理单元由一个8位乘法器和一个24位累加器组成。第5.3节所需的动态电压缩放系统由分布式低压差稳压器设计提供支持。我们的设计采用商用22纳米工艺设计套件进行综合，标称供电电压为0.9 V，时钟周期为2 ns。低压差稳压器基于 [103] 构建，可按 10 mV的步进将处理单元阵列电压动态缩放至0.6 V至0.9 V。该低压差稳压器的峰值电流效率达到99.8%，瞬态响应时间为90 ns/50 mV，如表2所示。片上静态随机存取存储器缓冲区（142 × 512 KB，总计71 MB）布置在处理单元阵列周围，高带宽内存2 [104] 用于片外存储。推理过程中，权重和输入最初从片外HBM2加载，并缓存在片上静态随机存取存储器缓冲区中以供重复使用，激活值也存储在片上。对于控制器，鉴于模型规模相对较小，我们的静态随机存取存储器容量足以容纳所有权重，从而避免每次调用时重复访问片外存储器。图12(a)和(b)展示了

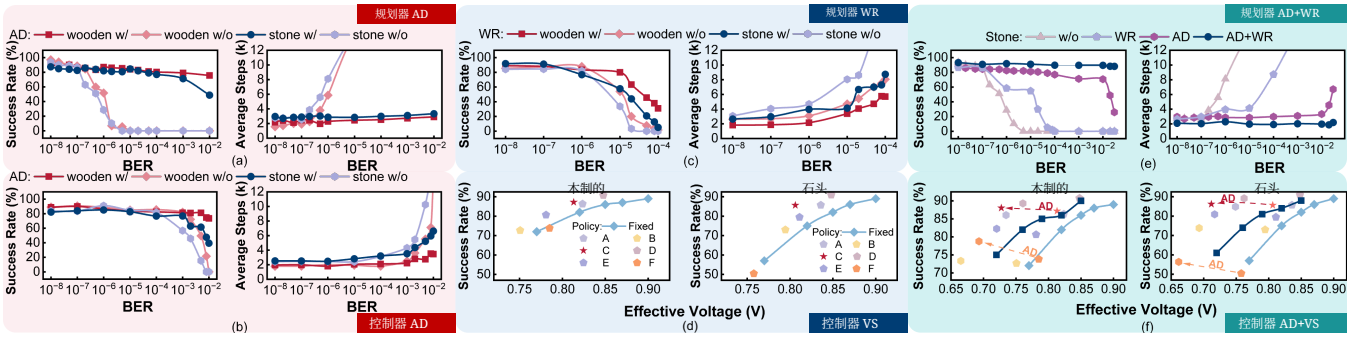

图13. CREATE 技术评估。(a) 规划器的 AD 评估。(b) 控制器的 AD 评估。(c) 规划器的 WR 评估。(d) 控制器的 VS 评估。(e) 规划器的消融研究（AD+WR）。(f)控制器的消融研究（AD+VS）。

表2。低压差稳压器的性能规格。

工艺技术	22 nm	Area	0.43 mm^2
V_{out}	0.6–0.9 V	$I_{load,max}$	15.2 A
t_{resp}	90 ns/50 mV	η_{peak}	99.8% @$I_{load,\ max}$
V_{step}	10 mV	J	35 A/mm^2

t_{resp}：响应时间，η_{peak}：峰值电流效率，J：电流密度。

表3。 完整加速器的性能。

峰值性能 144 每秒万亿次操作		切换延迟	540 纳秒
规划器乘加运算次数	5.3 T	规划器延迟	11.2 毫秒
控制器乘加运算次数	102 G	控制器延迟	942 μ秒
预测器乘加运算次数	43 M	预测器延迟	8.57 μ秒

每秒万亿次操作：每秒万亿次操作。乘加运算：乘法累加。

完整系统和单个低压差稳压器的布局布线后版图，以及图中示例性的电压缩放波形。12(d)和(e)。图12(c)中主要模块的面积和功耗数据基于布局后仿真得出。包括推理延迟和内存访问在内的周期级行为，基于 SCALE-Sim 建模 [105]。

熵预测器训练。 我们通过将所提出的熵预测器集成到实时任务回合中，对其进行评估。在我们的系统中，动作逻辑值的理论最大熵为 13.07，大多数值都小于 4。关于训练细节，我们使用 AdamW优化器 [106]，并以预测熵值与真实熵值之间的均方误差损失进行训练，权重衰减为 1×10^{-2}。模型训练 200 个周期，批量大小为 128，初始学习率为 1×10^{-4}。

评估任务。我们首先使用基础的木制的 和 石头 任务作为第 6.36.3-6.6节中的主要基准，展示其性能和消融结果，以便快速评估。每项实验重复 100 次，以减少随机性。为验证通用性，我们进一步在 JARVIS-1 的六个未见过且更具挑战性的基准上测试了该框架，这些基准具有不同的任务轨迹，从而证明其鲁棒性和适应性。为评估跨平台通用性，我们将方法扩展到独立的平台 [18, 26, 27, 75]，分别用于规划器和控制器。对于规划器，我们将实现移植到 LIBERO 上的 OpenVLA [26]上 [72] 以及 CALVIN [27] 上的 RoboFlamingo [74]。对于控制器，我们将方法与 Octo [75] 和 RT-1 [18]集成不同 OXE [76] 任务。时序错误通过随机翻转累积结果中的比特来模拟，随后[35, 89, 94–96]。比特错误率与工作电压之间的关系如图4(a) 所示。

评估指标。我们评估任务性能（成功率、平均步数）和系统效率（平均功耗、总能耗）。成功率表示完成试验的比例，平均步数则表示所有成功试验的平均步数。为简要衡量功耗，我们引入有效电压，其定义为：能够产生与实际变化电压下总能耗相同的恒定电压。计算能耗通过跟踪规划器、控制器和熵预测器执行过程中每一步的工作电压进行模拟。实际上，开关功耗可以忽略不计。计算能耗时，将失败的任务也纳入其中，并假设其已完整执行。

6.2 电路开销比较

图 12(b) 展示了所提出系统的面积和功耗分布。集成在处理单元阵列中的异常检测单元带来 0.08% 的面积开销和 0.10% 的功耗开销，而分布在阵列之间的低压差稳压器则带来 0.13% 的面积开销和 0.14% 的功耗开销——两者均可忽略不计。

如表3所总结，完整系统的推理延迟分别为：规划器 11.2 ms、 942 μs （控制器）和 8.57 μs（熵预测器）。电压切换延迟保持在 540 ns 以下，比控制器的推理延迟低几个数量级。这满足了我们的具身人工智能系统 30 Hz 的实时推理要求 [11]。

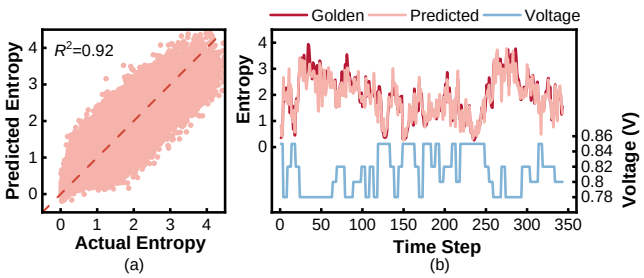

图14. 熵预测器准确度评估。(a) 预测熵与实际熵高度相关，with $R^2 = 0.92$。(b) 实时熵预测及对应的工作电压水平。熵预测器可以有效指导电压缩放。

6.3 异常检测与清除评估

规划器性能比较。为评估异常检测与清除对规划器的有效性，图13(a) 比较了启用和未启用异常检测与清除时的成功率和平均步数。异常检测与清除显著提升了系统抵御时序错误的鲁棒性：当比特错误率为 $=1\times10^{-5}$ 时，木制的成功率从 0% 提升至 85%，而石头的成功率从 0% 提升至 83%；与此同时，平均步数恢复至无错误场景的水平。在电路开销可忽略不计的情况下，异常检测与清除在比特错误率为$< 1\times10^{-5}$时仍能维持相当的性能，从而释放出巨大的潜在能量节省空间。

控制器性能比较。图13(b)评估了异常检测与清除对控制器的影响。异常检测与清除显著增强了控制器的鲁棒性：对于木制的和石头，成功率分别提高了73和48个百分点；在比特错误率$= 5 \times 10^{-3}$下，平均步数分别减少了51%和49%。异常检测与清除在比特错误率$> 1\times10^{-4}$下尤其显著地提升了控制器性能，展现了其在激进电压缩放条件下缓解时序错误的效果。

6.4 权重旋转增强规划评估

图 13(c)评估了 WR 对规划器可靠性的影响。通过重新分配大语言模型规划器中的激活分布，该方法可防止单个错误严重扭曲归一化结果。WR 在 木制的任务中将成功率提高了 43%，在 石头任务中提高了 40%；当比特错误率为 $= 2\times10^{-5}$时，平均步数分别减少了 33% 和 49%。这凸显了 WR 在时序错误下增强规划器鲁棒性的效果。

6.5 自主自适应电压调节评估

熵预测器的准确性与分析。我们的训练过程最终收敛至均方误差 9.96×10^{-2}。图 14(a) 展示了预测熵与真实熵之间的强相关性（$R^2 = 0.92$）。图 14(b) 展示了运行时性能：在各个时间步中，预测熵都能紧密跟踪实际的熵动态变化，从而确保其适用于实时电压缩放。

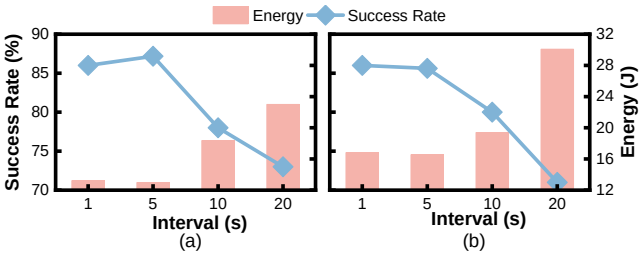

图 15. 电压更新间隔对（a）木制的任务和（b）石头任务的能耗与成功率的影响。

性能提升。为展示电压调节的影响，我们将本方法与恒定电压基线进行比较，该基线在整个执行过程中保持固定电压。任务性能通过在 木制的 和 石头 基准测试集上的成功率衡量，能量效率则通过有效电压量化，如图 13(d) 所示。该图包含六种熵到电压映射策略 A–F（图 21），它们从 100 个候选方案中搜索得出，每种策略都能更好地平衡性能与效率。与恒定电压策略（蓝线）相比，后者在电压较低时成功率会下降，而我们的自适应策略始终能实现更高的能量效率和任务性能。其中，策略C（红色星号）在不牺牲成功率的情况下将有效电压降低了 7.3%。因此，我们选择该策略作为最优策略，以推进可靠性—效率帕累托前沿，并将其用作后续实验的默认配置。

电压更新间隔分析。 本段评估了电压更新间隔的影响，即电压调整之间的步数。如图 15所示，间隔为 1 步和 5 步时，两项任务的成功率都保持较高；而间隔为 10 步和 20 步时，性能会下降，表明实时响应不足。能耗分析还显示，5 步间隔的开销略低于 1 步间隔。因此，5 步间隔在及时适应与 VS效率之间实现了最佳平衡，故选为默认电压更新间隔。

6.6 消融研究

我们通过消融研究评估框架中的各项技术。由于 AD 和 WR 针对规划器，而 AD 和 VS 优化控制器，因此我们分别测试 AD+WR 和 AD+VS 配置，以单独考察它们的贡献。

规划器消融研究。如图13(e) 所示，虽然异常检测与清除和权重旋转都能独立提升鲁棒性，但两者结合可显著增强韧性，即使在比特错误率$=1 \times 10^{-2}$下也能保持任务质量。这不仅表明异常检测与清除和权重旋转各自有效，也展现了它们潜在的协同效应。我们认为，这种协同效应源于权重旋转能够将离群值分散到不同维度，从而降低异常检测与清除的异常检测阈值，进一步减轻错误影响。

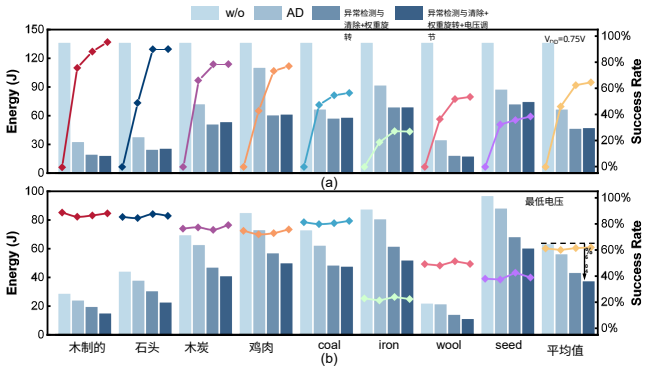

图 16. 不同任务的总体评估。(a) 在 0.75V 下，CREATE 在保持成功率的同时减少执行步骤和能耗。(b) 通过支持激进电压缩放，CREATE 在性能不下降的情况下实现了平均 40.6% 的能量节省。

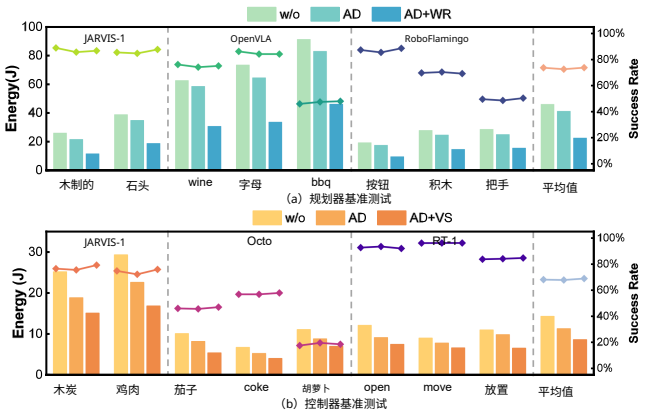

图 17. 跨平台评估。在各种平台和任务上，我们的方法始终能够实现能量节省。AD+WR 平均可将规划器能耗降低 50.7%，而 AD+VS 可为控制器实现平均 39.3% 的功耗节省。

控制器消融研究。图13(f) 对比了与异常检测与清除结合的电压调节（深色线条和五边形）和单独的电压调节（浅色线条和六边形）。通过提高低电压下的鲁棒性，异常检测与清除使电压调节能够在更低的电压水平下运行，从而提升能量效率。数据点向左移动直观展现了这一效果，虚线箭头表示有效电压的降低。与异常检测与清除协同工作时，最优策略将有效电压降低了 11.2%（木制的）和 13.2%（石头），相比之下，这一降幅是相对于结合异常检测与清除的恒定电压基线而言；与未结合异常检测与清除的基线相比，则分别降低了 16.4% 和 18.2%，同时保持任务成功率。这些结果凸显了电压调节与异常检测与清除相结合带来的另一项协同优势。

6.7 不同任务的总体评估 sks

本节评估整个框架的整体优势。为进一步评估其通用性，我们将实验扩展到 JARVIS-1 中另外六种更复杂的工作负载，包括木炭、鸡肉、煤、铁、羊毛和种子任务。我们首先报告 0.75V 下的成功率和计算能耗，以展示低电压运行时的可靠性提升。随后，我们量化在维持可接受性能的最低电压下运行所带来的能量节省，从而评估效率提升。

可靠性提升。 图16（a）比较了四种配置和八项任务下的成功率与总计算能耗：无保护、仅 AD、AD+WR，以及 AD+WR+VS。在 0.75V 电压下运行且不采取保护措施，会导致成功率接近于零。仅使用 AD 可恢复至无错误成功率平均值的 71%，而将 AD 与 WR 结合使用则可将可靠性提升至 97%，接近基准水平。VS 虽然并非为增强容错能力而设计，但只会带来极小的开销。由于任务质量得到提升，平均步数减少，从而显著降低了总能耗。

效率提升。图16（b）比较了各项任务在能够保持成功率的最低电压下的计算能耗。与标称电压情形相比，仅使用 AD 平均可降低 11.1% 的能耗。WR 还能进一步降低规划器电压，实现 18.8% 的计算能耗节省。VS 动态优化控制器电压，额外节省 10.7% 的能耗。这些技术协同作用，在不影响任务成功率的前提下，总计节省 40.6% 的能耗。

跨平台通用性评估。为进一步验证所提方法的通用性，我们将这些技术应用于其他具有代表性的具身人工智能平台。由于缺乏能够集成规划器和控制器的鲁棒端到端机器人平台的开源支持，我们分别评估能量节省情况。对于规划器，我们将异常检测与清除+权重旋转应用于 OpenVLA（葡萄酒、字母、烧烤，运行于 LIBERO）和 RoboFlamingo（按钮、积木、把手，运行于 CALVIN）。对于控制器，我们将异常检测与清除+电压调节应用于 Octo（茄子、可乐、胡萝卜，运行于 OXE）和 RT-1（打开、移动、放置，运行于 OXE）。附录B中提供了详细的任务描述。图17 展示了在保持任务质量的同时所实现的计算能耗节省。在所有平台和任务上，我们的方法都能持续降低能耗。异常检测与清除+权重旋转平均可降低规划器能耗 50.7%，而异常检测与清除+电压调节可为控制器平均节省 39.3% 的能量。

6.8 能耗细分与分析

第6.7节证明，我们的 CREATE 框架能够持续提升各类具身人工智能平台的可靠性和效率。在本小节中，我们分析这些计算能耗节省如何转化为

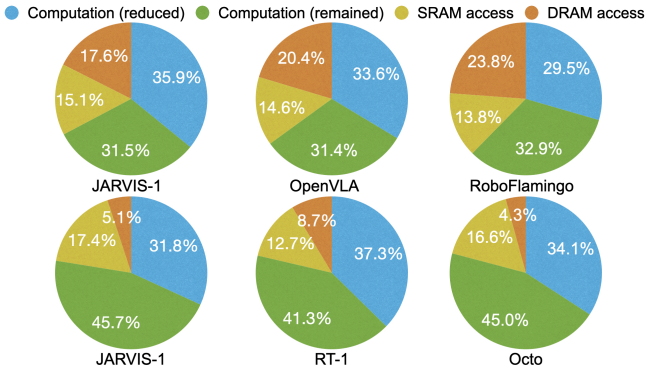

图18。芯片级能耗分解。对于规划器，计算约占总能耗的 65%，AD+WR 可将计算能耗降低约 55%，从而节省约 30% 的总能耗。对于控制器，计算占总能耗的 75% 以上，而 AD+VS 可将计算能耗降低约 45%，从而节省约 35% 的总能耗。

整个机器人系统芯片级能耗的整体降低和电池续航时间的延长。

芯片级能耗大致可分为两个部分：计算和内存访问，后者包括片上 SRAM 和片外 DRAM 访问。表4 汇总了本研究评估的所有模型的参数总数和计算量。能耗估算结合 SCALE-Sim 仿真结果、布局后综合报告和 HBM2 规格得出 [104]。规划器权重在每次推理时都需要从片外加载，而控制器权重则可以完全存储在片上 SRAM 中，从而避免执行期间反复进行片外访问。

表4. 模型参数与计算需求

模型	#参数 (M)	输入	输出	GOps[3]
JARVIS-1规划器 [11]	7,869	740[1]	25[1]	5,344
OpenVLA [26]	6,929	617[1]	7[1]	4,595
RoboFlamingo [27]	2,552	505[1]	6[1]	2,411
JARVIS-1 控制器 [81]	61	128[2]	-	102
RT-1 [18]	35	224[2]	-	78
Octo [75]	27	224[2]	-	76
熵预测器	0.055	-	-	0.043

[1]规划器的输入和输出大小分别对应具有代表性的预填充和解码词元数量，这些数量可能因执行而异。[2]对于控制器，输入大小指方形 RGB图像的分辨率。[3]INT8 下的十亿次运算。

图 18 展示了各规划器和控制器的能耗分解。计算分别占规划器总能耗的 67.4%、65.0% 和 62.4%，占控制器总能耗的 77.5%、78.6% 和 79.1%（这是由于 DRAM 开销经过摊销）。因此，第 6.7 节中分析的收益可转化为芯片级

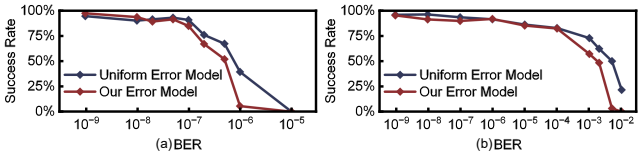

图19. 应用于 JARVIS-1 木制任务中 (a) 规划器和 (b) 控制器的统一错误模型与硬件特定误差模型对比

规划器分别节省 35.9%、33.6% 和 29.5% 的能量，控制器分别节省 31.8%、37.3% 和 34.1% 的能量。

尽管机械部件（如执行器和电机）的能耗可能会抵消部分节能效果，但根据几种常见配置 [13, 26, 30–32], , 在计算能耗与机械部件相当或更高的实际机器人部署场景中，我们估计电池续航时间总体可延长 15–30%。这些收益对任务成功率、计算延迟或芯片面积的影响微乎其微，因此，所提出的方法是面向真实世界具身 AI 部署的一种实用且有意义的优化。此外，降低计算功耗也有助于缓解散热问题。

6.9 配置分析

本小节研究若干手动定义的配置所带来的影响。首先，我们验证第 4 节在均匀错误模型下得出的结论。接着，我们分析实验重复次数对测得的任务性能有何影响。最后，我们探讨采用更激进的量化方式（如 INT4）所带来的影响。

韧性表征的有效性。在第4节中，我们采用均匀错误模型来推导具有普适性的算法见解；而在第 6节中，则使用图 4所示的误差模型，在电压缩放条件下进行更准确的功耗估算。图 19 展示了木制任务的成功率，体现了两种误差模型的影响。尽管数值略有差异，但整体性能趋势保持一致，这验证了韧性表征不依赖于特定的误差模型。

重复次数的统计显著性。在本研究中，我们将实验重复至少100 次，以确保获得 3% 至 5% 的 95% 置信区间。这足以揭示支持我们结论的总体趋势。桌子 5展示了随着重复实验次数增加，木制任务的测得成功率。如图所示，重复 100 次已足以获得收敛结果。

量化相关行为。尽管本研究选择 INT8 作为默认配置，但更激进的量化（例如 INT4）提供了一种正交的方法来降低功耗并延长电池续航时间，因为它不仅降低了计算复杂度，还

表5. 测得的成功率与重复次数。

# 重复次数	20	40	60	80	90	100
成功率	85.0%	90.0%	91.7%	92.5%	91.1%	90.0%
# 重复次数	110	120	140	160	180	200
成功率	89.1%	89.2%	90.7%	90.6%	89.4%	89.5%

Meas在木制任务上测得，向 JARVIS-1 控制器注入比特错误率 $= 1 \times 10^{-7}$ 器。

Tabl图 6. 应用 A 时在石头上的成功率 D+WR

BER	1×10^{-6}	1×10^{-5}	1×10^{-4}	1×10^{-3}	1×10^{-2}
INT8	90.4%	91.5%	91.4%	90.5%	85.6%
INT4	91.0%	88.3%	88.2%	89.5%	87.7%

内存开销。在电路级，将数据量化为 INT4 后，高位中的时序错误仍会频繁发生，因为其关键路径更长。在算法层面，我们在表 +6 中展示了 AD+WR 对 INT8 和 INT4 的影响。目前的结果在统计上没有显著差异。我们推测，这是因为与 INT8 相比，INT4 会因量化误差更大而降低无错误成功率。然而，应用 AD+WR 后，低于检测阈值的未检测错误范围被压缩，从而增强了对注入错误的鲁棒性。

6.10 与现有技术的比较

在本小节中，我们将 CREATE 与现有的电路级和算法级兼容基线方法进行比较。在电路级方面，我们选取了一种具有代表性的时序借用方法 ThUnderVolt [40]，以及广泛使用的双模冗余 [39]；在算法级方面，我们采用了当前最先进的算法容错变体 [49]。图 20展示了不同工作电压下的成功率和总能耗。双模冗余能够实现较高的可靠性，但至少会引入 2× 的开销和频繁的错误恢复，从而导致难以接受的能耗成本。ThUnderVolt 为所有处理单元添加旁路电路，以跳过错误结果，电路开销略低；但由于过度的神经元剪枝，其性能显著下降，并增加了执行步骤，最终导致低供电电压下的功耗增加。算法容错使用轻量级校验和来纠正错误，但由于比特错误率升高时错误恢复开销频繁，其在 0.85V 以下的应用受到限制。相比之下，CREATE 引入的电路开销可忽略不计，避免了恢复机制，同时保持任务质量，分别为木制的和石头实现了 35.0% 和 33.8% 的能量节省。

7 结论

我们提出 CREATE，这是一项通用设计原则，利用电路、算法和应用层之间的异构韧性，协同实现高效且可靠的具身人工智能系统。我们首次开展了大规模错误注入实验，以表征具身人工智能系统的韧性。实验结果表明，

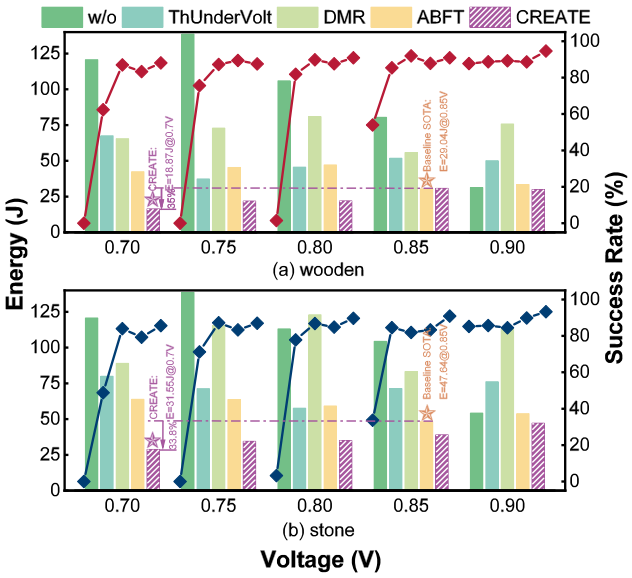

图20。现有技术比较。CRE-ATE 在各种工作电压下始终优于所有基线方法；与 SOTA 基线相比，对于木制的物体，能量消耗降低了 (a) 35.0%，对于石头物体降低了 (b) 33.8%。

由于激活分布中存在系统性离群值，规划器的韧性远逊于控制器；而控制器的韧性则取决于子任务和执行步骤。基于这些发现，CREATE 通过电压缩放架构设计，在电路、模型和应用层面协同优化效率与可靠性。通过大幅降低电压，CREATE 在各种具身人工智能平台和任务中持续实现能量节省。与标称电压运行相比，其计算能耗平均降低 40.6%；与基线最先进方法相比，降低 35.0%，折合芯片级能量节省 29.5% 至 37.3%。通过促进具身人工智能的节能部署，我们的框架可将系统整体电池续航时间延长约 15% 至 30%，拓宽具身人工智能系统的潜在应用范围。

致谢

本研究部分由国家自然科学基金委员会（NSFC）资助，项目编号为62125401、62495102、92464104和62341407；部分由国家重点研发计划资助，项目编号为2024YFB4505004；部分由北京市科技计划资助，项目编号为Z241100004224015；部分由北京市杰出青年科学家计划资助，项目编号为JWZQ20240101004；部分由111计划资助，项目编号为B18001。

参考文献

[1] Y. Liu、W. Chen、Y. Bai、X. Liang、G. Li、W. Gao和L. Lin，《连接网络空间与物理世界：具身人工智能综合综述》，arXiv预印本arXiv:2407.06886，2024年。[2] J. Duan、S. Yu、H. L. Tan、H. Zhu和C. Tan，《具身人工智能综述：从模拟器到研究任务》，IEEE《计算智能新兴主题汇刊》，第6卷，第2期，第230–244页，2022年。[3] Z. Xu、K. Wu、J. Wen、J. Li、N. Liu、Z. Che和J. Tang，《基础模型驱动的机器人学综述：迈向具身人工智能》，arXiv预印本arXiv:2402.02385，2024年。[4] A. Brohan、N. Brown、J. Carbajal、Y. Chebotar、X. Chen、K. Choromanski、T. Ding、D. Driess、A. Dubey、C. Finn等，《RT-2：视觉-语言-动作模型将网络知识迁移至机器人控制》，arXiv预印本arXiv:2307.15818，2023年。[5] C. Gan、S. Zhou、J. Schwartz、S. Alter、A. Bhandwaldar、D. Gutfreund、D. L. Yamins、J. J. DiCarlo、J. McDermott、A. Torralba等，《ThreeDWorld Transport挑战赛：面向物理真实具身人工智能的视觉引导任务与运动规划基准测试》，载于2022年国际机器人与自动化会议（ICRA），第8847–8854页，IEEE，2022年。[6] Q. Zhou、S. Chen、Y. Wang、H. Xu、W. Du、H. Zhang、Y. Du、J. B. Tenenbaum和C. Gan，《Hazard挑战赛：动态变化环境中的具身决策》，arXiv预印本arXiv:2401.12975，2024年。[7] M. Ahn、A. Brohan、N. Brown、Y. Chebotar、O. Cortes、B. David、C. Finn、C. Fu、K. Gopalakrishnan、K. Hausman等，《按我能做的去做，而非按我说的去做：将语言接地于机器人可供性》，arXiv预印本arXiv:2204.01691，2022年。[8] D. Driess、F. Xia、M. S. Sajjadi、C. Lynch、A. Chowdhery、B. Ichter、A. Wahid、J. Tompson、Q. Vuong、T. Yu等，《PaLM-E：具身多模态语言模型》，载于国际机器学习会议，第8469–8488页，PMLR，2023年。[9] C. H. Song、J. Wu、C. Washington、B. M. Sadler、W.-L. Chao和Y. Su，《LLM-Planner：利用大语言模型为具身智能体进行少样本接地规划》，载于IEEE/CVF国际计算机视觉会议论文集，第2998–3009页，2023年。[10] G. Wang、Y. Xie、Y. Jiang、A. Mandlekar、C. Xiao、Y. Zhu、L. Fan和A. Anandkumar，《Voyager：基于大语言模型的开放式具身智能体》，arXiv预印本arXiv:2305.16291，2023年。[11] Z. Wang、S. Cai、A. Liu、Y. Jin、J. Hou、B. Zhang、H. Lin、Z. He、Z. Zheng、Y. Yang等，《JARVIS-1：采用记忆增强多模态语言模型的开放世界多任务智能体》，IEEE《模式分析与机器智能汇刊》，2024年。[12] K. Black、N. Brown、D. Driess、A. Esmail、M. Equi、C. Finn、N. Fusai、L. Groom、K. Hausman、B. Ichter等，《用于通用机器人控制的视觉-语言-动作流模型》，arXiv预印本arXiv:2410.24164，2024年。[13] Y. Huang、Y. Hao、B. Yu、F. Yan、Y. Yang、F. Min、Y. Han、L. Ma、S. Liu、Q. Liu等，《Dadu-Corki：面向具身人工智能驱动的机器人操作的算法-架构协同设计》，载于第52届年度国际计算机体系结构研讨会论文集，第327–343页，2025年。[14] A. Suleiman、Z. Zhang、L. Carlone、S. Karaman和V. Sze，《Navion：用于纳米无人机自主导航的2毫瓦全集成实时视觉惯性里程计加速器》，IEEE《固态电路汇刊》，第54卷，第4期，第1106–1119页，2019年。[15] Y. Gan、Y. Bo、B. Tian、L. Xu、W. Hu、S. Liu、Q. Liu、Y. Zhang、J. Tang和Y. Zhu，《Eudoxus：自主机器中定位的特征分析与加速——产业专题论文》，载于2021年IEEE国际高性能计算机体系结构研讨会（HPCA），第827–840页，IEEE，2021年。[16] J. Achiam、S. Adler、S. Agarwal、L. Ahmad、I. Akkaya、F. L. Aleman、D. Almeida、J. Altenschmidt、S. Altman、S. Anadkat等，《GPT-4技术报告》，arXiv预印本arXiv:2303.08774，2023年。

[17] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen 等，“TinyVLA：面向机器人操作的快速、数据高效视觉-语言-动作模型”，IEEE《机器人与自动化快报》，2025。[18] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu 等，“RT-1：用于大规模现实世界控制的机器人Transformer”，arXiv预印本 arXiv:2212.06817，2022。[19] Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury 和 S. Liu，“基于FPGA的机器人计算综述”，IEEE电路与系统杂志，第21卷，第2期，第48–74页，2021。[20] 香港中文大学，“Airship：人工智能研究与创新中心。” https://airs.cuhk.edu.cn/en/airship，2023。[21] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers 等，“数据中心中张量处理单元的性能分析”，载于第44届年度国际计算机体系结构研讨会论文集，第1–12页，2017。[22] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng 和 J. S. Vetter，“NVIDIA张量核心的可编程性、性能与精度”，载于2018年IEEE国际并行与分布式处理研讨会研讨会（IPDPSW），第522–531页，IEEE，2018。[23] D. McNulty, A. Hennessy, M. Li, E. Armstrong 和 K. M. Ryan，“自主移动机器人用锂离子电池综述：未来展望”，《电源杂志》，第545卷，文章编号231943，2022。[24] Figure AI Inc.，“Helix：推进人工智能机器人技术。” https://www.figure.ai/news/helix，2025。[25] Z. Wan, Y. Du, M. Ibrahim, J. Qian, J. Jabbour, Y. Zhao, T. Krishna, A. Raychowdhury 和 V. J. Reddi，“Reca：面向实时高效协作式具身自主智能体的集成加速”，载于第30届ACM编程语言与操作系统架构支持国际会议论文集，第2卷，第982–997页，2025。[26] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi 等，“OpenVLA：一种开源视觉-语言-动作模型”，arXiv预印本 arXiv:2406 09246，2024。[27] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu 等，“视觉-语言基础模型作为有效的机器人模仿者”，arXiv预印本 arXiv:2311.01378，2023。[28] J. Ting, M. Kim, J. Zhu, H. Sheng 和 Z. Zhang，“Hiper：面向高效基于机器人学习的控制的分层组合处理”，载于第52届年度国际计算机体系结构研讨会论文集，第313–326页，2025。[29] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng 和 G. Huang，“Deer-VLA：多模态大语言模型的动态推理，实现高效机器人执行”，《神经信息处理系统进展》，第37卷，第56619–56643页，2024。[30] J. Jabbour 和 V. Janapa Reddi，“自主机器中的生成式人工智能智能体：安全视角”，载于第43届IEEE/ACM计算机辅助设计国际会议论文集，第1–13页，2024。[31] ANYbotics，ANYmal技术规格。ANYbotics，2023年12月。访问日期：2024-09-19。[32] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano 和 A. De Luca，“利用基于惩罚的优化检索可行参数，对Franka Emika Panda机器人进行动态辨识”，IEEE《机器人与自动化快报》，第4卷，第4期，第4147–4154页，2019。[33] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah 和 S. Sankar，“大规模静默数据损坏”，arXiv预印本 arXiv:2102.11245，2021。[34] I. Moghaddasi, S. Gorgin 和 J.-A. Lee，“面向安全关键系统的可靠深度神经网络加速器：从老化角度进行综述”，IEEE Access，2023。

[35] X. Jiao, A. Rahimi, Y. Jiang, J. Wang, H. Fatemi, J. P. De Gyvez 和 R. K. Gupta，“Clim：一种跨层级工作负载感知功能单元时序错误预测模型”，*IEEE*计算机汇刊，第67卷，第6期，第771–783页，2017。[36] R. Huang, X. Jiang, S. Guo, P. Ren, P. Hao, Z. Yu, Z. Zhang, Y. Wang 和 R. Wang，“面向16/14纳米及更先进工艺的可变性与可靠性感知设计”，载于*2017*年*IEEE*国际电子器件会议（*IEDM*），第12–4页，IEEE，2017。[37] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost 和 P. Rech，“面向现场可编程门阵列中神经网络的选择性加固”，*IEEE*核科学汇刊，第66卷，第222页，2018年1月，第216页。[38] N. Khoshavi, A. Roohi, C. Broyles, S. Sargolzaei, Y. Bi 和 D. Z. Pan，“Shieldenn：用于容错深度神经网络架构的在线加速框架”，载于*2020*年第*57*届*ACM/IEEE*设计自动化会议（*DAC*），第1–6页，IEEE，2020。[39] E. Talpes, D. D. Sarma, G. Venkataramanan, P. Bannon, B. McGee, B. Floering, A. Jalote, C. Hsiong, S. Arora, A. Gorti，等，“特斯拉全自动驾驶计算机的计算解决方案”，*IEEE Micro*，第40卷，第2期，第25–35页，2020。[40] J. Zhang, K. Rangineni, Z. Ghodsi 和 S. Garg，“Thundervolt：为高能效深度学习加速器实现激进的电压欠缩放与时序错误韧性”，载于第*55*届年度设计自动化会议论文集，第1–6页，2018。[41] J. J. Zhang, K. Basu 和 S. Garg，“用于深度神经网络执行的基于脉动阵列的容错加速器”，*IEEE*设计与测试，第36卷，第5期，第44–53页，2019。[42] P. Pandey, P. Basu, K. Chakraborty 和 S. Roy，“Greentpu：提升近阈值张量处理单元的时序错误韧性”，载于第*56*届年度设计自动化会议（*2019*）论文集，第1–6页，2019。[43] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner，等，“Razor：一种基于电路级时序推测的低功耗流水线”，载于第*36*届*IEEE/ACM*国际微体系结构研讨会论文集，*2003*年。*MICRO-36*。，第7–18页，IEEE，2003。[44] N. D. Gundi, T. Shabanian, P. Basu, P. Pandey, S. Roy, K. Chakraborty 和 Z. Zhang，“Effort：提升近阈值张量处理单元的能量效率与错误韧性”，载于*2020*年第*25*届亚洲及南太平洋设计自动化会议（*ASP-DAC*），第241–246页，IEEE，2020。[45] P. N. Whatmough, S. K. Lee, D. Brooks 和 G.-Y. Wei，“深度神经网络引擎：一种用于物联网应用的28纳米时序错误容忍型稀疏深度神经网络处理器”，*IEEE*固态电路期刊，第53卷，第9期，第2722–2731页，2018。[46] K.-H. Huang 和 J. A. Abraham，“矩阵运算的基于算法的容错”，*IEEE*计算机汇刊，第100卷，第6期，第518–528页，1984。[47] S. Bal, C. S. Mummidi, V. D. C. Ferreira, S. Srinivasan 和 S. Kundu，“一种用于分块矩阵乘法的新型容错架构”，载于*2023*年欧洲设计、自动化与测试会议暨展览会（*DATE*），第1–6页，IEEE，2023。[48] M. Safarpour, R. Inanlou 和 O. Silvén，“低电压脉动阵列中的算法层面错误检测”，*IEEE*电路与系统汇刊第二辑：简报，第69卷，第2期，第569–573页，2021。[49] X. Xue, C. Liu, H. Huang, B. Liu, Y. Wang, B. Yang, T. Luo, L. Zhang, H. Li 和 X. Li，“Approxabft：面向视觉Transformer的近似基于算法的容错”，*arXiv*预印本 *arXiv:2302 10469*，2023。。[50] C. Schorn, A. Guntoro 和 G. Ascheid，“一种面向深度神经网络的高效位翻转韧性优化方法”，载于*2019*年欧洲设计、自动化与测试会议暨展览会（*DATE*），第1507–1512页，IEEE，2019。

[51] S. Kim、P. Howe、T. Moreau、A. Alaghi、L. Ceze 和 V. Sathe，“Matic：通过围绕错误进行学习，实现高效低电压神经网络加速器”，载于*2018*年欧洲设计、自动化与测试会议暨展览会（*DATE*），第 1–6 页，IEEE，2018。[52] Z. He、J. Lin、R. Ewetz、J.-S. Yuan 和 D. Fan，“噪声注入适配：针对神经网络映射的 ReRAM 交叉阵列非理想效应进行端到端适配”，载于*2019*年第*56*届年度设计自动化会议，第 1–6 页，2019。[53] Z. Wan、N. Chandramoorthy、K. Swaminathan、P.-Y. Chen、K. Bhardwaj、V. J. Reddi 和 A. Raychowdhury，“Mulberry：为节能型多智能体自主系统实现位错误鲁棒性”，载于第*29*届 *ACM* 编程语言与操作系统架构支持国际会议论文集，第*2*卷，第 746–762 页，2024。[54] S. Jha、S. Cui、T. Tsai、S. K. S. Hari、M. B. Sullivan、Z. T. Kalbarczyk、S. W. Keckler 和 R. K. Iyer，“利用时间数据多样性检测自动驾驶计算系统中的安全关键故障”，载于*2022*年第*52*届 *IEEE/IFIP* 可靠系统与网络国际会议（*DSN*），第 88–100 页，IEEE，2022。[55] Z. Wan、Y. Gan、B. Yu、S. Liu、A. Raychowdhury 和 Y. Zhu，“脆弱性自适应保护范式”，《*ACM* 通讯》，第 67 卷，第 9 期，第 66–77 页，2024。[56] D. Shah、Z. Y. Xue、K. Pattabiraman 和 T. M. Aamodt，“刻画并提升自主机器人加速器应对内存错误的韧性”，《*ACM* 网络物理系统汇刊》，第 8 卷，第 3 期，第 1–33 页，2024。[57] G. Li、S. K. S. Hari、M. Sullivan、T. Tsai、K. Pattabiraman、J. Emer 和 S. W. Keckler，“理解深度学习神经网络（DNN）加速器及应用中的错误传播”，载于高性能计算、网络、存储与分析国际会议论文集，第 1–12 页，2017。[58] B. Reagen、U. Gupta、L. Pentecost、P. Whatmough、S. K. Lee、N. Mulholland、D. Brooks 和 G.-Y. Wei，“Ares：量化深度神经网络韧性的框架”，载于第*55*届年度设计自动化会议论文集，第 1–6 页，2018。[59] A. Mahmoud、S. K. S. Hari、C. W. Fletcher、S. V. Adve、C. Sakr、N. R. Shanbhag、P. Molchanov、M. B. Sullivan、T. Tsai 和 S. W. Keckler，“优化 CNN 鲁棒性的选择性保护”，载于*ISSRE*，第 127–138 页，2021。[60] Z. Wan、A. Anwar、Y.-S. Hsiao、T. Jia、V. J. Reddi 和 A. Raychowdhury，“分析并提升基于学习的导航系统的容错能力”，载于*2021*年第*58*届 *ACM/IEEE* 设计自动化会议（*DAC*），第 841–846 页，IEEE，2021。[61] U. K. Agarwal、A. Chan 和 K. Pattabiraman，“评估大语言模型在瞬态硬件故障下的韧性”，载于*2023*年第*34*届 *IEEE* 软件可靠性工程国际研讨会（*ISSRE*），第 659–670 页，IEEE，2023。[62] T. Xie、J. Zhao、Z. Wan、Z. Zhang、Y. Wang、R. Wang、R. Huang 和 M. Li，“Realm：采用基于统计算法的容错能力，实现可靠且高效的大语言模型推理”，载于*2025*年第*62*届 *ACM/IEEE* 设计自动化会议（*DAC*），第 1–7 页，2025。[63] A. Chowdhery、S. Narang、J. Devlin、M. Bosma、G. Mishra、A. Roberts、P. Barham、H. W. Chung、C. Sutton、S. Gehrmann、等，“PaLM：借助 Pathways 扩展语言建模”，《机器学习研究杂志》，第 24 卷，第 240 期，第 1–113 页，2023。[64] D. Kalashnikov、J. Varley、Y. Chebotar、B. Swanson、R. Jonschkowski、C. Finn、S. Levine 和 K. Hausman，“MT-Opt：大规模连续多任务机器人强化学习”，*arXiv* 预印本 *arXiv:2104.08212*，2021。[65] M. Shridhar、J. Thomason、D. Gordon、Y. Bisk、W. Han、R. Mottaghi、L. Zettlemoyer 和 D. Fox，“ALFRED：用于理解日常任务具身指令的基准测试”，载于*IEEE/CVF* 计算机视觉与模式识别会议论文集，第 10740–10749 页，2020。

[66] C. Lynch、A. Wahid、J. Tompson、T. Ding、J. Betker、R. Baruch、T. Armstrong 和 P. Florence，“交互式语言：实时与机器人对话”，*IEEE*《机器人与自动化快报》，2023。[67] T. Brown、B. Mann、N. Ryder、M. Subbiah、J. D. Kaplan、P. Dhariwal、A. Neelakantan、P. Shyam、G. Sastry、A. Askell、等，“语言模型是小样本学习器”，神经信息处理系统进展，第 33 卷，第 1877–1901 页，2020。[68] V. Blukis、C. Paxton、D. Fox、A. Garg 和 Y. Artzi，“用于高级自然语言指令执行的持久空间语义表示”，载于机器人学习会议，第706–717 页，PMLR，2022。[69] X. Chen 、 J. Djolonga 、 P. Padlewski、B. Mustafa、S. Changpinyo、J. Wu、C. R. Ruiz、S. Goodman、X. Wang、Y. Tay、等，“PaLI-X：扩展多语言视觉语言模型”，*arXiv* 预印本 *arXiv:2305.18565*，2023。[70] W. H. Guss、B. Houghton、N. Topin、P. Wang、C. Codel、M. Veloso 和 R. Salakhutdinov，“MineRL：一个《我的世界》演示的大规模数据集”，载于第*28*届国际人工智能联合会议论文集，第 2442–2448 页，2019。[71] H. Liu、C. Li、Q. Wu 和 Y. J. Lee，“视觉指令调优”，神经信息处理系统进展，第 36 卷，34916，2023，第 34892 页。[72] B. Liu、Y. Zhu、C. Gao、Y. Feng、Q. Liu、Y. Zhu 和 P. Stone，“LIBERO：为终身机器人学习的知识迁移建立基准”，神经信息处理系统进展，第 36 卷，第 44776–44791 页，2023。[73] A. Awadalla、I. Gao、J. Gardner、J. Hessel、Y. Hanafy、W. Zhu、K. Marathe、Y. Bitton、S. Gadre、S. Sagawa、等，“OpenFlamingo：用于训练大型自回归视觉语言模型的开源框架”，*arXiv* 预印本 *arXiv:2308.01390*，2023。[74] O. Mees、L. Hermann、E. Rosete-Beas 和 W. Burgard，“CALVIN：面向长时域机器人操作任务的语言条件策略学习基准测试”，*IEEE*《机器人与自动化快报》，第 7 卷，第 3 期，第 7327–7334 页，2022。[75] D. Ghosh、H. R. Walke、K. Pertsch、K. Black、O. Mees、S. Dasari、J. Hejna、T. Kreiman、C. Xu、J. Luo、等，“Octo：开源通用机器人策略”，载于《机器人学：科学与系统》，2024。[76] A. O'Neill、A. Rehman、A. Maddukuri、A. Gupta、A. Padalkar、A. Lee、A. Pooley、A. Gupta、A. Mandlekar、A. Jain、等，“Open X-Embodiment：机器人学习数据集与 RT-X 模型——Open X-Embodiment 协作项目 0”，载于*2024*年 *IEEE* 国际机器人与自动化会议（*ICRA*），第 6892–6903 页，IEEE，2024。[77] J. Huang、S. Yong、X. Ma、X. Linghu、P. Li、Y. Wang、Q. Li、S.-C. Zhu、B. Jia 和 S. Huang，“三维世界中的具身通用智能体”，载于第*41*届国际机器学习会议论文集，第 20413–20451 页，2024。[78] W.-L. Chiang、Z. Li、Z. Lin、Y. Sheng、Z. Wu、H. Zhang、L. Zheng、S. Zhuang、Y. Zhuang、J. E. Gonzalez、等，“Vicuna：一款以 90%* 的 ChatGPT 质量惊艳 GPT-4 的开源聊天机器人”，参见 *https://vicuna.lmsys. org*（访问日期：*2023*年*4*月*14*日），第 2 卷，第 3 期，第 6 页，2023。[79] D. Z. Chen、A. X. Chang 和 M. Nießner，“ScanRefer：使用自然语言在 RGB-D 扫描中定位三维物体”，载于欧洲计算机视觉会议，第 202–221 页，Springer，2020。[80] S. K. Ramakrishnan、A. Gokaslan、E. Wijmans、O. Maksymets、A. Clegg、J. M. Turner、E. Undersander、W. Galuba、A. Westbury、A. X. Chang、M. Savva、Y. Zhao 和 D. Batra，“Habitat-Matterport三维数据集（HM3D）：面向具身人工智能的 1000 个大规模三维环境”，载于第三十五届神经信息处理系统会议数据集与基准赛道，2021。[81] S. Lifshitz、K. Paster、H. Chan、J. Ba 和 S. McIlraith，“STEVE-1：用于《我的世界》文本到行为的生成模型”，神经信息处理系统进展，第 36 卷，第 69900–69929 页，2023。[82] G. Xiao、J. Lin、M. Seznec、H. Wu、J. Demouth 和 S. Han，“SmoothQuant：用于训练后量化的准确高效方法，

大语言模型”，载于国际机器学习会议，第38087–38099页，PMLR，2023年。[83] J. Lin、J. Tang、H. Tang、S. Yang、X. Dang和S. Han，“Awq：面向大语言模型压缩与加速的激活感知权重量化”，*arXiv*预印本 *arXiv:2306.00978*，2023年。[84] Z. Yuan、L. Niu、J. Liu、W. Liu、X. Wang、Y. Shang、G. Sun、Q. Wu、J. Wu和B. Wu，“Rptq：面向大语言模型的基于重排序的训练后量化”，*arXiv*预印本 *arXiv:2304.01089*，2023年。[85] Z. Liu、C. Zhao、I. Fedorov、B. Soran、D. Choudhary、R. Krishnamoorthi、V. Chandra、Y. Tian和T. Blankevoort，“Spinquant：采用学习旋转的大语言模型量化”，*arXiv*预印本 *arXiv:2405.16406*，2024年。[86] S. Ashkboos、A. Mohtashami、M. Croci、B. Li、P. Cameron、M. Jaggi、D. Alistarh、T. Hoefler和J. Hensman，“Quarot：旋转后大语言模型中的无异常值4位推理”，神经信息处理系统进展，第37卷，第100213–100240页，2024年。[87] A. Tseng、J. Chee、Q. Sun、V. Kuleshov和C. De Sa，“QuIP#：借助Hadamard非相干性和格码实现更优的大语言模型量化”，载于国际机器学习会议，第48630–48656页，PMLR，2024年。[88] S. Kim、Y. Choi、J. Oh、B. Kim和H.-J. Yoo，“LightRot：面向精确低比特大语言模型推理的轻量级旋转方案与架构”，*IEEE*新兴与精选电路和系统专题期刊，2025年。[89] Z. Zhang、R. Wei、M. Li、Y. Lin、R. Wang和R. Huang，“READ：利用关键输入模式缩减实现可靠性增强的加速器数据流优化”，载于*2023 IEEE/ACM*计算机辅助设计国际会议（*ICCAD*），第1–9页，IEEE，2023年。[90] J.-S. Kim和J.-S. Yang，“DRIS-3：基于故障分析的三维堆叠晶圆内存深度神经网络可靠性改进方案”，载于第*56*届年度设计自动化会议（*2019*）论文集，第1–6页，2019年。[91] S.-T. Zhou、S. Katariya、H. Ghasemi、S. Draper和N. S. Kim，“通过联合优化单元尺寸、冗余和纠错码，最小化低电压SRAM阵列的总面积”，载于*2010 IEEE*国际计算机设计会议，第112–117页，IEEE，2010年。[92] M. C. Casey、B. L. Bhuva、S. A. Nation、O. A. Amusan、T. D. Loveless、L. W. Massengill、D. McMorrow和J. S. Melinger，“单粒子效应对超低功耗CMOS电路的影响”，载于*2009 IEEE*国际可靠性物理研讨会，第194–198页，IEEE，2009年。[93] T. Tambe、C. Hooper、L. Pentecost、T. Jia、E.-Y. Yang、M. Donato、V. Sanh、P. Whatmough、A. M. Rush、D. Brooks、等，“EdgeBERT：面向延迟感知多任务自然语言处理推理的句子级能耗优化”，载于*MICRO-54*：第*54*届*IEEE/ACM*国际微体系结构研讨会，第830–844页，2021年。[94] B. Sangchoolie、K. Pattabiraman和J. Karlsson，“一位是否（不）够用：单个与多个位翻转错误影响的实证研究”，载于*2017*年第*47*届*IEEE/IFIP*国际可信系统与网络会议（*DSN*），第97–108页，IEEE，2017年。[95] Y. He、P. Balaprakash和Y. Li，“Fidelity：面向深度学习加速器的高效韧性分析框架”，载于*2020*年第*53*届*IEEE/ACM*国际微体系结构研讨会（*MICRO*），第270–281页，IEEE，2020年。[96] Y. S. Hsiao、Z. Wan、T. Jia、R. Ghosal、A. Mahmoud、A. Raychowdhury、V. J. Reddi、等，“机器人操作系统中的静默数据损坏：使用自主无人机进行端到端系统级故障分析的案例”，*IEEE*集成电路与系统计算机辅助设计汇刊，2023年。[97] G. Papadimitriou和D. Gizopoulos，“揭秘系统漏洞栈：跨层的瞬态故障效应”，载于*2021 ACM/IEEE*第*48*届年度国际计算机体系结构研讨会（*ISCA*），第902–915页，IEEE，2021年。[98] R. W. Hamming，“检错与纠错码”，贝尔系统技术期刊，第29卷，第2期，第147–160页，1950年。

[99] P. H. Hochschild、P. Turner、J. C. Mogul、R. Govindaraju、P. Ranganathan、D. E. Culler和A. Vahdat，“不计数的核心”，载于操作系统热点研讨会论文集，第9–16页，2021年。[100] A. Hedayat和W. D. Wallis，“Hadamard矩阵及其应用”，《统计年鉴》，第1184–1238页，1978年。[101] C. F. Van Loan，“无处不在的克罗内克积”，计算与应用数学期刊，第123卷，第100号，第85–125页，2000年。[102] R. B. Ash，信息论。Courier Corporation，2012年。[103] S. J. Kim、D. Kim、Y. Pu、C. Shi、S. B. Chang和M. Seok，“0.5–1 V、90–400 mA、基于事件驱动控制和多米诺采样与调节的模块化分布式数字低压差稳压器”，*IEEE*固态电路期刊，第56卷，第9期，第2781–2794页，2021年。[104] H. Jun、J. Cho、K. Lee、H.-Y. Son、K. Kim、H. Jin和K. Kim，“HBM（高带宽内存）动态随机存取存储器工艺技术与架构”，载于*2017 IEEE*国际存储器研讨会（*IMW*），第1–4页，IEEE，2017年。[105] A. Samajdar、Y. Zhu、P. Whatmough、M. Mattina和T. Krishna，“SCALE-Sim：脉动式卷积神经网络加速器模拟器”，*arXiv*预印本 *arXiv:1811.02883*，2018年。[106] I. Loshchilov和F. Hutter，“解耦权重衰减正则化”，2019年。

A 模型详情

这项工作涉及多个模型，其详细架构（例如层数、隐藏维度、参数数量和浮点运算次数）都是至关重要的超参数。表7至9 展示了规划器、控制器以及我们提出的熵预测器的详细架构。表4 进一步汇总了各模型的参数总数和计算量。为简洁起见，此处仅列出主要模块，省略了一些次要组件。Conv2d 的卷积核大小默认为 3。

表 7. 规划器的模型架构

模型	#层数	隐藏维度	MLP维度
JARVIS-1	32	4,096	14,336
OpenVLA	32	4,096	11,008
RoboFlamingo	24	2,048	8,192

表8. 控制器的模型架构

模型	组成部分	# 层数	隐藏维度	MLP维度
JARVIS-1	Img*	10	3-256	128-256
	解码器	4	1,024	4,096
RT-1	MaxViT	11	64-768	384-3 ,072
	FiLM	23	1,536	256-3 ,072
	ViT	6	768	3,072
Octo	T5	12	384	1,152
	Obs*	8	6-384	32-512
	Head	3	256	1,024

*Conv2d层，其中隐藏维度和MLP维度分别指输入通道和输出通道。

B 任务描述

本文评估了四个基准测试集中的 21 项任务，为简洁起见，全文均使用其缩写。详细的任务描述见表10。

表 9. 熵预测器的模型架构

积木	层类型	配置
图像处理（卷积神经网络）	二维卷积	输入=3，输出=16，步幅=3，填充=1
	ReLU	
	二维最大池化	卷积核_大小=2
	二维卷积	输入通道=16，输出通道=32，步幅=3，填充=1
	ReLU	
	二维最大池化	卷积核大小=2_
	二维卷积	输入=32，输出=64，步幅=3，填充=1
	ReLU	
	二维平均池化	输出_尺寸=1
提示多层感知机	线性层	输入=512，输出=64
融合多层感知机	拼接	
	线性层	输入=128，输出=128
	ReLU	
	线性层	输入=128，输出=1

表10。 任务描述

基准测试	缩写	任务描述
《我的世界》	木制的	在丛林中获得一把木镐
	石头	在平原上获得一把石镐
	木炭	在平原获得木炭
	鸡肉	在平原获得熟鸡肉
	coal	在热带草原获得煤
	iron	在平原获得铁剑
	wool	在平原获得 5 个白色羊毛
	seed	在热带草原中获得 10 颗小麦种子
	log	在森林中获得 10 根原木
LIBERO	wine	将葡萄酒瓶放在柜子顶部
	字母	拿起字母汤并将其放置在篮子中
	bbq	拿起烧烤酱并将其放置在篮子中
CALVIN	按钮	按下按钮以关闭LED灯
	积木	滑动积木，使其落入抽屉
	把手	拉把手打开抽屉
OXE	茄子	把茄子放进篮子
	coke	抓住单个打开的可乐罐
	胡萝卜	把胡萝卜放在盘子上
	open	打开中间的抽屉
	move	移动到 Google Baked Tex 附近
	放置	放入关闭的顶层抽屉

C 熵到电压映射策略

图 21 展示了 JARVIS-1 的熵到电压映射策略 A 至 F 的详细信息，这些策略是从 100 多个候选方案中搜索得出的。

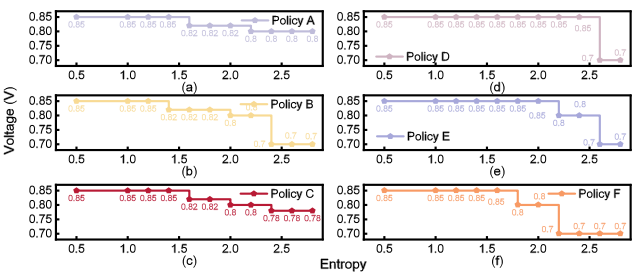

图21。 选定的熵到电压映射策略