BitVLA：用于机器人操作的 1 比特视觉-语言-动作模型

Hongyu Wang Chuyan Xiong Ruiping Wang Xilin Chen
Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences

Abstract

视觉-语言-动作（VLA）模型在广泛的机器人操作任务中展示了令人印象深刻的能力。然而，它们日益增长的模型规模对资源受限的机器人系统的部署提出了重大挑战。虽然 1 位预训练已被证明能够以最小的性能损失有效提高大型语言模型的推理效率，但其在 VLA 模型中的应用仍未得到充分探索。在这项工作中，我们推出了 BitVLA ，这是第一个用于机器人操作的 1 位 VLA 模型，其中每个参数都是三元的，即 $\{-1, 0, 1\}$ 。为了进一步减少视觉编码器的内存占用，我们提出了蒸馏感知训练策略，将全精度编码器压缩到 1.58 位权重。在此过程中，一个全精度的编码器作为教师模型，以更好地对齐潜在表示。尽管缺乏大规模的机器人预训练，BitVLA 在 LIBERO 基准测试中实现了与当前最先进的模型 OpenVLA-OFT 相当的性能，使用 4 位后训练量化，同时仅消耗 29.8 % 的内存。这些结果突显了 BitVLA 在内存受限的边缘设备上部署的前景。我们在 `https://github.com/ustcwhy/BitVLA` 中发布了代码和模型权重。

1 介绍

近年来，视觉-语言模型（VLMs）取得了显著进展 **????** 。这些模型在视觉问答 **??** 、数学推理 **??** 和人机交互 **??** 等各种下游任务中取得了令人印象深刻的成果。基于这一进展，该领域正不断向视觉-语言-行动（VLA）模型发展，这些模型扩展了 VLMs 的模态，以纳入用于机器人控制的动作生成 **??????** 。这些模型旨在赋予机器人理解视觉环境、遵循自然语言指令并自主执行任务的能力。VLA 模型提供了一个统一的框架来连接感知、语言理解和运动控制，使其成为体现人工智能的一个有前景的范式。

然而，在现实世界的机器人系统中部署如此大规模的 VLA 模型仍然非常具有挑战性，尤其是在资源受限的边缘设备上。这些系统通常在内存、计算吞吐量和能量可用性方面受到限制。最近在模型量化方面的努力表明，减少模型权重和激活值的位宽可以显著提高效率。特别是，每个参数都限制为三值（即 $\{-1, 0, 1\}$ ）的 1 位大型语言模型（LLMs） **???** ，已成为一种引人注目的解决方案。这些模型在各种 NLP 基准上实现了有竞争力的性能，同时显著减少了内存占用、能量消耗和推理延迟。此外，三值参数空间能够实现高效的硬件执行，并且可以简化在边缘加速器上的部署。尽管它们很有前途，但现有的 1 位模型大多局限于语言领域。据我们所知，它们向多模态任务和机器人控制的扩展尚未得到充分探索。

在这项工作中，我们引入了 BitVLA ，这是第一个用于机器人操作的 1-bit 视觉-语言-动作模型，其中每个参数是三元的，即 $\{-1, 0, 1\}$ 。BitVLA 建立在公开可用的 1-bit 大模型 BitNet b1.58 2B4T **?** 基础之上。我们首先使用 1-bit 大模型结合全精度视觉编码器训练视觉-语言模型，遵循 LLaVA **?** 的训练范式。为了进一步减少内存占

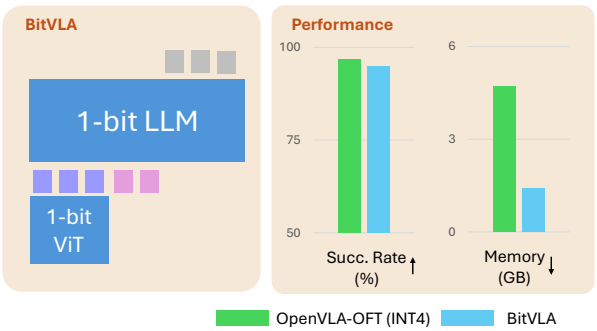

Figure 1: 在终端任务性能和内存占用方面，比较 BitVLA 和 OpenVLA-OFT 的 4 位后训练量化。我们报告在 LIBERO 基准测试上的平均成功率。

Preprint.

用，我们引入蒸馏感知训练，将视觉编码器量化为 1.58-bit 权重和 8-bit 激活。在此阶段，我们只训练模型的视觉编码器，使用全精度编码器作为教师模型以更好地对齐潜在表示。尽管没有进行大规模的机器人预训练，但如图 1 所示，BitVLA 在性能上可与最先进的模型 OpenVLA-OFT **?** 相媲美，其采用 4-bit 后训练量化，同时仅使用 29.8% 的内存空间。这些结果表明，BitVLA 提供了一种具有成本效益的高性能机器人操作解决方案，使其对于内存受限的机器人系统是可行的。

受 VLMs 快速进展的启发，机器人领域的研究人员开始探索直接生成低级控制信号的 VLA 模型。RT 系列引入了 Open X-Embodiment (OXE)，一个大规模标准化的机器人数据集，并利用它训练 RT-X，一个用于机器人操作任务的通用模型。OpenVLA 详细讨论了 VLA 的设计，从预训练架构到参数高效的微调方法和部署策略，全面开源了各个阶段的训练方法和预训练模型。RoboFlamingo 利用预训练的 VLMs 进行单步视觉语言推理，引入了一个策略头以捕捉顺序历史，并通过模仿学习进行了最少量的微调。OpenVLA-OFT 通过建模连续动作、采用并行解码和应用模仿学习中的动作分块来优化微调过程。为了提高推理效率，TinyVLA 采用了紧凑的 1.3B VLM 骨干网，并跳过预训练以提高数据效率。最近，NORA 通过利用 Qwen2.5-VL-3B 作为骨干，并通过 FAST+ 标记器增强动作生成，展示了具有竞争力的性能。

原生 1 位模型。 现代深度学习研究越来越关注量化感知训练和低精度推理 **???** 。最近的研究 **??????** 展示了对大型语言模型进行 1 位和 1.58 位预训练的潜力。**?** 经验证明，当参数数量增加时，1 位和全精度模型之间的性能差距会缩小。此外，BitNet b1.58 **?** 表明，1.58 位的大型语言模型可以从 3B 规模开始达到全精度模型的性能，同时大幅度减少内存占用、解码延迟和能量消耗方面的推理成本。OneBit **?** 进一步探讨了知识蒸馏在训练二值化大型语言模型中的应用。bitnet.cpp **?** 开发了一种针对 1 位大型语言模型优化的推理系统，大幅降低了能量消耗并提高了 CPU 设备上的解码延迟。最近，**?** 训练了一个 2B 参数的三值化大型语言模型，在总体性能上达到了与领先的开源权重大型语言模型相当的竞争力。1 位大型语言模型的低内存和能源需求使得它们在边缘应用中尤其具有吸引力，尤其是在机器人任务中。然而，据我们所知，1 位模型扩展到视觉-语言和视觉-语言-动作训练的应用仍然很少被探索。

2 BitVLA： 1-bit VLA

在这一节中，我们首先在第 2.1 节介绍 BitVLA 的模型架构。接下来，我们在第 2.2 节详细介绍蒸馏感知训练策略的目标，该策略旨在将全精度编码器权重压缩到 1.58 比特。最后，我们在第 2.3 节描述 BitVLA 在下游任务上的微调过程。

2.1 模型架构

BitVLA 采用 BitNet b1.58 2B4T **?** 作为 LLM 的主干网络，并使用 SigLIP-L **?** 作为视觉编码器。我们使用了在 224×224 分辨率图像上预训练的 SigLIP-L 版本，从而得到较短的视觉令牌序列。该选择在对性能影响最小的情况下提高了计算效率 **?** 。我们使用一个具有 GeLU 激活函数的两层 MLP 作为连接器，由于其对整体模型大小的贡献可以忽略不计，因此保持全精度。

图 2 提供了 BitVLA 的训练概览。在遵循 LLaVA **?** 的训练策略时，我们首先使用 1-bit LLM 和全精度视觉编码器来训练 VLM。在第一阶段，只有连接器在一个小的图像描述数据集上进行训练，以便将 LLM 与视觉编码器对齐。接下来是一个视觉指令微调阶段，在此阶段中视觉编码器被冻结，而其余组件会进行更新。最后，我们进行蒸馏感知微调，将视觉编码器量化为低位表示。

对于量化，我们采用绝对均值量化器用于权重，采用每标记绝对最大量化器用于激活 **?** 。权重被量化为三值（即，$\{-1, 0, 1\}$ ），而激活被量化为对称 INT8（即，$[-128, 127]$ ）。具体来说，量化可以被公式化为：

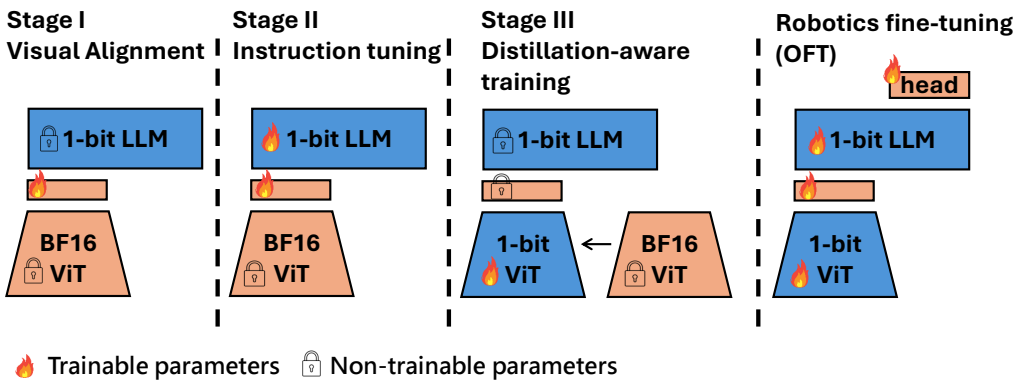

Figure 2: BitVLA 的训练概述。我们首先使用一个 1-bit LLM ? 结合全精度视觉编码器来训练一个视觉-语言模型。然后，我们应用蒸馏感知训练将视觉编码器的权重量化为 1.58 位精度。最后，通过 OFT 微调 ?，将 BitVLA 适配于特定的机器人任务。

$$Q_w(W) = \alpha \cdot \text{RoundClip}(\frac{W}{\alpha}, -1, 1),\ \alpha = \frac{1}{nm}||W||_1 \tag{1}$$

$$Q_a(x) = \frac{\beta}{127} \cdot \text{RoundClip}(\frac{127x}{\beta}, -128, 127),\ \beta = ||x||_\infty \tag{2}$$

$$\text{RoundClip}(x, a, b) = \max(a, \min(b, \text{round}(x))) \tag{3}$$

其中 $W \in \mathcal{R}^{m\times n}$ 表示线性层的可学习权重，$x \in R^{n\times 1}$ 表示输入。三值线性层的输出被计算为 $Y = Q_w(W)Q_a(x)$，其中 Q_w 和 Q_a 分别表示权重和激活的量化函数。

我们对视觉编码器中的所有线性层应用量化，但不包括输入和输出嵌入层。BitVLA 通过量化感知训练进行训练，在前向传播过程中实时进行量化。由于量化操作的不可微性，我们采用直通估计器 (STE) ? 来近似反向传播过程中梯度。具体而言，梯度直接通过量化函数传递，遵循如下近似：

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial Q_w(W)}, \quad \frac{\partial \mathcal{L}}{\partial X} = \frac{\partial \mathcal{L}}{\partial Q_a(X)} \tag{4}$$

为了保持训练的稳定性，梯度和优化器状态都保持全精度。

2.2 蒸馏感知训练

在本小节中，我们引入了蒸馏感知训练，以有效地将 VLM 的视觉编码器量化为 1.58 位宽度。我们在图 3 中展示了概述。我们首先从全精度模型初始化 1.58 位编码器的潜在权重。然后，我们采用全精度编码器作为教师模型。训练目标 $\mathcal{L}_{total}$ 需要同时最小化特定任务的损失 $\mathcal{L}_{\text{LM}}$ 以及全精度编码器和 1.58 位编码器之间的潜在表示的辅助对齐损失 $\mathcal{L}_{\text{aux}}$。

自回归语言建模损失 $\mathcal{L}_{\text{LM}}$ 被广泛用于训练 VLMs。设 $\mathcal{T}$ 表示输入文本序列，该序列分为指令部分 $\mathcal{T}_{\text{ins}}$ 和响应部分 $\mathcal{T}_{\text{ans}}$。由 1.58-bit 视觉编码器提取的视觉标记表示为 $\mathcal{V}_{\text{1.58-bit}}$。语言建模损失可以表示为：

$$\mathcal{L}_{\text{LM}} = -\sum_{\text{token}_i \in \mathcal{T}_{\text{ans}}} \log \Pr(\mathcal{Y}^i \mid \mathcal{V}_{\text{1.58-bit}}, \mathcal{T}^{[:i-1]})$$

其中，$\mathcal{Y}^i$ 表示模型在位置 i 预测的标记。损失仅在响应标记 $\mathcal{T}_{\text{ans}}$ 上计算，而指令和视觉标记作为上下文提供。

为了增强 1.58 位和全精度视觉编码器的潜在表示之间的对齐，我们通过知识蒸馏学习 1.58 位编码器，其中全精度编码器用作教师模型。令 h^l_{bf16} 和 $h^l_{\text{1.58-bit}}$ 分别表示来自全精度和 1.58 位视觉编码器的第 l 层的输出。对齐损失定义为：

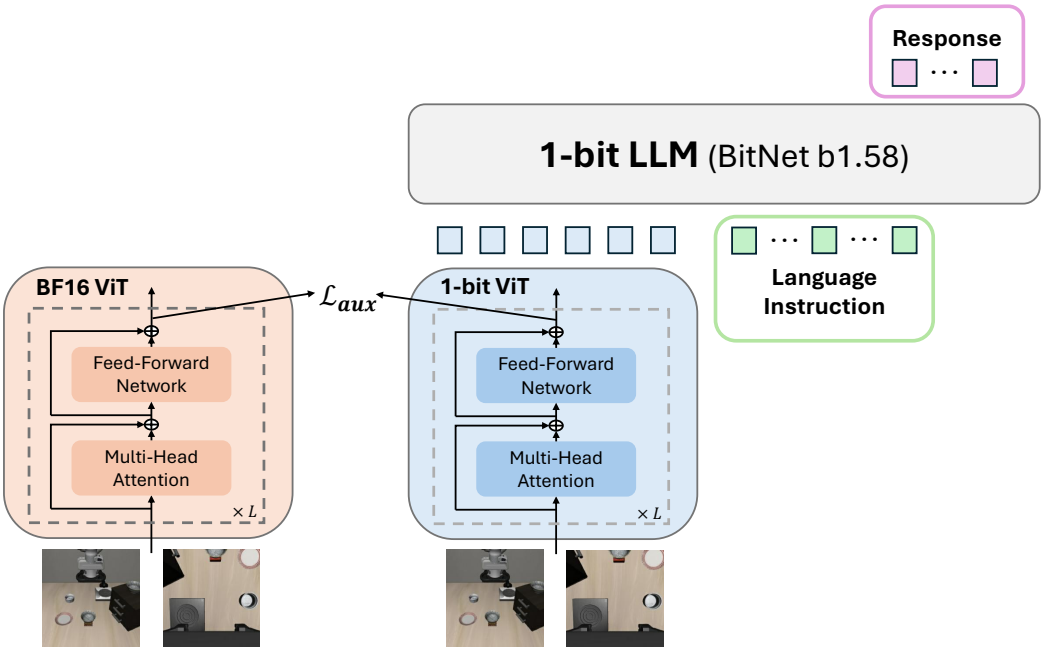

Figure 3: 蒸馏感知训练的概述。原始的全精度编码器作为教师模型以确保潜在表示的更好对齐。

$$\mathcal{L}_{\text{aux}} = \frac{1}{n} \sum_{l=1}^{L} \left\| h^{l}_{\text{bf16}} - h^{l}_{\text{1.58-bit}} \right\|^2$$

其中 n 是隐层维度，L 是视觉编码器中的总层数。这个辅助损失鼓励 1.58 位视觉编码器模仿其全精度对应物的表示行为。

首先，训练目标 $\mathcal{L}_{\text{total}}$ 是：

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \gamma \cdot \mathcal{L}_{\text{aux}} \tag{5}$$

其中 γ 是用于表示对齐的系数。在蒸馏感知训练期间，只有视觉编码器是可训练的，而其他组件（即 LLM 和连接器）是冻结的。在我们的实验中，我们观察到，与 LLMs 的 1.58 位预训练不同，1.58 位编码器的量化感知训练在从全精度教师模型蒸馏时具有高度的数据效率。它在只使用数十亿个训练标记的情况下保留了大部分全精度对应模型的性能。

2.3 机器人微调

在这一小节中，我们描述了 BitVLA 针对特定机器人任务的微调过程。遵循 OpenVLA-OFT **?**，我们利用并行解码和动作分块技术来提升 VLA 模型的吞吐量。具体而言，我们用双向注意力掩码替换了 LLM 中使用的传统因果掩码，使得每次前向传递能够在多个时间步内生成连贯的动作轨迹。与自回归的逐个令牌预测相比，这种方法显著提高了实时控制效率。此外，我们集成了基于 MLP 的动作头来将查询令牌的潜在表示投射到连续的机器人动作空间。模型训练的目标是最小化预测动作与真实轨迹之间的 L_1 损失。

3 实验

3.1 模型训练

BitVLA 的训练分为三个阶段。参考 LLaVA **?**，首先，我们训练连接器，使视觉编码器与 LLM 在 LLaVA 1.5-558k 数据集 **?** 上对齐。在第二阶段，我们冻结视觉编码器，并在由单图

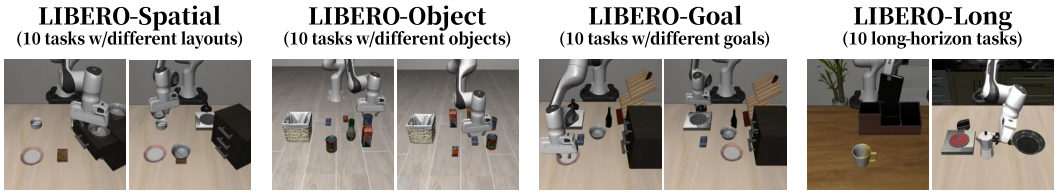

Figure 4: LIBERO 基准任务套件的概述。它有四个不同的维度来评估机器人操作模型的泛化和性能。

Table 1: BitVLA 和基准在 LIBERO 模拟环境中的成功率 (%)。

Models	Size	Memory Usage ↓	Spatial	Object	Goal	Long	Avg.
w/ Robotics pre-training							
OpenVLA ?	7.5B	15.1GB (10.79×)	84.7	88.4	79.2	53.7	76.5
SpatialVLA ?	4.2B	8.5GB (6.07×)	88.2	89.9	78.6	55.5	78.1
CoT-VLA ?	8.0B	16.2GB (11.57×)	87.5	91.6	87.6	69.0	81.1
NORA-Long ?	3.8B	7.5GB (5.36×)	92.2	95.4	89.4	74.6	87.9
π_0 ?	3.5B	7.0GB (5.00×)	96.8	98.8	95.8	85.2	94.2
OpenVLA-OFT ?	7.7B	15.4GB (11.00×)	97.6	98.4	97.9	94.5	97.1
w/o Robotics pre-training							
OpenVLA-OFT ?	7.7B	15.4GB (11.00×)	94.3	95.2	91.7	86.5	91.9
BitVLA (ours)	3.0B	1.4GB (1.00×)	97.4	99.6	94.4	87.6	94.8

像样本组成的 MammoTH-VL ? 的 1000 万个样本子集上训练 LLM 和连接器。在最后阶段，我们从全精度 (W16A16) 开始训练视觉编码器，至 1.58-bit 权重和 8-bit 激活 (W1.58A8)，训练来自第二阶段数据的 500 万个样本子集。阶段三的训练数据包含最多 100 亿个标记。潜在表示上的蒸馏损失加权系数为 $\gamma = 0.1$ 。根据 ? 的建议，我们对指令调整使用大学习率。在 8 张 80GB 内存的 NVIDIA A100 卡上，训练需要 14 天。我们在附录 ?? 中提供了详细的超参数配置。

3.2 机器人操作实验

基准。(或基准测试) 我们采用 LIBERO 模拟环境 ? 来评估机器人操控模型的泛化能力和性能。如图 4 所示，该基准测试从四个关键方面评估机器人智能：空间泛化（操控以新配置排列的物体）、物体泛化（适应以前未见过的物体类别）、目标泛化（解读多样化的语言指令）和长期推理（执行涉及各种物体、布局和目标的多阶段任务）。这些能力通过四个相应的任务套件系统地进行评估，分别是 LIBERO-Spatial、LIBERO-Object、LIBERO-Goal 和 LIBERO-Long。每个任务套件包含 500 个专家演示，系统地分布在 10 个不同的操控任务中。更多细节列在附录 6 中。

我们在微调过程中使用与 OpenVLA-OFT 相同的训练数据集 [1] 。我们处理来自手腕安装和外部摄像头的同步多视角视觉输入，同时编码本体感受信号，例如末端执行器的位置。物理状态测量通过一个基于 MLP 的投影器投射到一个单一的 token 中，然后附加到图像 token 上。对于动作分块，我们根据 OpenVLA-OFT 设置块大小为 $K = 8$ ，并在重新规划之前执行完整的块。

我们对所有实验进行全参数微调以加快收敛速度。具体而言，对 LIBERO-Spatial、LIBERO-Object 和 LIBERO-Goal 进行 10k 步的微调，对 LIBERO-Long 进行 100k 步的微调。我们采用余弦退火学习率调度，批量大小为 64。在 8 个具有 80GB 内存的 NVIDIA A100 显卡上，10k 步微调过程大约需要 4 小时。更多细节可以在附录 ?? 中找到。

我们在 LIBERO 数据集上比较 BitVLA 与监督微调下的基线，包括 OpenVLA-OFT ? 、OpenVLA ? 、SpatialVLA ? 、CoT-VLA ? 、NORA-Long ? 和 π_0 ? 。具体来说，π_0 采用了基于预训练视觉语言模型（VLM）的流匹配架构。NORA 从强大的轻量级视觉语言模型 Qwen2.5-VL-3B ? 训练，以提高效率。我们采用其 NORA-Long 变体，每次生成五步动作序列。CoT-VLA 通过在生成动作前自回归预测未来帧引入视觉连锁思维推理。SpatialVLA 结

[1] https://huggingface.co/datasets/openvla/modified_libero_rlds

Table 2: BitVLA 和 OpenVLA、OpenVLA-OFT 在 LIBERO 模拟环境中进行训练后量化的成功率 (%) 。

Models	Memory Usage ↓	Spatial	Object	Goal	Long	Average
INT8 post-training quantization						
OpenVLA ?	7.4GB (5.29×)	86.4	85.2	77.2	58.8	76.9
OpenVLA-OFT ?	7.7GB (5.50×)	98.8	98.0	96.6	94.0	96.7
INT4 post-training quantization						
OpenVLA ?	4.4GB (3.14×)	83.0	84.0	72.0	51.6	72.7
OpenVLA-OFT ?	4.7GB (3.36×)	98.2	98.2	97.2	93.8	96.9
BitVLA (ours)	1.4GB (1.00×)	97.4	99.6	94.4	87.6	94.8

Table 3: 在视觉问答任务中，BitVLA 的全精度和 1.58 位视觉编码器（VE）的零样本准确率。

Models	MMMU (val)	SeedBench (image)	SeedBench $^{2+}$ (test)	MMStar (test)	AI2D (test)	Avg.
BitVLA w/ 16-bit VE	37.4	70.6	45.0	43.6	68.6	53.0
BitVLA w/ 1.58-bit VE	35.4	69.3	43.7	41.5	67.6	51.5

合 3D 信息，学习一个通用操控策略。OpenVLA 是在 OXE 数据集上训练的 7B 开源视觉语言动作模型，在许多任务上超过了闭源模型如 RT-2-X ? 。OpenVLA-OFT 从 OpenVLA 微调，使用了一系列技术，例如并行解码和连续动作建模，以提高特定任务的速度和性能。由于资源限制，BitVLA 没有在大规模机器人数据集上预训练。因此，我们也报告直接从其基础视觉语言模型 ? 微调的 OpenVLA-OFT 结果以供参考。

主要结果。 表 1 总结了在 LIBERO 基准套件中 BitVLA 和各种基准的成功率。正如表中所示，尽管 BitVLA 并未在大规模机器人数据集上进行预训练（例如，Open X-Embodiment ? ），但它仍然超越了拥有 30 亿参数的强基准，包括 π_0 和 NORA-Long。特别是，BitVLA 在 LIBERO-Long 上的表现比 π_0 高出 2.4 % ，突出了它在机器人操作的长时间推理任务中的有效性。此外，BitVLA 的内存占用仅为 1.4GB，使得可以在一个单消费级 GPU 上进行部署，例如 NVIDIA GeForce RTX 3050 Ti 笔记本电脑（4GB）。

与较大的 OpenVLA-OFT 模型相比，BitVLA 在 LIBERO 基准的空间、对象和目标子集上表现相当，但在 LIBERO-Long 上仍有不足。我们将这一差距归因于 OpenVLA-OFT 从 OpenVLA 进行的微调，它受益于大规模机器人学预训练，因此在复杂的操控任务中表现出色。如表所示，在大规模机器人学数据集上的预训练将 OpenVLA-OFT 在 LIBERO-Long 上的成功率从 86.5 % 提高到 94.5 % 。值得注意的是，与未进行机器人学预训练的 OpenVLA-OFT 变体相比，BitVLA 在 LIBERO-Long 上取得了相当的表现。

我们将 BitVLA 与 OpenVLA 和 OpenVLA-OFT 模型在 8 位和 4 位后训练量化情况下进行比较。我们使用它们在 Hugging Face 上公开发布的微调检查点。对于量化，我们利用 bitsandbytes 工具包 ? 将模型骨干转换为 INT8 和 INT4 精度。我们在 LIBERO 基准上报告量化模型的内存占用和性能。如表 2 所示，OpenVLA 在 4 位量化下表现出相比 OpenVLA-OFT 更大的性能下降。值得注意的是，BitVLA 在使用不到三分之一的内存的情况下，实现了与 4 位量化的 OpenVLA-OFT 相当的性能。

3.3 视觉问答实验

我们在视觉问答（VQA）任务中评估 BitVLA 的零样本性能，包括使用全精度和 1.58 位视觉编码器。评估套件包括 MMMU ? 、SeedBench ? 、SeedBench-2-Plus ? 、MMStar ? 和 AI2D ? 。我们采用公开可用的 LMM-Eval 工具包 ? 来确保公平和一致的比较。如表 3 所示，配备 1.58 位编码器的 BitVLA 实现了与其全精度版本相当的性能。具体来说，在五个基准上，1.58 位编码器平均准确率仅下降 1.5 %，同时其内存占用从 0.8GB 减少到 0.1GB。这些结果表明，考虑蒸馏的训练能够有效地保留通用 VQA 任务上的性能，同时在推理过程中显著降低内存消耗。

Table 4: 关于视觉问答任务中考虑蒸馏的训练的数据大小和表示对齐损失的消融实验。

Training Tokens (Stage III)	$\mathcal{L}_{aux}$	MMMU (val)	SeedBench (image)	SeedBench $^{2+}$ (test)	MMStar (test)	AI2D (test)	Avg.
10B	✓	**35.4**	**69.3**	**43.7**	**41.5**	**67.6**	**51.5**
5B	✓	33.3	69.1	43.3	41.4	66.4	50.8
5B	✗	32.4	52.9	38.8	30.7	57.5	42.4

Table 5: 在 LIBERO 基准套件上进行的关于蒸馏感知训练的数据规模消融和表示对齐损失。

Training Tokens (Stage III)	$\mathcal{L}_{aux}$	Spatial	Object	Goal	Long	Average
10B	✓	**97.4**	**99.6**	**94.4**	**87.6**	**94.8**
5B	✓	96.8	98.6	93.8	85.2	93.6
5B	✗	96.2	98.6	91.4	85.2	92.9

3.4 消融研究

表示对齐损失 我们进行消融研究，以评估在蒸馏感知训练中所提出的表示对齐损失的影响。如表 4 所示，结合对齐损失显著提升了 1.58-bit 视觉编码器的 BitVLA 的零样本性能，使在五个 VQA 基准上的平均准确率从 42.4 % 提升到 50.8 %。在 LIBERO 基准套件中，模型针对特定任务进行了微调，性能提升较小，但仍然有意义。如表 5 所报告，对齐损失在 LIBERO-Goal 集上增加了 2.4 % 的性能。

我们比较了在蒸馏感知训练（阶段 III）中用 50 亿和 100 亿标记训练的 BitVLA 的性能。如表 4 所示，在视觉编码器的量化感知训练中增加训练数据提高了在一般 VQA 任务中的整体性能。具体而言，阶段 III 中用 100 亿标记训练的 BitVLA 的平均准确率较用 50 亿标记训练的同类产品高出 0.7 %。此外，在 LIBERO 基准上，经过微调后，100 亿标记模型的平均准确率提高了 1.2 %。

4 定性分析

在本节中，我们仔细分析了 BitVLA 在 LIBERO 基准套件上的失败案例，将它们分为三类：空间定位差异、目标误解和轨迹规划失败。

- 空间定位误差是指在操作过程中由于姿势预测的不准确而导致的失败。通常情况下，这类错误发生在四个主要情境中：(1) 操作具有不稳定重心的物体，比如葡萄酒瓶，这时小的误算会导致不稳定 (2) 选择不精确的抓取姿势，导致物体在接近时倾倒或在运输过程中掉落 (3) 在没有实际物体的情况下尝试虚拟操作 (4) 在目标位置放置物体时出现定位错误，导致任务失败。这些问题可能源于视觉编码器提供的粗略空间理解，或在某些操作条件下相对于视觉信息过于依赖本体感受信号。BitVLA
- 目标误解是指由于 BitVLA 对语言指令的错误解释或遵循而导致的失败。这种错误的典型场景是，机器人在任务执行过程中错误地与非目标对象交互，随后启动与接触对象相关的另一个任务展开。我们假设这主要是由于在目标切换的瞬间，模型推理过程中视觉和本体感觉信息的优势造成的。这种不匹配在 OpenVLA-OFT **?** 中也有提到，该文提出了 OpenVLA-OFT+ 增强变体，通过使用 FiLM 策略来缓解这一问题。
- 轨迹规划失败是指在运动规划过程中由于碰撞导致的执行错误。一个典型的案例是机械臂在放置碗时与打开的抽屉的下面板发生碰撞，导致碗掉落或机械臂卡住。这些失败突显了 BitVLA 需要更好地利用先验知识以生成更合理且无碰撞的轨迹。此外，系统在执行较早的子目标（例如打开抽屉）时，应预见到后续子目标（例如放置碗）的可行性，以避免操作上的冲突。例如，部分打开抽屉可能会降低轨迹复杂性并降低碰撞风险。

我们在图 **??** 中展示了每种故障类型的分布情况和具体示例。在所有任务套件中最频繁的故障类型是空间定位差异。我们发现，LIBERO 的成功标准非常严格，尤其是在放置任务中，只有将物体精确放置在盘子中心时才被认为成功。然而，在许多长套任务的失败案例中，物

体虽然成功放置在盘子上，但由于未在中心位置仍然被判定为失败。然而，这一类别的大量错误表明灵巧的操作任务仍然是 BitVLA 的最大瓶颈。

5 结论

我们提出了 BitVLA ，这是首个用于机器人操控的 1-bit 视觉-语言-动作模型，其每个参数都被限制为三元值。BitVLA 最初使用 1-bit 大语言模型和全精度视觉编码器进行训练。为了进一步压缩视觉编码器，我们引入了一种考虑蒸馏的训练策略，将其权重转换为三元值。在这一阶段，全精度编码器作为教师模型，以引导潜在表示的对齐。在 LIBERO 基准测试上的实验结果表明，BitVLA 的性能与采用 4-bit 后置量化的最新 OpenVLA-OFT 模型相当，同时将内存使用量最多减少 3.36 × 。这些结果突出显示了 BitVLA 作为在内存受限硬件上应用机器人的一种具有成本效益且高效的解决方案。

我们在表 6 中展示了用于训练 BitVLA 的超参数配置。根据 **?** 的建议，我们在视觉指令调优期间采用了双阶段的权重衰减计划。对于在 LIBERO-Spatial、LIBERO-Object 和 LIBERO-Goal 套件上的微调，我们报告从学习率集合 5e-5，1e-4，3e-4 中选择的最佳结果。对于 LIBERO-Long，所有模型的训练在视觉编码器和 LLM 分别达到 8e-5 和 4e-4 的最高学习率。

Table 6: BitVLA 训练的超参数。

Hyper-parameter	**Stage I**	**Stage II**	**Stage III**
Peak Learning rate	1e-3	3e-4	1e-4
Batch Size	256	256	256
Weight decay	✗	$0.1 \rightarrow 0$	0.01
Trainable modules	Connector	LLM, Connector	ViT
Training steps	25k	40k	20k
Training sequence	1024	2048	2048
Vision sequence		256	
Learning rate scheduling		polynomial decay	
AdamW β		(0.9, 0.999)	
AdamW ϵ		1e-8	
Gradient Clipping		1.0	
Dropout		✗	
Attention Dropout		✗	

Table 7: 在 LIBERO 数据集上微调 BitVLA 的超参数。

Hyper-parameter	**Spatial**	**Object**	**Goal**	**Long**
Peak Learning rate	{ 5e-5, 1e-4, 3e-4 }			4e-4,8e-5
Training steps	10k	10k	10k	100k
Learning rate scheduling	cosine decay			
Warmup steps	375			
Batch Size	64			
Weight decay	0.01			
Trainable modules	LLM, Connector, ViT			
AdamW β	(0.9, 0.999)			
AdamW ϵ	1e-8			
Gradient Clipping	✗			

6 LIBERO 中的任务

在本节中，我们介绍了 LIBERO 中每个任务组的详细任务构成。如表 8 所示，它展示了 LIBERO 框架中四个任务组的不同任务配置。图 5 展示了部分任务的场景可视化。

Table 8: LIBERO 基准任务套件中的任务描述。

Task suite	Task description
Spatial	pick up the black bowl between the plate and the ramekin and place it on the plate pick up the black bowl next to the ramekin and place it on the plate pick up the black bowl from table center and place it on the plate pick up the black bowl on the cookie box and place it on the plate pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate pick up the black bowl on the ramekin and place it on the plate pick up the black bowl next to the cookie box and place it on the plate pick up the black bowl on the stove and place it on the plate pick up the black bowl next to the plate and place it on the plate pick up the black bowl on the wooden cabinet and place it on the plate
Object	pick up the alphabet soup and place it in the basket pick up the cream cheese and place it in the basket pick up the salad dressing and place it in the basket pick up the bbq sauce and place it in the basket pick up the ketchup and place it in the basket pick up the tomato sauce and place it in the basket pick up the butter and place it in the basket pick up the milk and place it in the basket pick up the chocolate pudding and place it in the basket pick up the orange juice and place it in the basket
Goal	open the middle drawer of the cabinet put the bowl on the stove put the wine bottle on top of the cabinet open the top drawer and put the bowl inside put the bowl on top of the cabinet push the plate to the front of the stove put the cream cheese in the bowl turn on the stove put the bowl on the plate put the wine bottle on the rack
Long	put both the alphabet soup and the tomato sauce in the basket put both the cream cheese box and the butter in the basket turn on the stove and put the moka pot on it put the black bowl in the bottom drawer of the cabinet and close it put the white mug on the left plate and put the yellow and white mug on the right plate pick up the book and place it in the back compartment of the caddy put the white mug on the plate and put the chocolate pudding to the right of the plate put both the alphabet soup and the cream cheese box in the basket put both moka pots on the stove put the yellow and white mug in the microwave and close it

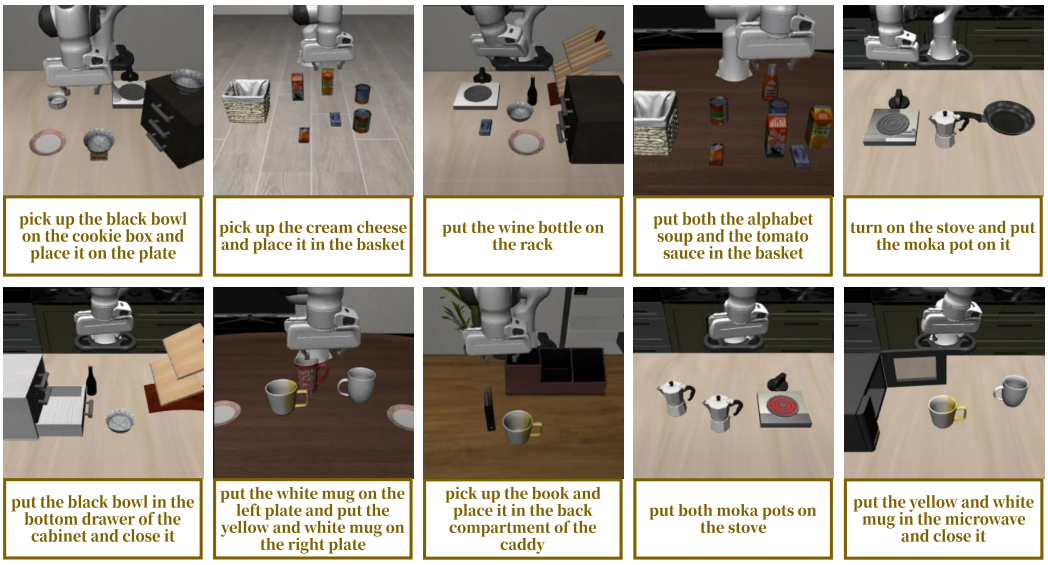

Figure 5: LIBERO 基准任务套件中的示例。