

# 端侧具身智能系统:挑战与展望



## On-Device Embodied Intelligence System: Challenges and Opportunities

徐梦炜/Xu Mengwei, 徐大亮/Xu Daliang

(北京邮电大学计算机学院(国家示范性软件学院), 中国 北京 100876)  
(School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China)

DOI: 10.12142/ZTETJ.202604008

网络出版地址: <https://link.cnki.net/urlid/34.1228.TN.20260828.1141.004>

网络出版日期: 2026-08-28

收稿日期: 2026-06-10

**摘要:** 端侧具身智能将具身大模型从云端远程推理下沉至设备本体并实现闭环执行, 是机器人进入真实物理世界的关键系统基础。揭示视觉-语言-动作模型(VLA)与世界-动作模型(WAM)两类模型在多模态感知、扩散动作生成及世界模型滚动中产生的高计算与高访存开销, 与端侧硬件在10~100 Hz控制频率、有限功耗及网络不稳定条件下可靠运行之间的系统性矛盾。针对该矛盾, 提出以动作成功率、闭环时延与能效为共同优化目标, 围绕少步化生成、动作感知压缩、缓存复用、异步分块执行与专用运行时等策略, 开展算法与系统层面的协同设计。

**关键词:** 端侧智能; 具身智能; 端侧推理优化

**Abstract:** Endowing embodied intelligence on the device side shifts large embodied models from cloud-based remote inference to on-device closed-loop execution, serving as a crucial system foundation for robots to operate in real physical environments. Two representative model families—vision-language-action models (VLA) and world-action models (WAM)—introduce high computational and memory-access overheads through multimodal perception, diffusion-based action generation, and world model rolling. These overheads create a systematic contradiction with the requirements of end-side hardware, which must support 10–100 Hz control frequencies, operate under limited power budgets, and maintain reliability despite unstable network conditions. To address this contradiction, action success rate, closed-loop latency, and energy efficiency are adopted as the joint optimization objectives. Algorithm-system co-design strategies are proposed, centering on few-step generation, action-aware compression, cache reuse, asynchronous chunk execution, and specialized runtimes.

**Keywords:** on-device intelligence; embodied intelligence; on-device inference optimizations

**引用格式:** 徐梦炜, 徐大亮. 端侧具身智能系统: 挑战与展望 [J]. 中兴通讯技术, 2026, 32(4): 47–55. DOI: 10.12142/ZTETJ.202604008

**Citation:** Xu M W, Xu D L. On-device embodied intelligence system: challenges and opportunities [J]. ZTE technology journal, 2026, 32(4): 47–55. DOI: 10.12142/ZTETJ.202604008

莫拉维克悖论指出, 人类感到困难的高级抽象推理, 对计算机而言往往轻而易举; 而儿童都能完成的感知、运动及物理交互, 对机器来说却极具挑战。过去十余年, 深度学习尤其是大模型技术在语言、视觉、代码等数字空间任务中取得了显著突破, 但仍难以使机器人在开放物理环境中可靠地完成抓取、导航、协作、维护等基础实体操作任务。随着人工智能应用从屏幕、文本及云端服务扩展至家庭、工厂、交通、医疗和空间设施等真实场景, 面向物理世界交互的具身智能正成为人工智能下一个十年的核心挑战。具身智能系统需在连续观测、有限算力和实时约束条件下, 将模型能力转化为可执行动作, 其成效直接决定大模型能否从数字

助手发展为物理代理。

近年来, 具身智能模型架构快速演进, 形成了两种具有代表性的技术路线: 第一类为视觉-语言-动作模型(VLA), 以视觉-语言模型为主干, 将相机观测、语言指令及历史状态统一编码, 直接映射为机器人动作; 第二类为世界-动作模型(WAM), 以预测未来状态(如图像、隐空间特征或其他表征形式)为核心训练目标。该模型继承世界模型的思想, 并针对真实机器人的动作生成需求进行了专项优化。这两类模型共同推动具身智能从任务专用策略向通用基础模型演进, 使机器人有望在更复杂、更开放的场景中理解指令, 预判环境变化并生成连续动作。然而, 具身智能从实验室走向真实部署, 仍面临系统效率方面的关键挑战。机器人在与物理世界实时交互过程中需持续决策, 网络抖动将直接导致控制性能退化、断连, 甚至可能引发任务失败。为此, 具身智能正从依赖云端大模型的远程推理, 转向模型在设备本体

基金项目: 国家自然科学基金项目(62522202)

的就地部署与闭环控制。然而，机器人实现稳定控制通常需要 10~100 Hz 的控制频率，而参数量达数十亿的 VLA 模型及基于视频扩散的 WAM 模型，在端侧计算平台上往往难以满足这一推理效率要求。基于此，本文从算法与系统两个层面面对高效端侧具身智能系统展开讨论与展望，二者的关键技术及其相互关系如图 1 所示。

## 1 模型架构与瓶颈

端侧具身智能主要由两类大模型驱动：一类是以视觉-语言模型为主干，将观测与指令直接映射为动作的 VLA；另一类则通过预测未来观测来支撑规划与控制的 WAM。二者的推理架构与计算特性不同，所面临的端侧瓶颈亦各有侧重。本节首先概述两类模型的执行流程，继而剖析其各自存在的瓶颈，为后续各层优化提供依据。

VLA 的推理流程如下：一个 VLA 模型通常由视觉编码器、视觉-语言主干网络及动作头（action head）级联组成。推理时，机器人当前的视觉观测首先由编码器转换为视觉词元，这些词元与经分词处理的语言指令共同输入主干网络，通过一次前向计算（即预填充阶段）获得融合场景理解与指令意图的上下文表征；随后，动作头以该表征为条件，生成低层控制指令，通常以动作分块的形式一次性输出未来若干步动作并按序执行。就动作头的实现方式而言，早期方案将连续动作离散化后以逐词元自回归方式生成；近年来的高性能模型则多采用基于扩散（diffusion）或流匹配（flow

matching）的动作专家（action expert）。这类方法从随机噪声出发，通过多步去噪迭代生成连续动作向量，因其在连续控制与多模态动作分布建模方面的优势，正逐渐成为主流。VLA 和 WAM 的模型架构及推理过程如图 2 所示。

VLA 的端侧瓶颈主要体现在以下几个方面，且彼此关联，共同制约系统效率。首先，视觉编码器与主干网络承担了推理过程中的主要算力开销：单帧图像通常被展开为数百至上千个视觉词元，其数量远超语言与动作词元，导致预填充阶段的注意力计算与前馈网络运算量显著偏大。由于该阶段每次访存均伴随大量浮点运算，性能主要受限于算力而非访存带宽，因此基于 roofline 模型可判定，视觉编码与主干部分属于计算受限型。

相比之下，动作专家的多步去噪过程在每一步均需反复读取模型权重与键值缓存，而单步运算量相对有限，这导致其性能瓶颈转向内存带宽。本文在两类端侧图形处理器（GPU）上的实测结果（如表 1 所示）进一步表明：在 Jetson Thor 上， $\pi 0.5$  模型的动作专家时延占比约为 38%；而在消费级 RTX 5090 上，GR00T N1.7 模型的动作头时延占比升至 59.2%，但其模型算力利用率仅为 13.2%，而主干部分则达到 33.0%。主干网络加速越充分，动作生成环节的相对开销反而越突出，单纯提升峰值算力无法有效消除这一瓶颈。

这一计算受限的主干网络与访存受限的动作专家并存的两阶段结构，直接导致了控制频率与模型推理时延之间的失配：未经优化的 3B~7B 参数级 VLA 在边缘平台上通常仅能

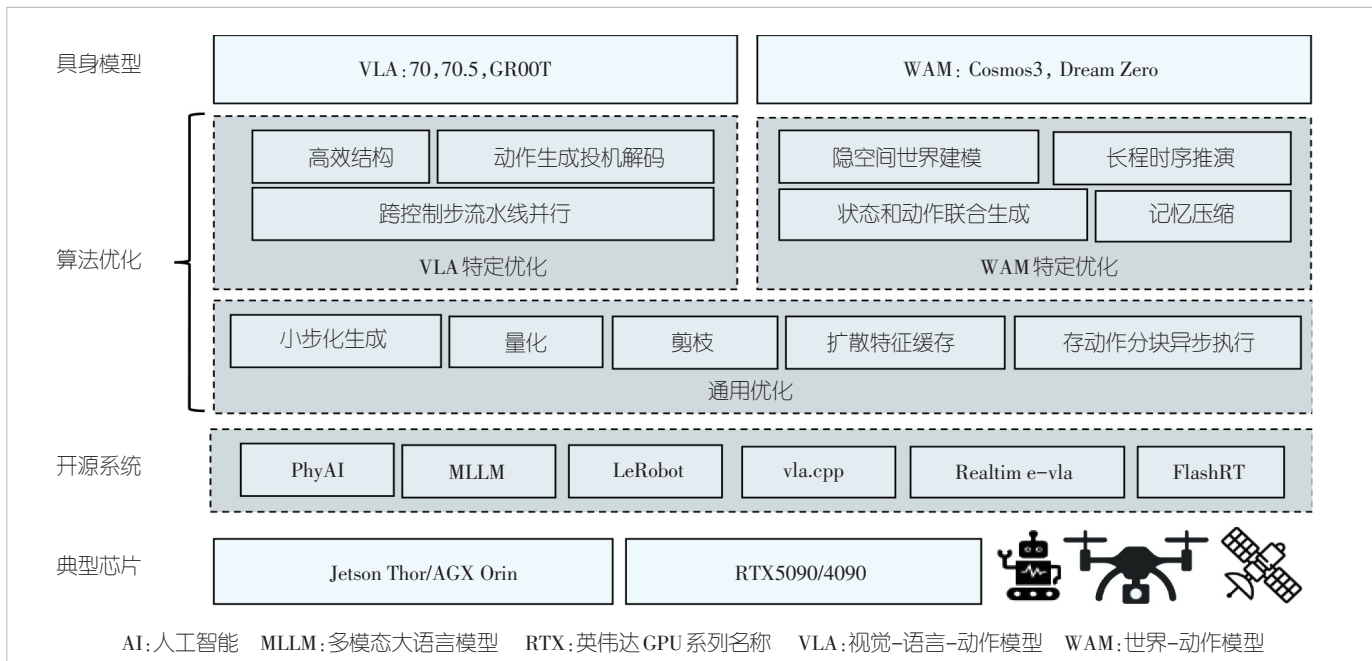


图1 端侧具身智能算法、系统与芯片的全栈协同总览

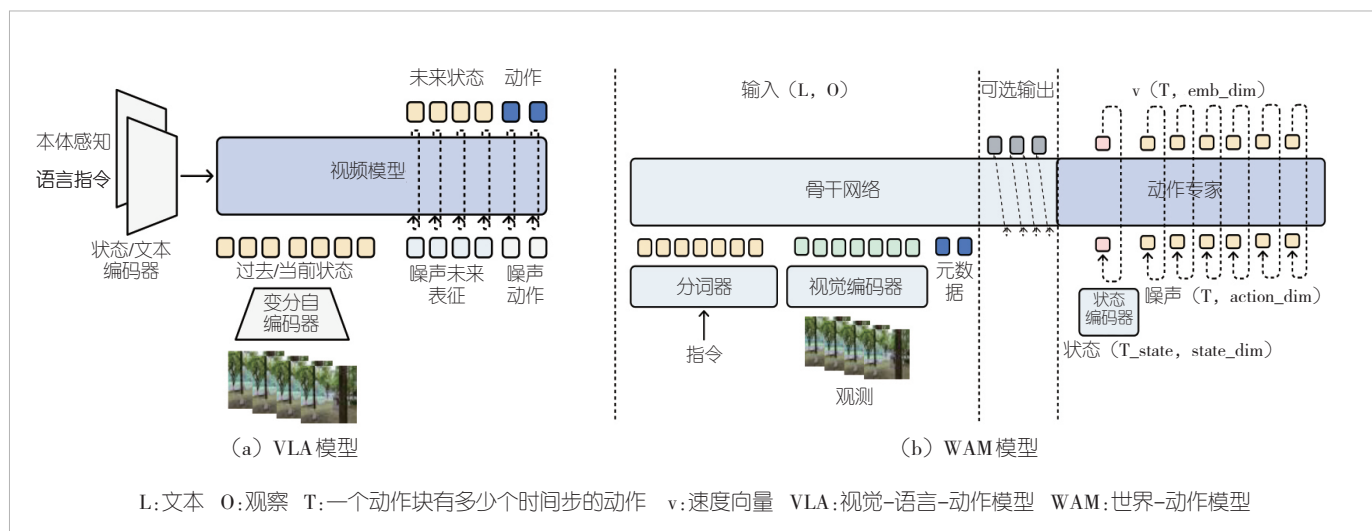


图2 VLA和WAM模型架构及推理过程

表1  $\pi 0.5$ 、GR00T N1.7 与 Cosmos3-Nano-Policy 的阶段剖析与瓶颈定位(批大小为1)

| 指标                                      | $\pi 0.5$<br>(Jetson Thor) | GR00T N1.7<br>(RTX 5090) | Cosmos3-Nano<br>(Jetson Thor) |
|-----------------------------------------|----------------------------|--------------------------|-------------------------------|
| 端到端时延/ms                                | 124.6                      | 20.5                     | 8 121                         |
| 等效频率/Hz                                 | 8.00                       | 48.80                    | 0.12                          |
| 主干/条件阶段/ms                              | 76.8<br>(61.6%)            | 8.3<br>(40.8%)           | 1 824<br>(22.5%)              |
| 动作头/生成阶段/ms                             | 47.8<br>(38.4%)            | 12.1<br>(59.2%)          | 6 298<br>(77.5%)              |
| 该阶段算术强度/<br>(FLOP·byte <sup>-1</sup> )  | 50.8                       | 135.8                    | 926.2                         |
| 设备性能瓶颈拐点/<br>(FLOP·byte <sup>-1</sup> ) | 450.4                      | 139.4                    | 448.9                         |
| 该阶段模型算力<br>利用率/%                        | 7.6                        | 13.2                     | 65.8                          |
| 瓶颈判定                                    | 访存受限                       | 访存受限                     | 计算受限                          |

FLOP:浮点运算量 RTX:英伟达GPU系列名称

达到约 6~19 Hz 的处理频率。本文在典型端侧设备 Jetson Thor 上对当前最先进的开源模型  $\pi 0.5$  进行了性能度量分析,实验数据如表 1 所示。结果表明,官方参考实现的等效推理频率仅为 5.6 Hz,即使经过系统优化,也仅提升至 9.3 Hz,均远低于灵巧操控任务所期望的 30 Hz 目标频率。

此外,预填充阶段与动作生成阶段所关注的词元并不一致:前者侧重于全局场景理解,后者则聚焦于与当前操作密切相关的局部区域。因此,若将面向通用视觉-语言模型设计的视觉词元剪枝方法直接迁移至 VLA 场景,往往会导致动作所依赖的关键词元被误删,进而损害任务成功率。这一

现象要求端侧 VLA 的压缩策略必须具备动作感知能力。

WAM 的推理流程与瓶颈如下。与 VLA 不同,世界模型在具身场景中有两种典型用法:一是作为动作生成器,在预测未来的同时直接输出动作;二是作为可闭环滚动的环境模拟器,对候选动作序列进行前瞻推演并择优。无论采用何种方式,其计算代价均集中于未来预测的生成环节。未来帧或隐状态需经多步扩散采样方可获得,单次预测的开销已较为高昂;面向规划的长程滚动生成则进一步放大这一代价,一方面使内存占用随滚动步数线性增长,另一方面逐步累积预测误差,二者共同限制了可用的规划深度。此外,在像素空间逐帧生成的代价远高于在隐空间中进行滚动生成。

本文对 Cosmos3-Nano 的实测结果亦显示,在端到端时延中,单次动作分块生成所包含的 4 步去噪循环占比约 78% (Jetson Thor, 见表 1)。与 VLA 的动作专家不同, WAM 的生成模块在批大小为 1 时即已呈现计算受限特征,其算术强度约为 926 FLOP/byte, 远高于 Jetson Thor 的性能瓶颈拐点(约 450 FLOP/byte),模型算力利用率达 65.8% (图 3 (b))。因此,其时延主要由多步去噪的原始算力需求决定,单纯提升访存带宽的收益有限。更为严峻的是显存约束:该模型单次推理的峰值显存已达 30 GiB 量级,在 32 GB 的消费级 RTX 5090 上因显存溢出而无法运行。此外,当前能够实现实时滚动的世界模型仍多依赖服务器级 GPU,如何将其压缩至端侧设备并在控制节拍内完成滚动生成,仍是一个亟待解决的关键问题。

## 2 高效算法

高效算法在不依赖专用硬件的前提下降低推理成本,既

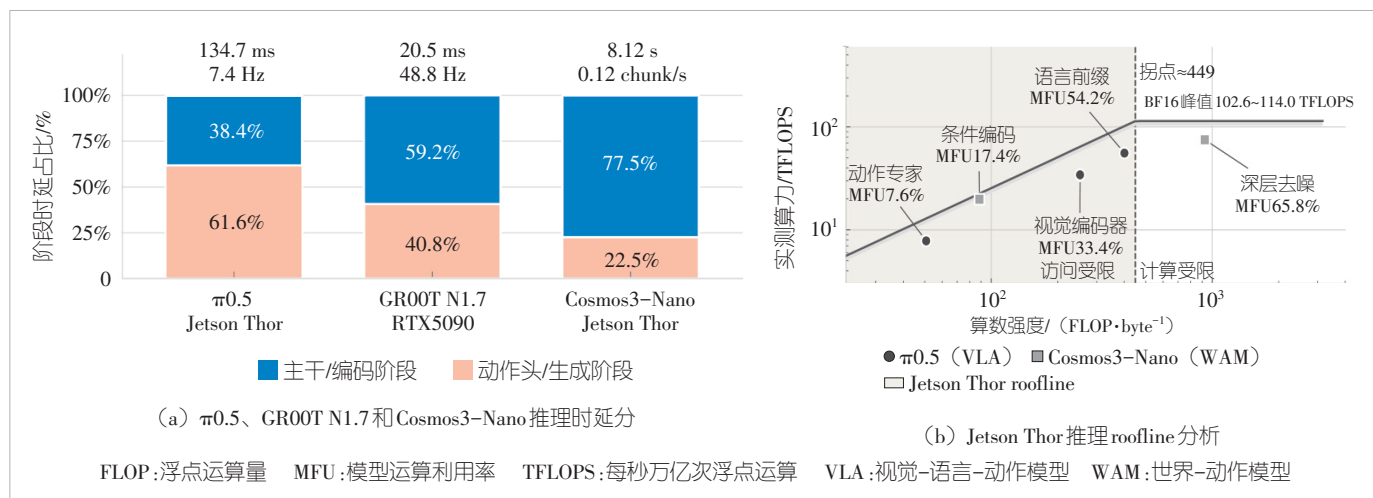


图3  $\pi 0.5$ (VLA)、GR00T N1.7(VLA)与 Cosmos3-Nano(WAM)在 Jetson Thor 与 RTX 5090 上的阶段时延构成及 Jetson Thor 上推理的 roofline 分析

包括直接对模型自身进行改动的方法,也包括对推理执行过程进行改造的方法,二者均属于算法层面的手段。鉴于 VLA 的扩散动作专家与 WAM 的视频扩散 Transformer (DiT) 均以迭代式扩散为计算核心,本节首先论述二者共享的扩散生成加速策略,随后分别讨论 VLA 与 WAM 各自利用具身结构特征的专有优化方法。

## 2.1 通用算法优化

1) 少去噪步生成。扩散与流匹配生成过程需经多步迭代去噪,推理时延随步数增加而近似线性增长,因此压缩采样步数是最直接的加速途径。一致性流匹配蒸馏方法将多步教师模型压缩为单步或少步学生模型,在尽量保持动作生成质量的同时显著提升推理速度。例如,OneDP<sup>[1]</sup>以最小化扩散链 KL 散度得分差为训练目标,将多步扩散策略蒸馏为单步生成器,在 NVIDIA V100 GPU 上使动作预测频率由原来的 1.5 Hz 提升至 62 Hz;FreqPolicy<sup>[2]</sup>进一步考虑动作本质为时序信号,在频域施加一致性约束以保障单步生成的动作质量;Streaming Diffusion Policy<sup>[3]</sup>则维护一个含变噪声的动作缓冲区,仅对当前待执行动作完全去噪,后续动作保留噪声状态,从而以较少计算量产出可执行动作。

2) 降低权重或激活精度。量化是另一条与减步正交的加速途径。面向视频 DiT 的后训练量化方法(如 ViDiT-Q、PTQ4DiT)可将权重与激活值压缩至 4~8 bit,从而有效削减访存开销;而 VLA 侧的量化需与控制任务紧密耦合,须以操控成功率为核心约束条件。

3) 扩散特征缓存。在迭代去噪过程中,相邻步骤(或相邻生成块)的中间特征通常高度相似,复用这些特征可在不重新训练模型的前提下跳过冗余计算。需要指出的是,通

用扩散缓存方法因架构差异难以直接迁移至策略与世界模型,因此已有研究提出了针对性设计。例如,块级自适应缓存(BAC)<sup>[4]</sup>为扩散策略设计了块级自适应缓存机制,并指出不同时间步与网络块之间的特征相似性呈非均匀变化。

4) 动作分块异步执行。动作分块虽能提升吞吐量,但会在分块边界引入响应滞后:生成下一分块所需的推理时延,可能导致机器人在动态环境中无法及时响应。异步执行策略的基本思路是,在执行当前分块的同时生成下一分块,从而将推理时延隐藏于执行过程之中。由于该方法仅涉及分块的生成与执行调度,可适用于任意输出动作分块的策略,因此具有良好的通用性。其代表性工作实时分块执行(RTC)<sup>[5]</sup>将该问题形式化为图像修复问题:将已承诺执行的动作冻结为已知量,其余部分作为待修复的未知量,通过引导项确保衔接平滑。该方法无须训练,可适配任意扩散或流匹配策略,能够容忍超过 300 ms 的推理时延。

## 2.2 VLA 模型特定优化

VLA 模型特定优化的共性,在于显式利用机器人时序连续性、动作相关性等结构化特征,可归纳为高效架构、动作感知压缩、动作生成加速与推理执行级加速 4 类。

1) 高效架构。此类工作针对主干网络与编码器的高开销问题,从多条路径对模型结构进行重塑。为缓解自注意力的二次复杂度,RoboMamba<sup>[6]</sup>以线性复杂度的状态空间模型替代标准注意力主干;为降低整体模型规模,TinyVLA、SmolVLA 等采用亚十亿参数级主干网络,并配合扩散动作专家实现轻量化;针对第 1 节中指出的视觉词元主导预填充阶段计算量的问题,FastVLM<sup>[7]</sup>从编码器设计层面入手,直接削减视觉词元数量,使首词元时延显著下降。

2) 动作感知压缩。与通用模型压缩不同,这类方法以操控成功率而非重建精度作为评价准则,将有限的比特或词元预算优先分配给对动作生成最为关键的部分。在量化方面,基于显著性的量化模仿学习(SQIL)<sup>[8]</sup>以动作显著性与量化鲁棒的行为克隆损失为依据,将精度预算集中于对控制性能影响较大的关键状态,使得四比特模型在Jetson GPU上提速约2.5倍,且性能近乎无损;BitVLA<sup>[9]</sup>进一步采用原生三值权重,模型内存开销仅为1.4 GB,相当于OpenVLA原始模型的1/11。在剪枝方面,VLA-Pruner<sup>[10]</sup>针对预填充阶段与动作解码阶段注意力分布存在差异的语义-动作鸿沟,对每个词元同时按语义重要性与动作相关性进行双级评分,从而在删减冗余的同时保留动作真正依赖的关键词元。

3) 动作生成加速。该方法直接缩短动作专家(或自回归动作头)的解码过程。一类工作聚焦于减少迭代步数或实现并行化解码,例如CEED-VLA<sup>[11]</sup>通过一致性蒸馏与早退策略改造Jacobi解码,在NVIDIA H100 GPU上将OpenVLA模型的推理加速约4.1倍。另一类研究则改进动作的表示与解码形式,例如OpenVLA-OFT<sup>[12]</sup>以并行解码、动作分块以及连续动作表示替代原有的自回归离散生成方式,在NVIDIA A100 GPU上实现约26倍加速,并在ALOHA真机上达到25 Hz的动作控制频率;FAST<sup>[13]</sup>则通过离散余弦变换压缩动作词元序列,从表示层面降低生成代价。

4) 推理执行级加速。除对模型结构与权重进行改动外,现有方法还致力于在不改变模型本身的前提下,重塑解码与服务执行流程。投机动作解码通过草稿-验证机制缓解逐词元生成带来的串行瓶颈,Spec-VLA<sup>[14]</sup>与FLASH<sup>[15]</sup>分别覆盖自回归与扩散两类动作头,后者借助动作专家并行验证流一致性,在单张RTX 4090D GPU上取得约3.04倍加速。键值(KV)缓存复用利用帧间冗余降低计算开销,如VLA-Cache<sup>[16]</sup>复用相邻帧中变化极小的静态视觉词元的键值表示,并选择性重算任务相关词元,从而削减每步视觉处理开销,在单张RTX 4090 GPU上对OpenVLA模型进行推理时最高取得约1.7倍加速。跨请求流水线方法ActionFlow<sup>[17]</sup>进一步将访存受限的解码阶段与计算受限的预填充阶段跨控制步进行打包调度,使算力与带宽资源同时趋于饱和。

### 2.3 WAM模型特定优化

WAM的端侧效率,归根结底取决于如何以较低计算代价从世界模型中生成动作。本文从两个正交维度刻画其动作生成范式:其一为预测与动作在时序上的耦合方式,可分为先完成未来预测再据此导出动作的顺序模式,以及在同一生成过程中同时产出预测与动作的联合模式;其二为动作所依

赖的隐状态成熟度,即依赖于完全去噪后的预测结果,还是依赖于尚未完成去噪的中间表征。在上述二维划分基础上,本节进一步补充两条与之正交的降本路径:采用隐空间表征替代显式像素生成,以及面向长程推演的执行级优化。

1) 顺序范式:从完整状态预测到动作生成。顺序范式遵循先预测、后动作的执行流程。其早期实现依赖完整的未来状态预测,保真度较高但计算开销大,例如Seer<sup>[18]</sup>通过预测未来帧,再由逆动力学模型从中导出动作;DreamVLA<sup>[19]</sup>则转而预测动态区域、深度与语义等紧凑型世界知识表征,而非重建整幅未来图像,以降低预测成本。

2) 联合范式:状态与动作联合生成。联合范式通过共享的去噪或前向过程同时产出未来状态与动作,使二者能够利用共同的物理动态先验信息。例如,PAD<sup>[20]</sup>在单一扩散过程中联合去噪图像与动作,WorldVLA<sup>[21]</sup>则以自回归方式统一建模动作与世界模型。该范式的局限在于,动作需等待完整去噪过程结束后方可获得。因此,如何在联合框架内从中间表征提前产出动作,以兼顾信息共享与低时延,迄今尚无成熟的代表性工作,是一个值得探索的开放问题;不过,统一世界模型(UWM)<sup>[22]</sup>已为各模态设置独立的噪声时间步,为从中间表征提前生成动作提供了基础支持。

3) 隐空间世界建模:一条更为根本的降本思路是放弃像素级生成,转而在隐空间中完成预测。LAPA<sup>[23]</sup>通过无标注视频学习离散隐动作并据此进行预训练,FLARE<sup>[24]</sup>则使策略特征对齐未来观测的隐嵌入,而不生成完整的未来像素表征。二者均避免了显式像素级预测,从而有效降低世界模型在表示与生成层面的计算负担。

4) 长程时序推演与记忆压缩:在执行层面,流式滚动与长程记忆通过联合去噪、注意力汇聚及历史记忆压缩,抑制长程误差累积,并管理日益膨胀的键值缓存。代表性工作Rolling Forcing<sup>[25]</sup>同时采用了3项设计:以联合去噪方式对窗口内多帧施加递增噪声,一次性更新整个窗口而非逐帧迭代采样,从而放松相邻帧间的严格因果约束;以注意力汇聚长期保留首帧的KV状态作为全局锚点,并调整其旋转位置编码偏移,使相对位置保持有界;在显著加长的去噪窗口上施行少步蒸馏,以自生成历史为条件缓解暴露偏差。

## 3 高效端侧具身推理系统

面向端侧具身负载的推理引擎与运行时近年来集中涌现,开源生态已初具规模:既有面向物理智能模型进行全栈优化的专用框架,也有从端侧大语言模型演进而来、覆盖多种异构后端的通用底座,还有由机器学习社区维护的端到端参考实现。本文在Jetson Thor与RTX 5090两类端侧GPU

上,对上述运行时承载 $\pi 0$ 、 $\pi 0.5$ 、GR00T N1.7与Cosmos3–Nano4个典型具身模型的单请求时延进行了系统性度量,结果见表2;各运行时所支持的模型族、可用精度、硬件后端及端云协同形态等方面的差异见表3。

1) PhyAI<sup>[26]</sup> (github.com/mingti-org/phyai): 面向物理智能模型(VLA和WAM)的高性能推理框架。其核心特点是以同一套代码路径贯通端侧本体、边缘服务器与云端:模型评测、云端强化学习回放、边缘GPU服务与机器人本体部署共用同一份检查点与同一条推理路径,无须为每种场景单独编写推理程序。该框架支持NVFP4(NVIDIA 4位浮点格式)/INT4(4位整数)等低比特推理优化,并实现了DiT缓存、算子融合、视觉–语言主干与扩散动作专家分离部署等关键技术;模型适配层与运行时相互解耦,新架构只需新增一个适配器即可接入,调度、内存管理、算子选择与并行服务则由运行时统一提供。在Jetson Thor上,该框架将 $\pi 0.5$ 的

单请求时延由官方参考实现的179.5 ms压缩至107.7 ms(约9.3 Hz,加速1.67倍),已逼近操控所需的10 Hz下限;在消费级RTX 5090上进一步降至28.6 ms(约35 Hz),达到灵巧操控所需的实时控制频率区间。

2) mllm<sup>[27]</sup> (github.com/UbiquitousLearning/mllm): 面向移动端的多模态大模型推理引擎,以C/C++实现,不依赖重型运行时,支持W4A8、W8A8与W4A16等多档权重与激活量化方式。其突出优势在于后端覆盖广泛:除统一计算设备架构(CUDA)与OpenCL外,还支持高通Hexagon神经网络处理单元(NPU)、昇腾NPU与Arm CPU,这些正是其余框架尚未覆盖的端侧异构算力资源。在该引擎基础上进一步提出的llm.npu系统<sup>[32]</sup>,通过提示分块、离群值提取与块级乱序调度,化解了NPU对动态形状与激活离群值的不友好特性,将十亿级参数模型的预填充阶段提升至每秒超过1 000词元。目前,mllm尚未包含VLA/WAM模型,因此在表2中

表2 端侧具身智能推理系统在GPU上的单请求时延性能对比<sup>1</sup>

| 模型           | 设备          | LeRobot/ms | PhyAI/ms | FlashRT/ms | vla.cpp/ms | realtime–vla/ms | mllm/ms |
|--------------|-------------|------------|----------|------------|------------|-----------------|---------|
| $\pi 0$      | Jetson Thor | 321.0      | 121.5    | 61.0       | 267.8      | 116.5           | NA      |
|              | RTX 5090    | 127.5      | 31.5     | 22.2       | 61.6       | 32.7            | NA      |
| $\pi 0.5$    | Jetson Thor | 179.5      | 107.7    | 86.9       | 309.1      | 123.8           | NA      |
|              | RTX 5090    | 52.1       | 28.6     | 28.5       | 54.2       | 27.5            | NA      |
| GR00T N1.7   | Jetson Thor | 133.5      | 95.4     | 65.8       | 164.6      | NA              | NA      |
|              | RTX 5090    | 47.0       | 20.6     | 19.3       | 32.7       | NA              | NA      |
| Cosmos3–Nano | Jetson Thor | NA         | 8 121.4  | 8 027.5    | NA         | NA              | NA      |

GPU:图形处理器

表3 端侧具身智能推理系统对比<sup>2</sup>

| 系统                           | 支持模型       | 支持精度                               |  | 硬件后端                                        | 是否支持端云协同推理         |
|------------------------------|------------|------------------------------------|--|---------------------------------------------|--------------------|
| PhyAI <sup>[26]</sup>        | VLA + WAM  | BF16、FP8、INT8、NVFP4 / MXFP4 / INT4 |  | Jetson Thor、RTX 5090/4090、A40/A100、H20/H100 | 是(DP / TP / CFG并行) |
| mllm <sup>[27]</sup>         | 通用多模态大语言模型 | W4A8、W8A8、W4A16                    |  | Arm CPU、高通Hexagon NPU、昇腾NPU、OpenCL、CUDA     | 否                  |
| LeRobot <sup>[28]</sup>      | VLA + WAM  | FP32 / BF16                        |  | Jetson Thor、RTX5090/4090、A40/A100、H20/H100  | 是                  |
| vla.cpp <sup>[29]</sup>      | VLA        | W2A8                               |  | Jetson Orin Nano/Thor                       | 是                  |
| realtime–vla <sup>[30]</sup> | VLA        | BF16                               |  | Jetson Thor、RTX 4090/5090                   | 否                  |
| FlashRT <sup>[31]</sup>      | VLA + WAM  | BF16、FP8、NVFP4、INT8 / INT4         |  | Jetson Thor/Orin、RTX 5090/4090              | 否                  |

BF: BrainFloat16位浮点格式  
CFG: 无分类器引导  
CUDA: 统一计算设备架构

DP: 数据并行  
FP: 浮点数  
INT: 整数

MLLM: 多模态大语言模型  
MXFP: 微缩放浮点格式  
NPU: 神经网络处理单元

NVFP: 英伟达浮点格式  
RTX: 光线追踪极致纹理  
TP: 张量并行

VLA: 视觉–语言–动作模型  
WAM: 世界–动作模型

1 性能评测依据各项目官方仓库(截至2026年8月2日版本)本地测试所得;NA表示该运行时不支持该模型或无可用数据。

2 软件能力依据各项目论文及官方仓库截至2026年8月2日的公开状态核对。

其各项时延均记为不适用 (NA); 但其可作为承载 VLA 视觉-语言主干、并向 NPU 侧迁移的通用底座。

3) LeRobot<sup>[28]</sup> (github.com/huggingface/lerobot): HuggingFace 维护的端到端机器人学习开源框架, 面向真实机器人场景, 兼容低成本与边缘硬件, 提供检查点加载、数据处理、训练代码、参考实现行为以及通用的异步策略推理栈, 已集成 SmolVLA、基于 Transformer 的动作分块 (ACT)、DiT、 $\pi 0$  等可直接在端侧运行的模型。该框架的定位覆盖模型全生命周期并提供参考实现, 而非追求算子级的极致优化, 因此以 FP32/BF16 参考实现为主。本文实测其  $\pi 0.5$  推理路径在 Jetson Thor 上的时延为 179.5 ms (约 5.6 Hz), 在 RTX 5090 上为 52.1 ms (约 19.2 Hz)。

4) vla.cpp<sup>[29]</sup> (github.com/VinRobotics/vla.cpp): 首个 ggml/llama.cpp 级、原生支持流匹配与扩散动作专家的统一 C++ 运行时, 已覆盖多种 VLA 架构。该运行时支持视觉-语言前缀的键值跨求解复用, 避免在每个去噪步上重复计算前缀; 并采用三值 1.58 bit (W2A8) 权重量化, 以压缩访存与显存占用, 使其能够在 8 GB 显存的 Orin Nano 等紧显存设备上运行; 通过零消息队列 (ZMQ) 客户端/服务端形态支持端云分离部署。本文实测其  $\pi 0.5$  模型在 Jetson Thor 上的时延为 309.1 ms (约 3.2 Hz), 在 RTX 5090 上为 54.2 ms (约 18.4 Hz), 表明其广泛的可移植性以一定的时延为代价。

5) realtime-vla<sup>[30]</sup> (github.com/Dexmal/realtime-vla): 将  $\pi 0.5$  模型拆解为 24 个通用矩阵乘法 (GEMM) 类算子, 逐级施以 Triton 内核优化, 实现全流式推理, 目前已覆盖  $\pi 0$ 、 $\pi 0.5$  与 DM0 等模型。该框架不引入量化, 仅依靠 BF16 内核调优与流式调度获得加速效果, 因此精度与参考实现完全对齐。本文实测其  $\pi 0.5$  模型在 Jetson Thor 上的时延为 123.8 ms (约 8.1 Hz), 在 RTX 5090 上为 27.5 ms (约 36.4 Hz)。

6) FlashRT<sup>[31]</sup> (github.com/flashrt-project/FlashRT): 面向 VLA、WAM 等小批量、低时延负载的实时推理引擎, 支持 FP8、NVFP4 及 INT8/INT4 等多档低精度选项, 硬件后端覆盖 Jetson Thor、RTX 系列与 Jetson AGX Orin; 除 VLA 外, 亦已支持 Cosmos3 等 WAM 模型。本文实测其  $\pi 0.5$  模型在 Jetson Thor 上的时延为 86.9 ms (约 11.5 Hz), 在 RTX 5090 上为 28.5 ms (约 35.1 Hz)。

综合表 2 与表 3 可见, 6 个系统各有侧重。就推理速度而言, 专用运行时在特定模型与硬件组合上确实更快: FlashRT 在 Jetson Thor 上推理  $\pi 0.5$  与 GR00T N1.7 的时延分别为 86.9 ms 与 65.8 ms, realtime-vla 在 RTX 5090 上推理  $\pi 0.5$  的时延为 27.5 ms, 均略优于 PhyAI 的对应结果 (107.7 ms、95.4 ms 与 28.6 ms)。但专用运行时的适用面明显更窄:

realtime-vla 仅覆盖  $\pi$  系列与 DM0, 其与 vla.cpp 均不支持世界-动作模型 (WAM), 且二者连同 FlashRT 均未提供端云协同推理能力; mllm 虽后端覆盖范围最广, 却尚未包含任何具身模型。相比之下, 在本文测试的 6 个系统中, PhyAI 是唯一在全部 4 个模型上均完成了实测、同时覆盖 VLA 与 WAM 两类架构, 并支持以数据并行 (DP)/张量并行 (TP)/无分类器引导 (CFG) 并行方式进行云侧扩展的框架。其时延与最快的专用运行时处于同一梯队: 在 RTX 5090 上推理  $\pi 0.5$  时, PhyAI 仅比 FlashRT 慢 0.1 ms, 比 realtime-vla 慢 1.1 ms。因此, 在本文的测试范围内, PhyAI 在覆盖面与时延之间取得了较好的折中——它以一种代码路径与一份检查点统一覆盖端、边、云 3 类部署场景, 避免为不同环境重复实现推理程序, 同时将时延维持在专用运行时的量级水平。

## 4 未来展望

尽管前文已在算法与系统两个层面梳理出相当丰富的优化手段, 端侧具身智能距离实际部署仍有明显差距: 即便经过优化, 现有系统往往仍难以稳定达到操控所需的 10 Hz 控制频率, 更难以企及操作任务所期望的 30 Hz 甚至更高频率。本文实测结果亦印证了这一差距:  $\pi 0.5$  在 Jetson Thor 上的官方参考实现仅为 5.6 Hz, 经优化后也仅提升至 9.3 Hz; Cosmos3-Nano 则低至每秒 0.12 个动作分块, 且在消费级 RTX 5090 上因显存溢出而无法运行。两类模型的性能瓶颈各有不同: VLA 在机器人必须采用的单样本批处理下, 访存与启动开销成为主要瓶颈, 模型算力利用率在 Jetson Thor 上仅为 7.6%, 在 RTX 5090 上 GR00T N1.7 的动作头也仅为 13.2%; WAM 的生成模块在批大小为 1 时即呈现计算受限特征, 利用率高达 65.8%, 其时延主要由多步去噪的原始算力需求决定。前者需降低访存与启动开销, 后者需减少去噪所需的算力需求。图 4 展示了算法-系统-芯片协同优化的潜在研究方向。

1) 评测层: 构建面向具身的端到端性能评测指标体系。一切优化均以正确的衡量为前提, 而当前对具身大模型推理的评测仍主要沿用传统大模型的时延与能耗指标, 缺乏具身特定的视角。这类指标既未刻画控制频率与动作成功率之间的耦合关系, 也未反映感知-动作闭环条件下的实时性与能效约束, 因而难以有效指导面向操控任务的优化。本文认为, 亟需从具身角度定制测量方法与指标体系, 将控制频率、动作成功率、端到端时延与闭环能效进行联合考量; 唯有评测先行, 后续算法与系统层的优化方能具备明确的优化目标与可比的衡量基准。

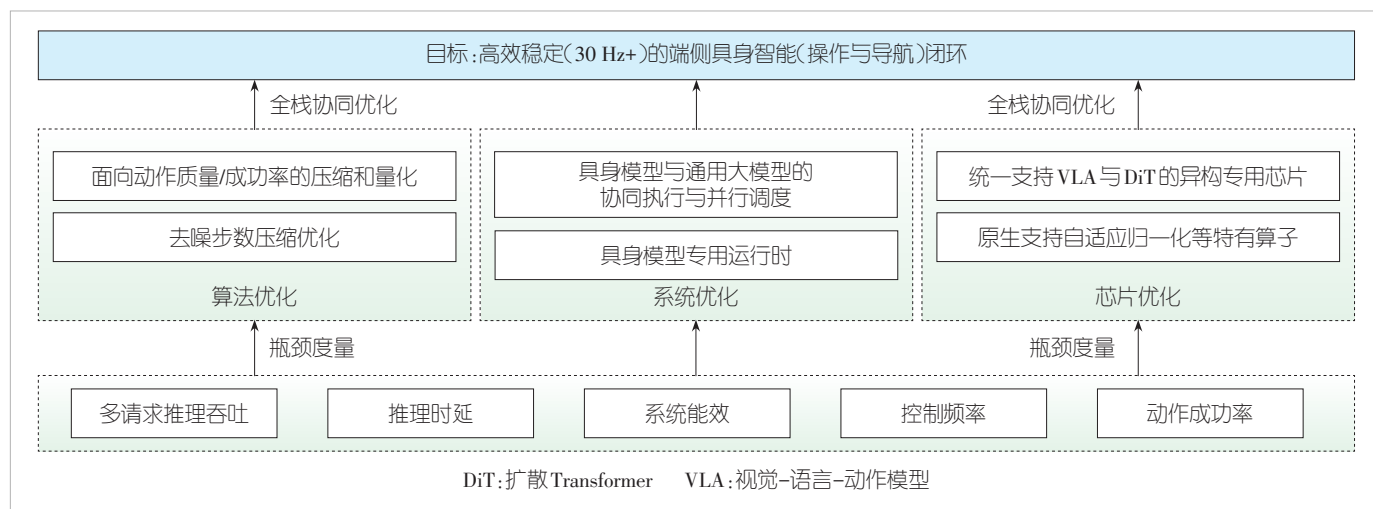


图4 算法-系统-芯片协同优化

2) 算法层: 以动作质量为目的的少步化与压缩。针对上述两类问题, 算法层的优化策略也应有所区分。对访存受限的 VLA 动作专家而言, 收益主要来自两方面: 一是将去噪步数压缩至单步, 二是通过低比特量化降低权重与键值缓存的访存量。对计算受限的 WAM 生成模块, 则应减少或简化去噪步数、压缩隐空间维度, 因为其时延随原始算力需求增长而增加。值得关注的是优化目标的转变: 扩散缓存、量化等技术虽对降低内存占用与提升速度具有重要意义, 但目前其在 VLA/WAM 上的应用仍较为有限, 且大多沿用最小量化误差或重建误差的通用目标。然而, 在具身场景下, 误差最小并不等同于操控最优; 更合理的做法是直接以动作成功率作为优化目标与评判准则, 使压缩与加速过程对动作负责, 而非仅对像素或张量负责。

3) 系统层: 专用运行时与多模型协同。当前, 专用具身推理运行时(如 vla.cpp、realtime-vla)尚处于起步阶段, 多数实机部署仍沿用并非为具身场景设计的通用大模型引擎, 缺乏对控制频率约束及感知-动作闭环特性的针对性优化。算法报告的加速效果与其在真实 CPU、GPU、NPU 上的实际执行效率之间仍存在明显差距, 需要通过算法-系统跨层协同设计来缩小这一差距。另一方面, 端侧具身设备上往往不止运行控制模型, 还并存全模态 (OMNI) 式的通用交互大模型, 而现有工作大多仅考虑单模型执行场景。控制模型与交互模型如何在有限的端侧资源上实现协同调度与并行执行, 是一个尚未被充分研究的关键系统问题。

4) 芯片层: 统一支持 VLA 与 DiT 的具身专用加速器。面向具身负载的专用芯片研究尚少, 其核心难点在于异构执行需求: VLA 中视觉-语言主干为计算受限型, 而扩散动作专家为访存受限型且需多步迭代, 二者执行模式差异显著。

如何使同一加速器同时高效支持这两种计算模式, 并原生支持自适应归一化等具身与扩散特有的算子, 是关键的结构问题。总体而言, 该层的研究仍比较有限, 但值得深入探索。

#### 参考文献

- [1] Wang Z D, Li Z S, Mandlekar A, et al. One-step diffusion policy: fast visuomotor policies via diffusion distillation [PP/OL]. arXiv (2024-10-28)[2026-04-09]. <http://arxiv.org/html/2410.21257v1>
- [2] Su Y F, Liu N, Chen D, et al. Freqpolicy: efficient flow-based visuomotor policy via frequency consistency [PP/OL]. arXiv(2026-02-18)[2026-04-08]. <http://arxiv.org/abs/2506.08822>
- [3] Høeg S H, Du Y, Egeland O. Streaming diffusion policy: fast policy synthesis with variable noise diffusion models [PP/OL]. arXiv (2024-06-07)[2026-04-09]. <http://arxiv.org/html/2406.04806v1>
- [4] Ji K Y, Meng Y, Cui H Y, et al. Block-wise adaptive caching for accelerating diffusion policy [PP/OL]. arXiv (2025-06-16) [2026-04-08]. <http://arxiv.org/abs/2506.13456>
- [5] Black K, Galliker M Y, Levine S, et al. Real-time execution of action chunking flow policies [PP/OL]. arXiv(2025-06-09) [2026-04-08]. <http://arxiv.org/abs/2506.07339>
- [6] Liu J M, Liu M Z, Wang Z Y, et al. Robomamba: efficient vision-language-action model for robotic reasoning and manipulation [PP/OL]. arXiv(2024-06-06) [2026-04-08]. <http://arxiv.org/abs/2406.04339>
- [7] Vasu P K A, Faghri F, Li C L, et al. FastVLM: efficient vision encoding for vision language models [PP/OL]. arXiv (2024-12-17) [2026-04-08]. <http://arxiv.org/abs/2412.13303>
- [8] Park S M, Kim H M, Kim S W, et al. Saliency-aware quantized imitation learning for efficient robotic control [PP/OL]. arXiv(2025-05-24)[2026-04-08]. <http://arxiv.org/abs/2505.15304>
- [9] Wang H Y, Xiong C Y, Wang R P, et al. BitVLA: 1-bit vision-language-action models for robotics manipulation [PP/OL]. arXiv (2025-06-10)[2026-04-08]. <http://arxiv.org/abs/2506.07530>
- [10] Liu Z Y, Chen Y Q, Cai H Y, et al. Bridging the semantic-action gap in visual token pruning for efficient VLA inference [PP/OL]. arXiv(2025-11-20) [2026-04-08]. <http://arxiv.org/abs/2511.16449>

- [11] Song W X, Chen J Y, Ding P X, et al. CEED-VLA: consistency vision-language-action model with early-exit decoding [PP/OL]. arXiv(2025-06-16) [2026-06-08]. <http://arxiv.org/abs/2506.13725>
- [12] Kim M J, Finn C, Liang P, et al. Fine-tuning vision-language-action models: optimizing speed and success [PP/OL]. arXiv(2025-02-27)[2026-04-08]. <http://arxiv.org/abs/2502.19645>
- [13] Pertsch K, Stachowicz K, Ichter B, et al. FAST: efficient action tokenization for vision-language-action models [PP/OL]. arXiv(2025-01-17)[2026-04-08]. <http://arxiv.org/abs/2501.09747>
- [14] Wang S S, Yu R C, Yuan Z H, et al. Spec-VLA: speculative decoding for vision-language-action models with relaxed acceptance [PP/OL]. arXiv(2025-07-29) [2026-04-08]. <http://arxiv.org/abs/2507.22424>
- [15] Niu J H, Gu K F, Zhao Y C, et al. Realtime-VLA FLASH: speculative inference framework for diffusion-based VLAs [PP/OL]. arXiv(2026-05-13) [2026-04-08]. <http://arxiv.org/abs/2605.13778>
- [16] Xu S Y, Wang Y K, Xia C H, et al. VLA-cache: efficient vision-language-action manipulation via adaptive token caching [PP/OL]. arXiv(2025-10-21) [2026-04-12]. <https://arxiv.org/abs/2502.02175>
- [17] Dai Y T, Gu H, Wang T, et al. ActionFlow: a pipelined action acceleration for vision language models on edge [PP/OL]. arXiv(2025-12-24)[2026-04-08]. <http://arxiv.org/abs/2512.20276>
- [18] Tian Y, Yang S Z, Zeng J, et al. Predictive inverse dynamics models are scalable learners for robotic manipulation [PP/OL]. arXiv(2024-12-19) [2026-04-08]. <http://arxiv.org/abs/2412.15109>
- [19] Zhang W Y, Liu H S, Qi Z K, et al. DreamVLA: a vision-language-action model dreamed with comprehensive world knowledge [PP/OL]. arXiv(2025-07-07) [2026-04-08]. <http://arxiv.org/abs/2507.04447>
- [20] Guo Y J, Hu Y C, Zhang J K, et al. Prediction with action: visual policy learning via joint denoising process [PP/OL]. arXiv(2024-11-27)[2026-04-08]. <http://arxiv.org/abs/2411.18179>
- [21] Cen J, Yu C H, Yuan H J, et al. WorldVLA: towards autoregressive action world model [PP/OL]. arXiv(2025-06-26) [2026-04-08]. <http://arxiv.org/abs/2506.21539>
- [22] Zhu C N, Yu R, Feng S Y, et al. Unified world models: coupling video and action diffusion for pretraining on large robotic datasets [PP/OL]. arXiv(2025-04-03) [2026-04-08]. <http://arxiv.org/abs/2504.02792>
- [23] Ye S H, Jang J, Jeon B G, et al. Latent action pretraining from videos [PP/OL]. arXiv(2024-10-15) [2026-04-08]. <http://arxiv.org/abs/2410.11704>
- [24] Zheng R J, Wang J, Reed S, et al. Flare: robot learning with implicit world modeling [PP/OL]. arXiv(2025-05-23) [2026-04-08]. <http://arxiv.org/abs/2505.15659>
- [25] Liu K H, Hu W B, Xu J L, et al. Rolling forcing: autoregressive long video diffusion in real time [PP/OL]. arXiv(2025-03-24) [2026-04-08]. <http://arxiv.org/abs/2503.18452>
- [26] Wang C H, Xu D L, Cai D Q, et al. PhyAI: real-time physical AI at the edge, scalable rollouts in the cloud [PP/OL]. arXiv(2026-04-08). <http://arxiv.org/abs/2608.03682>
- [27] Yi R J, Li X, Lu Z Y, et al. mllm: fast and lightweight multimodal LLM inference engine for mobile and edge devices [PP/OL]. arXiv(2023-08-30) [2026-04-08]. <https://github.com/UbiquitousLearning/mllm>
- [28] Cadene R, Aliberts S, Capuano F, et al. Lerobot: an open-source library for end-to-end robot learning [PP/OL]. arXiv(2025-02-08)[2026-04-08]. <http://arxiv.org/abs/2502.05637>
- [29] Nguyen K D, Ho H T, Nguyen C T, et al. vla.cpp: a unified inference runtime for vision-language-action models [PP/OL]. arXiv(2026-06-16) [2026-04-08]. <http://arxiv.org/abs/2606.08094>
- [30] Ma Y C, Zhou Y Z, Yang Y H, et al. Running VLAs at real-time speed [PP/OL]. arXiv(2025-10-31)[2026-04-08]. <http://arxiv.org/abs/2510.26742>
- [31] Su L. Execution-state capsules: graph-bound execution-state checkpoint and restore for low-latency, small-batch, on-device physical-AI serving [PP/OL]. arXiv(2026-06-30) [2026-04-08]. <http://arxiv.org/abs/2606.20537>
- [32] Xu D L, Yang L Y, Liu R Q, et al. Fast on-device LLM inference with NPUs [EB/OL]. (2025-08-06) [2026-04-10]. <http://dl.acm.org/doi/proceeding/10.1145/3676642>

## 作者简介



徐梦炜, 北京邮电大学副教授; 主要研究方向为端侧具身智能算法和系统; 先后主持和参与基金项目10余项, 获得2项科研成果奖; 发表论文50余篇。



徐大亮, 北京邮电大学副研究员; 研究方向为端侧大模型与具身智能系统; 先后主持和参与基金项目2项; 发表论文10余篇。