POLICYFLOW：强化学习中基于连续归一化流的策略优化

ShunpengYang[1,4]**, BenLiu**[2] **与 Hua Chen**[3], [4]†[1]香港科技大学， [2]南方科技大学[3]浙江大学-伊利诺伊大学厄巴纳-香槟分校联合学院， [4]LimX Dynamics

† 通讯作者，huachen@intl.zju.edu.cn

arXiv:2602.01156v1 [cs.LG] 1 Feb 2026

摘要

在在策略强化学习算法中，近端策略优化（PPO）因其简洁性、数值稳定性以及出色的实证表现而广受青睐。标准 PPO 依赖于通过重要性比率定义的代理目标，这需要评估策略似然；当策略被建模为高斯分布时，该评估通常较为直接。然而，将 PPO 扩展到更具表达能力的高容量策略模型（如连续归一化流（CNF），亦称流匹配模型）颇具挑战，因为沿完整流轨迹进行似然评估计算开销高昂，且往往数值不稳定。为解决这一问题，我们提出PolicyFlow，一种新颖的基于连续归一化流的在策略强化学习算法，它将富有表达力的连续归一化流策略与 PPO 式目标相结合，而无需沿完整流路径进行似然评估。PolicyFlow 利用简单插值路径上的速度场变化来近似重要性比率，从而在不损害训练稳定性的前提下降低计算开销。为进一步防止模式坍缩并进一步鼓励多样化行为，我们提出布朗正则项，一种受布朗运动启发的隐式策略熵正则项，其概念简洁且计算轻量。在包括 MultiGoal、PointMaze、IsaacLab 和 MuJoCo Playground 在内的多种环境的多样化任务上的实验表明，与使用高斯策略的 PPO 以及包括 FPO 和 DPPO 在内的基于流的基线相比，PolicyFlow 取得了具有竞争力或更优的性能。值得注意的是，MultiGoal 上的结果凸显了 PolicyFlow 捕捉更丰富多峰动作分布的能力。

1 引言

强化学习（RL），尤其是策略梯度（PG）方法，在复杂的序贯决策任务中取得了显著成功，应用范围从机器人控制（Andrychowicz 等人，2020；Rudin 等人，2022；Cheng 等人，2024；He 等人，2025）到使大语言模型与人类偏好对齐（Ouyang 等人，2022；Zhai 等人，2024；Guo 等人，2025）。在策略梯度方法中，近端策略优化（PPO）（Schulman 等人，2017）因其简单且通常表现可靠，至今仍是标准方法；它广泛用于复杂的机器人控制任务（Lee 等人，2020；Chen 等人，2025），近来也用于微调生成式策略（Black 等人，2023；Ren 等，2024）。PPO 基于重要性比率，通过代理目标优化策略，而这需要进行较为复杂的似然评估。为便于计算，策略通常建模为高斯分布。尽管使用方便，高斯策略在表示复杂、多模态或高度偏斜的动作分布方面存在局限，因此促使人们采用表达能力更强的生成模型。

近年来，扩散模型（Ho 等人，2020；Song 等人，2020）和连续归一化流（Lipman 等人，2022；Tong 等人，2023）等生成模型已成为一类强大的模型，能够捕捉复杂的多模态分布。这些模型已成功应用于模仿学习，能够直接根据示范数据对策略分布进行建模（Chi 等人，2023；Ze 等人，2024），有效捕捉轨迹多样性和

复杂行为。然而，计算这些模型的重要性比率或似然通常需要迭代进行常微分方程/随机微分方程模拟以及路径反向传播（Chen 等人，2018），这在计算上成本高昂，且容易出现梯度爆炸或消失。这使得在近端策略优化式更新中直接应用此类模型的速度较慢、内存消耗较大，并且可能不稳定，从而限制了它们在高效同策略强化学习中的实用性。

受这些挑战的启发，我们提出了PolicyFlow，这是一种新颖的同策略强化学习算法，将连续归一化流的表达能力与近端策略优化风格的裁剪目标函数相结合，从而实现高效、稳定的策略优化。我们的贡献如下：

• **连续归一化流策略的重要性比率近似。** PolicyFlow 通过评估连续归一化流速度场沿插值路径的变化来近似重要性比率，避免了代价高昂的路径反向传播。

• **布朗运动熵正则化。** 我们提出了一种受布朗运动启发的轻量级熵正则项（Einstein，1905），它能促进熵单调增长、缓解模式坍塌，并鼓励产生多样化的动作，而无需显式计算连续归一化流策略的熵。

这些结果展现了 PolicyFlow 作为同策略强化学习实用且富有表达力的框架所具备的潜力。代码和项目主页见https://policyflow2026.github.io/。

2 相关工作

2.1 基于流/扩散的强化学习策略表示

流模型和扩散模型能够提供表达能力极强的多模态分布，因此在强化学习中可作为颇具吸引力的策略参数化形式。与传统的类别策略或高斯策略相比，这些生成模型能够表示更丰富的动作分布，并有望捕捉更广泛的行为。在机器人学中，基于流和基于扩散的模型已被广泛用作策略表示（Chi 等人，2023；Lei 等人，2025年；Intelligence 等人，2025年；Gao 等人，2025年）。然而，这些模型通常仅通过离线数据集上的去噪分数匹配或流匹配进行训练，而不纳入强化学习，因此其发展已接近瓶颈。这一局限促使研究人员探索利用强化学习直接训练生成式策略。

在**离线强化学习**中，基于扩散的策略已被广泛用于根据静态数据集建模复杂的动作模式，通常由价值函数或基于能量的目标函数引导（Wang 等人，2022；Lu 等人，2023；Psenka 等人，2023；Zhang 等人，2025年）。这些方法在 D4RL 基准上取得了出色的结果，并催生了将生成式策略与价值估计相结合的演员–评论家变体（Wang 等人，2024；Fang 等人，2024）。为减轻扩散模型在采样和计算方面的高昂开销，近期研究也探索了更高效的方案（Kang 等人，2023）。

在**在线强化学习**中，任务要求更高，因为它需要高效采样、稳定的重要性比率估计，以及易于处理（或近似效果良好）的似然。多项研究（Wang 等人，2024；Chao 等人，2024；Ding 等人，2025年）直接通过完整的扩散/流链反向传播策略梯度，这使端到端优化成为可能，但也有梯度爆炸或消失的风险。DPPO（Ren 等人，2024）还提出了实用方案，利用类似策略梯度的更新对表达能力强的扩散策略进行微调，从而实现结构化的流形内探索和稳定的长时域训练。尽管该方法在微调场景中效果良好，但从头开始训练时性能往往会下降，因为此时需要进行流形外探索。FPO（McAllister 等人，2025年）则通过证据下界目标函数估计策略重要性比率。这种方法提供了可扩展的近似，但会引入非对称估计偏差——重要性比率增加时估计更可靠，降低时则不然——可能放大方差并影响稳定性。为缓解这一问题，FPO 通常需要在更新时使用更大的批量大小。

总体而言，先前的研究展现了在线强化学习中表达能力强的生成式策略既有潜力也有局限：虽然它们拓展了可表示的策略类别，但现有方法可能面临优化不稳定、计算成本高昂或近似有偏等问题。PolicyFlow 是一种同策略算法，旨在应对这些挑战，同时无需对完整的

生成链进行反向传播，也不必像 DPPO 那样将扩散视为内部的马尔可夫决策过程，并且能够避免 FPO 中的不对称偏差。

2.2 策略熵正则化

熵正则化长期以来一直用于鼓励强化学习中的探索并防止模式坍缩。经典方法已证明其对离散动作空间中的分类策略（Mnih 等人，2016）以及连续控制中的高斯策略（Haarnoja 等人，2018）的有效性。然而，将这一原则扩展到基于流的策略颇具挑战：动作对数似然通常难以计算，使得熵估计代价高昂。

原则上，可以通过沿流路径积分速度场的散度，推导连续归一化流中熵的闭式动力学（Chen 等人，2018；Tian 等人，2024）。尽管理论上严谨，但这种方法需要进行代价高昂的散度计算和路径积分，从而限制了可扩展性。研究者也探索了更具启发性的解决方案： Wang 等人(2024) 使用高斯混合模型近似熵，并相应调整注入扩散输出的噪声；而 Ding 等人(2024) 则向训练样本中注入均匀噪声，以人为提高熵。这些策略在特定情况下十分有效；如果不必顾虑额外的计算开销，或者事先已知动作样本的取值范围，它们都是不错的选择。

我们的方法引入了一种受布朗运动启发的隐式熵正则项，直接将速度场塑造为趋向熵增的动力学。该设计避免了昂贵的对数似然计算，并绕过了对特设噪声启发式方法的需求。由于熵正则化在FPO等方法中并未被显式处理，我们的正则项为基于流的策略优化提供了一种原则性且轻量的替代方案。

3 背景

我们考虑由元组 $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ 定义的马尔可夫决策过程（马尔可夫决策过程），其中 $\mathcal{S}$ 为状态空间，$\mathcal{A}$ 为动作空间，$p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$表示状态$\mathbf{s} \in \mathcal{S}$与动作$\mathbf{a} \in \mathcal{A}$下的转移动力学，$r(\mathbf{s}, \mathbf{a})$为奖励函数，$\gamma \in [0, 1)$为折扣因子。智能体的目标是学习一个策略 $\pi(\mathbf{a}|\mathbf{s})$，以最大化期望累积折扣回报：

$$J(\pi) = \mathbb{E}_{p(\tau|\pi)} \left[\sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_k, \mathbf{a}_k) \right] \tag{1}$$

where $\tau = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots\}$ 表示在策略 π下从环境中采样得到的一条轨迹。

在策略梯度算法中，近端策略优化（PPO）因其简洁性和实证稳定性，已成为应用最广泛的算法之一。PPO 通过最大化裁剪代理目标进行策略优化（Schulman 等人，2017）：

$$J^{\text{PPO}}(\pi) = \mathbb{E}_{p_{\hat{\pi}}(\mathbf{s})} \mathbb{E}_{\hat{\pi}(\mathbf{a}|\mathbf{s})} \left[\min \left(\frac{\pi}{\hat{\pi}}(\mathbf{a}|\mathbf{s}) A_{\hat{\pi}}(\mathbf{s}, \mathbf{a}), \text{clip} \left(\frac{\pi}{\hat{\pi}}(\mathbf{a}|\mathbf{s}), 1 - \epsilon, 1 + \epsilon \right) A_{\hat{\pi}}(\mathbf{s}, \mathbf{a}) \right) \right] \tag{2}$$

其中 $\hat{\pi}(\mathbf{a}|\mathbf{s})$ 是参考策略，$p_{\hat{\pi}}(\mathbf{s})$是策略的状态分布（Schulman 等人，2015），$A_{\hat{\pi}}(\mathbf{s}, \mathbf{a})$ 是相应的优势函数，而裁剪范围 ϵ 是一个较小的正超参数（通常取值范围为 $[0.1, 0.3]$），用于控制似然比相对于 1 的最大允许偏差。这种方法可以防止更新后的策略偏离参考策略过远，从而确保实践中的学习过程更加稳定。

近期，Frans 等人（2025年）表明，近端策略优化及相关算法也可以通过以下形式的代理目标来解释：

$$\hat{J}(\pi) = \mathbb{E}_{p_{\hat{\pi}}(\mathbf{s})} \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})} A_{\hat{\pi}}(\mathbf{s}, \mathbf{a}). \tag{3}$$

只要所得策略 $\pi^* = \text{argmax}_\pi \hat{J}(\pi)$ 与参考策略$\hat{\pi}$ 之间的散度保持有界，优化该代理目标即可保证真实目标的单调改进，即 $J(\pi^*) > J(\hat{\pi})$。

4 使用连续归一化流进行策略优化

为克服高斯参数化的局限性，我们提出使用连续归一化流来表示策略。具体而言，我们定义一个条件流 φ： $[0,1]\times\mathbb{R}^d\times\mathbb{R}^n\to\mathbb{R}^d$由常微分方程（常微分方程）支配：

$$\frac{d}{dt}\varphi_t(\mathbf{z};\mathbf{s})=v_t(\varphi_t(\mathbf{z};\mathbf{s});\mathbf{s}),\quad \varphi_0(\mathbf{z};\mathbf{s})=\mathbf{z} \tag{4}$$

其中 $\mathbf{z}\in\mathbb{R}^d$ 是潜变量，$\mathbf{s}\in\mathbb{R}^n$ 是状态，而 $v:[0,1]\times\mathbb{R}^d\times\mathbb{R}^n\to\mathbb{R}^d$ 是一个可由神经网络参数化的时变速度场。

与Wang 等人 (2024)类似，策略通过对流进行积分直至其终止时间，并添加高斯噪声来生成动作：

$$\mathbf{a}=\varphi_1(\mathbf{z};\mathbf{s})+\mathbf{n},\quad \mathbf{z}\sim p_z(\mathbf{z}),\quad \mathbf{n}\sim\mathcal{N}(\mathbf{n};\mathbf{0},\boldsymbol{\sigma}^2). \tag{5}$$

这里，注入的噪声 $\mathbf{n}$ 不仅有助于探索，还能确保与近端策略优化风格的替代目标函数兼容，使我们能够自然地将近端策略优化的原始形式扩展到连续归一化流。原则上，$p_z(\mathbf{z})$ 有多种选择，但我们遵循生成模型中的常见做法，选择标准高斯分布，即 $p_z(\mathbf{z})=\mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{1})$。

这一构造产生了一个策略分布

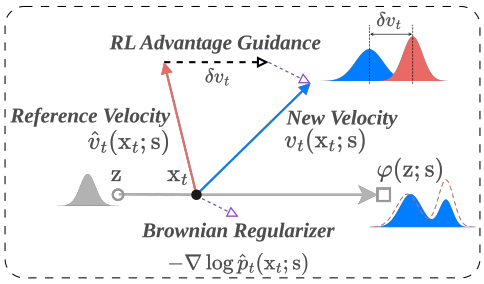
$$\pi(\mathbf{a}|\mathbf{s})=\int\pi(\mathbf{a}|\mathbf{z},\mathbf{s})p_z(\mathbf{z})\,\mathrm{d}\mathbf{z},\quad \pi(\mathbf{a}|\mathbf{z},\mathbf{s})=\mathcal{N}(\mathbf{a};\varphi_1(\mathbf{z};\mathbf{s}),\boldsymbol{\sigma}^2). \tag{6}$$

这种表示的表达能力严格强于传统的高斯策略：它既能还原简单的单峰高斯分布，也能对任意复杂的多模态或高度非高斯的动作分布进行建模。

在这种参数化方式下，式（3）中的策略代理目标可以改写为

$$\hat{J}(\pi)=\mathbb{E}_{p_{\hat{\pi}}(\mathbf{s}),p_z(\mathbf{z})}\mathbb{E}_{\pi(\mathbf{a}|\mathbf{z},\mathbf{s})}A_{\hat{\pi}}(\mathbf{s},\mathbf{a})=\mathbb{E}_{p_{\hat{\pi}}(\mathbf{s}),p_z(\mathbf{z})}\mathbb{E}_{\hat{\pi}(\mathbf{a}|\mathbf{z},\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{z},\mathbf{s})}{\hat{\pi}(\mathbf{a}|\mathbf{z},\mathbf{s})}A_{\hat{\pi}}(\mathbf{s},\mathbf{a})\right], \tag{7}$$

这明确地将基于流的策略表示与
策略优化。



原则上，可以在训练期间通过模拟流 φ 和 $\hat{\varphi}$ 的常微分方程来计算重要性比率。然而，这通常计算开销很大，且数值不稳定，因为神经常微分方程可能会遭遇梯度爆炸/消失，或在训练期间占用大量内存。接下来，我们将介绍一种替代目标，它无需在训练期间直接模拟常微分方程来计算这一重要性比率。

现在，设 $p_n(\cdot;\boldsymbol{\mu},\boldsymbol{\sigma}^2)$ 表示均值为 $\boldsymbol{\mu}$、方差为 $\boldsymbol{\sigma}^2$ 的高斯密度函数。一个关键观察是，高斯分布之间的似然比具有平移不变性，这意味着

$$\frac{\pi(\mathbf{a}|\mathbf{z},\mathbf{s})}{\hat{\pi}(\mathbf{a}|\mathbf{z},\mathbf{s})}=\frac{p_n\left(\mathbf{a};\,\varphi_1(\mathbf{z};\mathbf{s}),\boldsymbol{\sigma}^2\right)}{p_n\left(\mathbf{a};\,\hat{\varphi}_1(\mathbf{z};\mathbf{s}),\hat{\boldsymbol{\sigma}}^2\right)}=\frac{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\,\delta_{\varphi_1}(\mathbf{z};\mathbf{s}),\boldsymbol{\sigma}^2\right)}{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\,\mathbf{0},\hat{\boldsymbol{\sigma}}^2\right)} \tag{8}$$

其中 $\delta_{\varphi_1}(\mathbf{z};\mathbf{s})=\varphi_1(\mathbf{z};\mathbf{s})-\hat{\varphi}_1(\mathbf{z};\mathbf{s})$。直接计算 $\delta_{\varphi_1}(\mathbf{z};\mathbf{s})$ 需要模拟常微分方程，计算代价很高。为缓解这一问题，我们利用沿以下线性插值路径的速度变化来近似终端偏移：

$$\mathbf{x}_t=(1-t)\mathbf{z}+t\hat{\varphi}_1(\mathbf{z};\mathbf{s}),\quad t\in[0,1]. \tag{9}$$

该近似将参考轨迹上的积分替换为沿插值路径对 t 的期望，从而得到：

$$\frac{\pi(\mathbf{a}|\mathbf{z},\mathbf{s})}{\hat{\pi}(\mathbf{a}|\mathbf{z},\mathbf{s})}\approx\mathbb{E}_{p(t)}\left[\frac{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\delta_{v_t}(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2\right)}{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\mathbf{0},\hat{\boldsymbol{\sigma}}^2\right)}\right] \tag{10}$$

其中 $p(t)=U[0,1]$ 且速度场变化 $\delta_{v_t}(\mathbf{x}_t;\mathbf{s})=v_t(\mathbf{x}_t;\mathbf{s})-\hat{v}_t(\mathbf{x}_t;\mathbf{s})$。

注（近似误差界）理论分析（详见附录A）表明，在较小的更新范围内，这种基于插值的近似只会在对数中引入一阶误差，而这可以通过 ϵ 近端策略优化（*PPO*）中的裁剪范围自然地加以控制。

$$\left|\mathbb{E}_{p(t)}\left[\frac{p_n\left(\mathbf{a}-\hat{}\varphi_1(\mathbf{z};\mathbf{s});\delta_{v_t}(\mathbf{x}_t;\mathbf{s})\ \boldsymbol{\sigma}^2\right)}{p_n\left(\mathbf{a}-\hat{}\varphi_1(\mathbf{z};\mathbf{s});\mathbf{0},\hat{}\boldsymbol{\sigma}^2\right)}\right]-\frac{p_n\left(\mathbf{a}\quad-\varphi_1\qquad\delta_{\varphi_1}\qquad\boldsymbol{\sigma}^2\right)\hat{}(\mathbf{z};\mathbf{s});}{\begin{array}{c}p_n\quad-\varphi_1\qquad\mathbf{0},\boldsymbol{\sigma}^2\\(\mathbf{z};\mathbf{s})\ (\mathbf{a}\hat{}(\mathbf{z};\mathbf{s});\hat{}\ )\end{array}}\right|=\mathcal{O}(\epsilon). \tag{11}$$

重要的是，这种方法使我们能够避免模拟完整流轨迹并在训练期间沿其传播梯度，从而保持与采用高斯策略的近端策略优化相当的计算效率。

现在，速度场 v 由参数为 θ 的神经网络参数化。最后，与近端策略优化类似，我们采用裁剪代理式目标函数来稳定训练：

$$J^{\text{Flow}}(\theta,\boldsymbol{\sigma})=\mathbb{E}_{p_{\hat{\pi}}(\mathbf{s}),p_z(\mathbf{z})}\mathbb{E}_{\hat{\pi}(\mathbf{a}|\mathbf{z},\mathbf{s}),\,p(t)}\left[\min\left(\rho A_{\hat{\pi}}(\mathbf{s},\mathbf{a}),\text{clip}\left(\rho,1-\epsilon,1+\epsilon\right)A_{\hat{\pi}}(\mathbf{s},\mathbf{a})\right)\right] \tag{12}$$

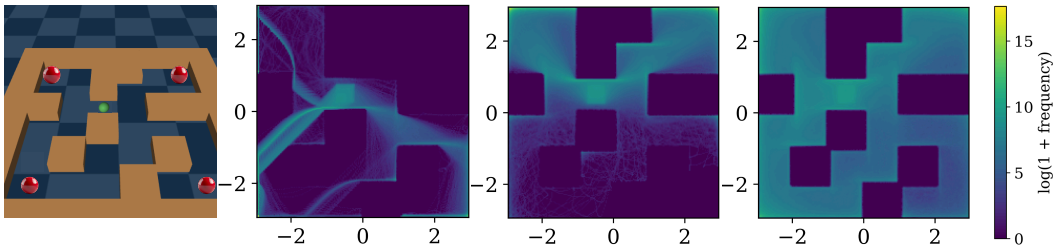
使用近似的重要性比率

$$\rho=\frac{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\ v_t(\mathbf{x}_t;\mathbf{s},\theta)-\hat{v}_t(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2\right)}{p_n\left(\mathbf{a}-\hat{\varphi}_1(\mathbf{z};\mathbf{s});\ \mathbf{0},\hat{\boldsymbol{\sigma}}^2\right)}. \tag{13}$$

因此，仅在采样期间才需要模拟常微分方程（以计算 $\hat{}\varphi_1(\mathbf{z};\mathbf{s})$），而训练目标则可以沿着具有速度场变化的插值路径高效估计，无需在训练期间模拟常微分方程或对模拟得到的流轨迹进行反向传播。

4.1 策略熵正则化

策略熵最大化是强化学习中一项历史悠久的技术，用于鼓励探索并缓解模式坍缩。针对基于流的策略进行熵正则化时面临的困难（如第2.2节所述），我们提出了一种受布朗运动启发的新型熵正则项。我们的方法不同于以往的熵正则化：我们不显式计算策略熵，也不通过启发式方法注入噪声，而是直接调节速度场，使其遵循一个熵递增过程。这一视角使我们能够避免代价高昂的对数似然评估，同时仍能鼓励多样化探索，如图1所示。



(a) (b) (c) (d)

图 1：（`PointMaze-Medium-Diverse-GDense-v3`）探索密度图。(a) 环境概览：每个回合中，智能体都从绿色点出发；四个红色点表示奖励相同的目标位置。(b)PPO 的探索热力图，显示由于策略采用简单的高斯策略，覆盖范围有限。(c) 未使用布朗正则项的 PolicyFlow 探索热力图。覆盖范围有所改善，但仍有一些区域探索不足。(d)使用布朗正则项的 PolicyFlow 探索热力图，几乎覆盖了所有可行位置。

在布朗动力学中，粒子会自然地向均匀分布扩散，且在这一过程中熵单调增加。尽管布朗运动由随机微分方程定义，其概率路径遵循经典热方程 $\partial p_t(\mathbf{x})/\partial t=\nabla^2_{\mathbf{x}}p_t(\mathbf{x})$(Jor- dan 等人，1998）。该方程与连续性方程 $\partial p_t(\mathbf{x})/\partial t=-\nabla_{\mathbf{x}}\cdot(p_t(\mathbf{x})v_t(\mathbf{x}))$直接相关。通过选择 $v_t(\mathbf{x})=-\nabla_{\mathbf{x}}$ 对数 $p_t(\mathbf{x})$，连续性方程便可还原热方程，这表明，通过精心设计与负分数函数对齐的速度场，可以促使熵增长。

实际上，为了促进流轨迹像布朗运动一样扩展，而不是收缩，我们希望学习到的速度场遵循参考流的负分数函数。为了获得该分数函数

算法 1 PolicyFlow

1: 输入：初始速度场参数 θ_0、初始噪声方差 $\boldsymbol{\sigma}_0^2$、初始价值函数参数 $\phi_0$2: 对于迭代 $i = 0$，1、2、.....，执行 3: 设置参考参数 $\hat{}\theta \leftarrow \theta_i$、$\hat{}\boldsymbol{\sigma}^2 \leftarrow \boldsymbol{\sigma}_i^2$。参考速度场为 $\hat{}v = v_{\hat{\theta}}$4: 使用参考策略 $\pi_{\hat{\theta}}$、$\hat{\sigma}$ 收集一组轨迹 $\mathcal{D}_i$ 5: 对于每个马尔可夫决策过程步骤 k，状态为 $\mathbf{s}_k$，执行 6: 对潜变量 $\mathbf{z}_k \sim p_z(\mathbf{z})$ 进行采样，然后 $\hat{}\varphi_0 = \mathbf{z}_k$7: 通过从 $t = 0$ 到 1 模拟常微分方程 $\frac{d}{dt}\hat{}\varphi_t = \hat{}v_t(\hat{}\varphi_t; \mathbf{s}_k)$，计算 $\boldsymbol{\varphi}_k = \hat{}\varphi_1(\mathbf{z}_k; \mathbf{s}_k)$ 8: 采样噪声 $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \hat{}\boldsymbol{\sigma}^2)$ 并构成动作 $\mathbf{a}_k = \boldsymbol{\varphi}_k + \mathbf{n}_k$9: 执行 $\mathbf{a}_k$，观察下一个状态 $\mathbf{s}_{k+1}$ 和奖励 r_k10: 将状态转移 $(\mathbf{s}_k, \mathbf{a}_k, r_k, \mathbf{s}_{k+1}, \mathbf{z}_k, \boldsymbol{\varphi}_k)$ 存入 $\mathcal{D}_i$11: 结束循环 $\hat{}$ $\hat{}$ 12: 对于每个步骤 k，使用广义优势估计计算未来累计奖励 R_k 和优势估计 A_k 13: 对于训练轮次 $= 1$、.....、E，执行 14: 对于来自 $\mathcal{D}_i$ 的每个状态转移小批量 $(\mathbf{s}_k, \mathbf{a}_k, A_k, \mathbf{z}_k, \boldsymbol{\varphi}_k)$，执行 15: 采样 $t_k \sim U[0, 1]$16: 或从用于流常微分方程数值模拟的离散时间点中采样 t_k 17: 计算插值点 $\mathbf{x}_{t_k} = (1 - t_k)\mathbf{z}_k + t_k\boldsymbol{\varphi}_k$18: 计算近似重要性比率 $\rho_k = p_n\left(\mathbf{a}_k - \boldsymbol{\varphi}_k; v_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}_k, \theta) - \hat{}v_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}_k)\ \boldsymbol{\sigma}^2\right)/p_n\left(\mathbf{a}_k - \boldsymbol{\varphi}_k; \mathbf{0}, \hat{}\boldsymbol{\sigma}^2\right)$19: 通过 $J^{\text{Flow}} = \mathbb{E}_k\left[\min(\rho_k \hat{}A_k, \text{clip}(\rho_k, 1 - \epsilon, 1 + \epsilon)\ \hat{}A_k)\right]$ 计算小批量的裁剪代理目标 20: 计算布朗正则项向量 21: 计算小批量的正则化项： 22: 更新策略参数 $(\theta_{i+1}, \boldsymbol{\sigma}_{i+1}) = \arg\max_{\theta,\sigma}(J_{\text{ow}} + J_{\text{eg}})$。23: 通过最小化均方误差更新价值函数参数：24: 结束循环 25: 结束循环 26: 结束循环

$$\eta_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}_k, \theta) = (1 - t_k)v_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}_k, \theta) - (\mathbf{x}_{t_k} - t_k\,\hat{v}_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}))$$

$$J^{\text{Reg}} = \mathbb{E}_k\left[-w_b\|\eta_{t_k}(\mathbf{x}_{t_k}; \mathbf{s}, \theta)\|_2^2 + \frac{w_g}{2}\sum_{i=1}^{d}\log(2\pi e\sigma_i^2)\right]$$

Fl R

$$\phi_{i+1} = \arg\min_{\phi}\mathbb{E}_k\left(\mathcal{V}_\phi(\mathbf{s}_k) - \hat{R}_k\right)^2$$

来自参考速度场的函数，我们可以利用Liu等人(2025年)的结果，速度场与得分函数存在显式关系：

$$\nabla_{\mathbf{x}}\log\hat{p}_t(\mathbf{x}_t; \mathbf{s}) = \frac{1}{1 - t}(t\,\hat{v}_t(\mathbf{x}_t; \mathbf{s}) - \mathbf{x}_t). \tag{14}$$

基于这一联系，我们提出一种实用的熵正则项：

$$J^{\text{Reg}}(\theta, \boldsymbol{\sigma}) = \mathbb{E}_{p_{\hat{\pi}}(\mathbf{s}),\, p_z(\mathbf{z}),\, p(t)}\Big[-w_b\|\eta_t(\mathbf{x}_t; \mathbf{s}, \theta)\|_2^2 + \frac{w_g}{2}\sum_{i=1}^{d}\log(2\pi e\sigma_i^2)\Big], \tag{15}$$

其中 w_b， $w_g \geq 0$ 为可调系数，且

$$\eta_t(\mathbf{x}_t; \mathbf{s}, \theta) = (1 - t)v_t(\mathbf{x}_t; \mathbf{s}, \theta) - (\mathbf{x}_t - t\,\hat{v}_t(\mathbf{x}_t; \mathbf{s})). \tag{16}$$

式（布朗正则项）中的第一项（称为15）鼓励学习到的速度场与参考流的负分数函数对齐，从而促进轨迹扩展，并防止轨迹塌缩到狭窄模态。请注意，我们不直接计算速度场与负分数函数之差，因为这会使分母中出现$(1 - t)$ 这一因子；当 $t \to 1$时，这会带来问题； η_t 的定义旨在安全地确保对齐，同时避免这种

奇异性。式（15）中的第二项对应于在流的终点注入的高斯噪声$\mathbf{n}$的熵，它增强了随机性并鼓励多样化探索。这两项共同促进轨迹多样性，并保持连续归一化流在建模复杂的多模态动作分布方面的表达能力（见图2）。

值得注意的是，不同于以往需要计算对数似然、进行代价高昂的散度积分（Chen 等人，2018；Tian 等人，2024）或启发式噪声注入（Frans 等人，2025；Wang 等人，2024）的熵正则项，布朗正则项提供了一种有原则且计算开销较低的替代方案。

说明：布朗正则项不应被视为理论上精确的推导。具体而言，尽管我们的公式利用了整流流中速度场与得分函数之间的关系，但策略中的速度场并非通过流匹配梯度得到，因此并不严格对应整流流动力学。

5 实验

我们将 PolicyFlow 与 FPO 和 DPPO 进行基准比较，因为这两种方法都将近端策略优化扩展到无法显式进行似然评估的生成式策略类别。目前，这些算法代表了将同策略强化学习应用于表达能力强的非高斯策略参数化的最先进方法。因此，与 FPO 和 DPPO 进行比较对于证明 PolicyFlow 作为一种通用且有原则的生成式策略训练替代方案的有效性至关重要。我们在 MuJoCo Playground（Zakka等人，2025年）和 IsaacLab（Mittal 等人，2023年）的基准测试中评估这些算法。我们使用 MultiGoal 环境，测试布朗正则项在充分利用连续归一化流来捕捉复杂的多模态分布并避免模式坍缩方面的作用。

5.1 MULTIGOAL 测试

MultiGoal 环境最初由 Haarnoja 等人（2017年）提出，是一个具有六个固定目标位置的二维方形工作空间，我们用它来评估布朗正则项如何防止模式坍缩，以及连续归一化流如何实现表达能力更强的多模态策略。我们将动力学修改为二阶系统：状态包含位置和速度，动作则为加速度。智能体和环境的完整详细信息见
附录 C.2。

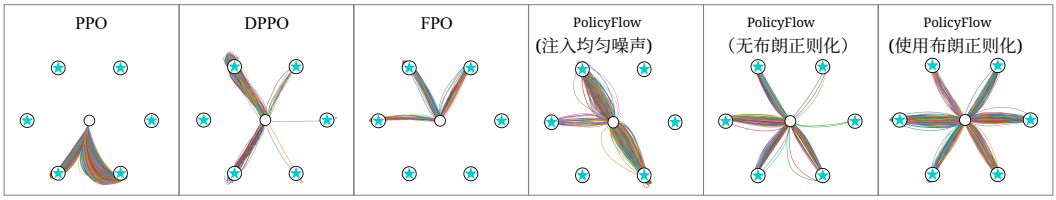


图 2：MultiGoal 测试（附录 C.2）：从同一个原始点出发采样 1000 条轨迹。(a) 采用高斯熵正则化的近端策略优化（$w_g = 0.001$）仅覆盖有限的一组目标。(b,c) DPPO 和 FPO 坍缩到少数几种模式，这很可能是因为这两种方法都没有采用任何形式的熵正则化。(d) 采用均匀噪声注入的 PolicyFlow（Ding 等人，2024）（权重 0.05）仍然存在模式坍缩，仅集中于少数几种模式。(e) 仅采用高斯熵正则化的 PolicyFlow（$w_g = 0.001$）在一定程度上缓解了模式坍缩。(f) 采用所提出的布朗正则项（$w_b = 0.25$）和高斯熵正则化（$w_g = 0.001$的 PolicyFlow 实现了最多样且分布更均衡的目标到达行为。

如果智能体从工作空间中心出发，那么六个目标的距离相同，奖励也具有对称性，因此，最优策略应以大致相同的概率到达每个目标，这反映了该任务的多模态特性。如图 2所示，采用高斯策略的近端策略优化只能表示简单分布，因此难以生成能够到达全部六个目标位置的轨迹。虽然

FPO 和 DPPO 使用能够表达更复杂分布的生成模型，但由于缺乏有效的熵正则化机制，智能体无法学到足够多样的轨迹。相比之下，采用布朗正则项的 PolicyFlow 充分发挥了连续归一化流的表达能力，从而生成更加均衡的多模态动作模式，并覆盖更多目标。

5.2 MUJOCO PLAYGROUND 和 ISAACLAB 基准测试

MuJoCo Playground 基准测试。 我们在 MuJoCo Playground 基准测试中，将 PolicyFlow 与当前最先进的基于流的方法进行比较，包括 FPO（McAllister 等人，2025年）和 DPPO（Ren 等人，2024）。这些方法都基于 PPO 框架，因此我们也将 PPO 纳入基线比较。FPO 使用连续归一化流（CNF）表示策略，而 DPPO 使用扩散模型。FPO 和 DPPO 的原始实现均未包含显式熵正则化，这可能会限制所学习策略的多样性。如 MultiGoal 测试所示，PolicyFlow 能有效保留多模态行为和多样化轨迹。在 MuJoCo Playground 的各项任务中，PolicyFlow 在大多数环境中的表现与 FPO 相当或更优，优于 DPPO，且总体上与 PPO 不相上下或胜过 PPO。仔细分析训练曲线可见，PolicyFlow 往往收敛得更快，这表明其样本效率更高、探索更有效，也补充了 MultiGoal 实验中关于策略多样性的观察结果。

PolicyFlow 的超参数设置见附录C.4。为确保公平比较，FPO 和 DPPO 的超参数遵循 FPO 论文中的调优配置，近端策略优化则使用 MuJoCo Playground 代码库推荐的默认设置。

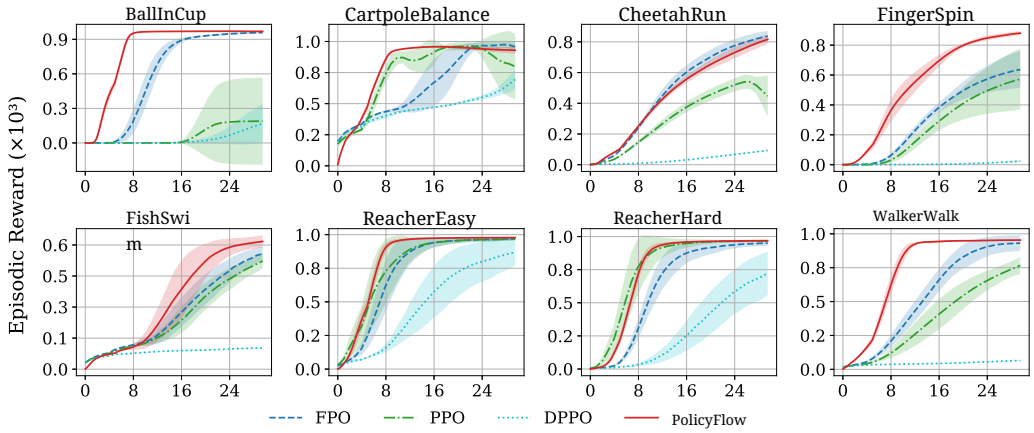


图 3：MuJoCo Playground 基准测试上的学习曲线。图中展示了回合奖励均值及标准误（纵轴）随环境步数（横轴，共计 3000 万步）的变化，结果为 5 个随机种子的平均值。

IsaacLab 基准测试。 我们进一步在 IsaacLab 基准测试中评估 PolicyFlow。该基准套件包含一系列机器人环境，涵盖运动控制、操作和 Navigation。IsaacLab 是由英伟达维护的近期开发且快速发展的框架，专为大规模机器人学习而设计。其高仿真保真度、强大的工程支持以及在机器人学界日益增长的普及度，使其成为评估强化学习算法性能的理想试验平台。 在此基准测试中，我们仅将 PolicyFlow 与近端策略优化进行比较。尽管 FPO 和 DPPO 都是最先进的生成式策略方法，但它们最初的基准测试套件均未包含 IsaacLab 任务，而且要将其公开发布的代码直接适配到 IsaacLab 环境栈，需要投入大量工程工作，并进行复杂的环境重新集成。因此，我们采用 `RSL-RL` 中的近端策略优化实现作为基线，并遵循官方的 IsaacLab 超参数配置。PolicyFlow 使用的超参数几乎与近端策略优化相同（见附录 C.3），但增加了熵正则化机制所需的配置。参数的完整细节见附录 C.3。如表 1 和附录 5 中的图 C.3所示，PolicyFlow 在所有任务上的渐近性能始终与近端策略优化持平或更优。由于 PolicyFlow 学习的是时变速度场，而非直接的动作映射，其优化问题本质上更加复杂，因此早期学习速度可能较慢。

表 1：IsaacLab 基准测试上的训练终止回合奖励。

方法	Lift-Cube	Navigation	Open-Drawer	Quadcopter	ANYmal-D	G1	H1	Go2
PPO	153.1 ± 3.0	3.5 ± 0.3	$\mathbf{99.8 \pm 1.7}$	$\mathbf{141.8 \pm 0.5}$	24.5 ± 0.1	25.4 ± 1.2	$\mathbf{29.3 \pm 0.9}$	$\mathbf{27.9 \pm 0.3}$
PolicyFlow	$\mathbf{154.6 \pm 0.6}$	$\mathbf{4.2 \pm 0.1}$	99.1 ± 0.7	141.0 ± 0.09	$\mathbf{24.6 \pm 0.2}$	$\mathbf{30.0 \pm 1.1}$	27.3 ± 0.2	27.4 ± 0.9
p-值	0.32	**0.0027**	0.41	0.099	0.26	**0.00026**	**0.0069**	0.33

每次迭代训练时间。 我们比较了 PolicyFlow 和近端策略优化在 IsaacLab 环境中的每次迭代训练时间，该指标反映单次训练步骤的计算成本。如表 2所示，当模型参数规模与近端策略优化大致相当时，PolicyFlow 在前六个 IsaacLab 环境中的每次迭代训练时间增加不到 50%。即使将嵌入维度提高至原来的八倍，计算成本仍不到近端策略优化的两倍，表明 PolicyFlow 在实践中效率很高。

表2：近端策略优化与PolicyFlow在IsaacLab基准测试上的每次迭代训练时间，在RTX 5090 GPU上对50次迭代取平均。

环境	Lift-Cube	Navigation	Open-Drawer	Quadcopter	Anymal-D	G1	H1	Go2
嵌入维度（时间 / 观测）	64	64	64	64	64	256	512	512
近端策略优化（毫秒）	43.0 ± 16.1	36.9 ± 6.3	81.3 ± 14.7	37.8 ± 13.8	41.2 ± 13.4	66.9 ± 14.4	63.4 ± 15.5	63.9 ± 15.7
PolicyFlow（毫秒）	57.7 ± 20.8	54.1 ± 10.1	104.1 ± 16.5	55.6 ± 15.3	57.1 ± 15.7	90.6 ± 12.3	115.5 ± 17.3	111.5 ± 15.1

说明：我们没有直接比较 *FPO* 或 *DPPO*，因为 *FPO* 开源代码库中这些算法的实现基于 *JAX*，而 *PolicyFlow* 则使用 *PyTorch* 实现。在不同深度学习框架之间进行直接比较可能导致结果不可靠，因此我们将分析重点放在采用相同框架实现的 *PPO* 上，以提供公平的基线。

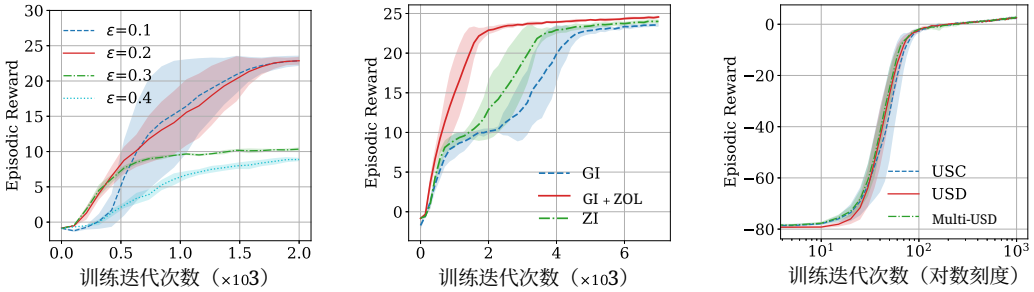
5.3 对裁剪范围参数的敏感性

我们提出的重要性比率近似公式（13）会引入近似误差，理论上该误差受裁剪范围 ϵ所限，如附录 A所示。较小的 ϵ 会使界限更紧，从而减小近似误差；然而，这也会限制策略优化中的有效更新步长，可能减缓策略改进。为通过实证验证这一权衡，我们在 IsaacLab ANYmal-D 环境中进行敏感性分析，评估四种裁剪范围 $\epsilon \in \{0.1, 0.2, 0.3, 0.4\}$，每种使用五个随机种子。图 4a所示结果印证了我们的理论见解：较小的裁剪范围虽然能降低近似误差，但由于更新过于保守，会阻碍学习进展。在所有 IsaacLab 基准测试中，我们采用 $\epsilon = 0.2$作为默认裁剪范围，这与 IsaacLab 提供的官方 PPO 配置一致。

5.4 对网络初始化和时间采样策略的敏感性

我们还研究了不同网络初始化策略对性能的影响。多层感知机网络的一种常见选择是 Glorot 初始化（Glorot 和 Bengio，2010），它从具有适当方差的高斯分布中采样权重。另一种方法是将所有网络参数初始化为零。在本研究中，我们比较了 IsaacLab ANYmal-D 环境中的三种初始化方案：(i)**GI**：标准 Glorot 初始化；(ii)**GI+ZOL**：Glorot 初始化，并将输出层额外设为零；以及 (iii)**ZI**：将所有参数完全零初始化。如图 4b所示，将输出层设为零的 Glorot 初始化取得了最佳实证性能。因此，我们在所有基准实验中均采用该方案初始化模型。

此外，为了估计目标函数 t式（12）中关于时间的期望，还需要进行时间采样。我们在 IsaacLab 导航环境中评估了几种时间采样策略：(i) **USC**：从连续区间 $[0, 1]$中均匀采样；(ii) **USD**：从离散常微分方程仿真时间网格 $\{0, 0.05, 0.1, \ldots, 0.95, 1.0\}$中均匀采样；(iii) **Multi-USD**：针对每个状态–动作样本采样多个时间点 t，其中每个 t 均从同一离散网格中均匀抽取。结果如图4c所示。总体而言，这三种时间采样策略对性能的影响差异很小。因此，所有基准实验均采用 USD 作为默认选项。通常不建议使用 Multi-USD，因为它会增加额外的计算开销，却没有明显收益。



(a) 裁剪范围比较 (b) 不同的初始化 (c) 时间采样策略

图4： PolicyFlow关键组成部分的消融研究。

5.5 插值路径的不同选择

前文各节采用了与整流流相同的插值路径。然而，流匹配领域的既有研究也探索了其他插值策略。表4 总结了整流流路径之外的另外两种选择。我们在一个 IsaacLab 运动控制任务和 Multi-目标测试中，除插值方案及其对应的布朗正则项外，所有智能体设置均保持一致。如表3所示，在 ANYmal-D 运动任务中，三种路径最终获得的回合奖励几乎相同；而在 MultiGoal 测试中，TriFlow路径和整流流路径的表现优于随机插值路径。随机插值路径的表现略低，可能是因为我们在该插值过程中使用了得分函数与速度场之间的近似关系，而非精确等式。有关该近似的详细信息，请参见附录B 。

表3：使用不同插值路径时训练终止回合奖励。

路径 / 环境	ANYmal-D	MultiGoal
整流流路径	24.6 ± 0.2	$\mathbf{8.79 \pm 0.02}$
随机插值路径	$\mathbf{24.7 \pm 0.2}$	8.22 ± 0.18
TrigFlow路径	24.5 ± 0.1	8.74 ± 0.03

表4：流匹配的不同插值路径

方法 / 插值公式	得分函数与速度场之间的关系
整流流（Liu 等人，2022） $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \quad t \in [0,1]$	$(1-t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = t\, v_t(\mathbf{x}) - \mathbf{x}$
随机插值器（Albergo et al.，2023） $\mathbf{x}_t = (1-t)\,\mathbf{x}_0 + t\,\mathbf{x}_1 + \sqrt{2t(1-t)}\,\mathbf{h} \;\; t \in [0,1], \mathbf{h} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$	$(2(t-\frac{1}{2})^2 + \frac{1}{2})\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \approx t\, v_t(\mathbf{x}) - \mathbf{x}$
TrigFlow（Lu &Song，2024） $\mathbf{x}_t = \cos(t)\mathbf{x}_0 + \sin(t)\mathbf{x}_1, \quad t \in [0, \frac{\pi}{2}]$	$\cos(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \sin(t)\, v_t(\mathbf{x}) - \cos(t)\mathbf{x}$

6 结论与未来工作 KS

在这项工作中，我们提出了 PolicyFlow，这是一种将连续归一化流与 PPO 式优化相结合的同策略强化学习算法。PolicyFlow 通过沿插值路径利用速度场变化近似重要性比率，无需进行计算开销高昂的路径反向传播，同时保持了稳定性和效率。此外，我们提出的布朗正则项以一种有理论依据且轻量的方式缓解模式坍缩，并鼓励多样化探索。通过在 MultiGoal、IsaacLab 和 MuJoCo Playground 上开展大量实验，PolicyFlow 的表现始终与 PPO 和最先进的方法 FPO、DPPO 相当或更优。尤其是，MultiGoal 上的结果展现了 PolicyFlow 捕捉复杂多模态动作分布的能力。展望未来，PolicyFlow 为连接生成建模与强化学习提供了一个灵活的基础。尽管不同的插值路径在实践中显示出良好前景，但其形式化验证和理论意义仍有待探索。其他值得探索的方向包括微调流匹配策略、将 PolicyFlow 拓展至离线强化学习及更广泛的生成建模任务，以及通过融入基于分数的目标来探索其与扩散模型的联系。最后，进一步全面理解 PolicyFlow 的理论基础，或可启发算法改进，并增强其在现实世界决策中的适用性。

可复现性声明

我们在第 4 节和算法1中提供了详细的算法说明，在附录C中提供了完整的超参数设置，并在附录C.2中提供了 MultiGoal 环境配置。我们的实现基于以下公开的 PPO 实现：`RSL-RL` 和 `SKRL`。大多数结果均为多个随机种子的平均值；即使存在随机性，我们的结论依然可靠。

参考文献

Michael S Albergo、Nicholas MBoffi 和 Eric Vanden-Eijnden。随机插值器：流与扩散的统一框架。*arXiv* 预印本 *arXiv:2303.08797*，2023。

OpenAI：Marcin Andrychowicz、BowenBaker、Maciek Chociej、Rafal Jozefowicz、Bob McGrew、Jakub Pachocki、Arthur Petron、Matthias Plappert、Glenn Powell、Alex Ray 等。学习灵巧手内操作。国际机器人研究期刊，39(1)：3–20，2020。

Kevin Black、Michael Janner、Yilun Du、Ilya Kostrikov 和 Sergey Levine。使用强化学习训练扩散模型。*arXiv* 预印本 *arXiv:2305.13301*，2023。

Chen-Hao Chao、Chien Feng、Wei-Fang Sun、Cheng-Kuang Lee、Simon See 和 Chun-Yi Lee。通过基于能量的归一化流实现最大熵强化学习。神经信息处理系统进展，37：56136–56165，2024。

Ricky TQ Chen、Yulia Rubanova、Jesse Bettencourt 和 David K Duvenaud。神经常微分方程。神经信息处理系统进展，31，2018。

Zixuan Chen、Mazeyu Ji、Xuxin Cheng、Xuanbin Peng、Xue Bin Peng 和 Xiaolong Wang。GMT：用于人形全身控制的通用运动跟踪。*arXiv* 预印本 *arXiv:2506.14770*，2025年。

Xuxin Cheng、Kexin Shi、Ananye Agarwal 和 Deepak Pathak。足式机器人的极限跑酷。收录于*2024 IEEE* 机器人与自动化国际会议（*ICRA*），第11443–11450页。IEEE，2024。

Cheng Chi、Zhenjia Xu、Siyuan Feng、Eric Cousineau、Yilun Du、Benjamin Burchfiel、Russ Tedrake 和 Shuran Song。Diffusion policy：通过动作扩散进行视觉运动策略学习。《国际机器人研究期刊》，第 02783649241273668 页，2023 年。

Shutong Ding、Ke Hu、Zhenhao Zhang、Kan Ren、Weinan Zhang、Jingyi Yu、Jingya Wang 和 Ye Shi。通过 Q 加权变分策略优化实现基于扩散的强化学习。神经信息处理系统进展，37：53945–53968，2024。

Shutong Ding、Ke Hu、Shan Zhong、Haoyang Luo、Weinan Zhang、Jingya Wang、Jun Wang 和 Ye Shi。Genpo：生成式扩散模型与同策略强化学习相遇。*arXiv* 预印本 *arXiv:2505.18763*，2025年。

¨阿尔伯特·爱因斯坦。关于分子运动论所要求的悬浮于静止液体中的微粒运动。《物理学年鉴》，4，1905年。

Linjiajie Fang、Ruoxue Liu、Jing Zhang、Wenjia Wang 和 Bing-Yi Jing。扩散演员-评论家：将约束策略迭代表述为用于离线强化学习的扩散噪声回归。*arXiv* 预印本 *arXiv:2405.20555*，2024。

Kevin Frans、Seohong Park、Pieter Abbeel 和 Sergey Levine。扩散引导是一种可控策略改进算子。*arXiv* 预印本 *arXiv:2505.23458*，2025年。

Dechen Gao、Boqi Zhao、Andrew Lee、Ian Chuang、Hanchu Zhou、Hang Wang、Zhe Zhao、Junshan Zhang 和 Iman Soltani。Vita：视觉到动作流匹配策略。*arXiv* 预印本 *arXiv:2507.13231*，2025年。

Xavier Glorot 和 Yoshua Bengio。理解深度前馈神经网络训练的难点。载于第十三届人工智能与统计国际会议论文集，第 249–256 页。《机器学习研究杂志》研讨会与会议论文集，2010年。Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Peiyi Wang、Qihao Zhu、Runxin Xu、Ruoyu Zhang、Shirong Ma、Xiao Bi 等。DeepSeek-R1 通过强化学习激励大语言模型进行推理。《自然》，645(8081):633–638，2025年。Tuomas Haarnoja、Haoran Tang、Pieter Abbeel 和 Sergey Levine。采用深度基于能量的策略进行强化学习。载于国际机器学习会议，第 1352–1361 页。机器学习研究论文集，2017年。Tuomas Haarnoja、Aurick Zhou、Pieter Abbeel 和 Sergey Levine。软演员-评论家：采用随机演员的离策略最大熵深度强化学习。载于国际机器学习会议，第 1861–1870 页。机器学习研究论文集，2018年。

Junzhe He、Chong Zhang、Fabian Jenelten、Ruben Grandia、Moritz B¨acher和Marco Hutter。用于学习泛化腿式运动的基于注意力的地图编码。科学机器人学，10(105):eadv3604，2025年。

Jonathan Ho、Ajay Jain 和 Pieter Abbeel。去噪扩散概率模型。神经信息处理系统进展，33:6840–6851，2020。Physical Intelligence、Kevin Black、Noah Brown、James Darpinian、Karan Dhabalia、Danny Driess、Adnan Esmail、Michael Equi、Chelsea Finn、Niccolo Fusai 等。 $pi_{0.5}$：具有开放世界泛化能力的视觉-语言-动作模型。*arXiv* 预印本 *arXiv:2504.16054*，2025。Richard Jordan、David Kinderlehrer 和 Felix Otto。福克-普朗克方程的变分形式。《工业与应用数学学会数学分析杂志》，29(1):1–17，1998。Bingyi Kang、Xiao Ma、Chao Du、Tianyu Pang 和 Shuicheng Yan。用于离线强化学习的高效扩散策略。神经信息处理系统进展，36:67195–67212，2023。Joonho Lee、Jemin Hwangbo、Lorenz Wellhausen、Vladlen Koltun 和 Marco Hutter。在复杂地形上学习四足运动。《科学·机器人学》，5(47):eabc5986，2020。Kun Lei、Huanyu Li、Dongjie Yu、Zhenyu Wei、Lingxiao Guo、Zhennan Jiang、Ziyu Wang、Shiyu Liang 和 Huazhe Xu。RL-100：利用真实世界强化学习实现高性能机器人操作。*arXiv* 预印本 *arXiv:2510.14830*，2025。Yaron Lipman、Ricky TQ Chen、Heli Ben-Hamu、Maximilian Nickel 和 Matt Le。用于生成建模的流匹配。*arXiv* 预印本 *arXiv:2210.02747*，2022。Jie Liu、Gongye Liu、Jiajun Liang、Yangguang Li、Jiaheng Liu、Xintao Wang、Pengfei Wan、Di Zhang 和 Wanli Ouyang。Flow-GRPO：通过在线强化学习训练流匹配模型。*arXiv* 预印本 *arXiv:2505.05470*，2025。Xingchao Liu、Chengyue Gong 和 Qiang Liu。流动得笔直且快速：利用整流流学习生成和迁移数据。*arXiv* 预印本 *arXiv:2209.03003*，2022。Cheng Lu 和 Yang Song。简化、稳定并扩展连续时间一致性模型。*arXiv* 预印本 *arXiv:2410.11081*，2024。Cheng Lu、Huayu Chen、Jianfei Chen、Hang Su、Chongxuan Li 和 Jun Zhu。用于离线强化学习中精确能量引导扩散采样的对比能量预测。在 国际机器学习会议，第 22825–22855 页。PMLR，2023。David McAllister、Songwei Ge、Brent Yi、Chung Min Kim、Ethan Weber、Hongsuk Choi、Haiwen Feng 和 Angjoo Kanazawa。流匹配策略梯度。*arXiv* 预印本 *arXiv:2507.21053*，2025。

Mayank Mittal、Calvin Yu、Qinxi Yu、Jingzhou Liu、Nikita Rudin、David Hoeller、Jia Lin Yuan、Ritvik Singh、Yunrong Guo、Hammad Mazhar、Ajay Mandlekar、Buck Babich、Gavriel State、Marco Hutter 和 Animesh Garg。Orbit：用于交互式机器人学习环境的统一仿真框架。*IEEE*机器人与自动化快报，8(6):3740–3747，2023。doi：10.1109/LRA.2023.3270034。

Volodymyr Mnih、Adria Puigdomenech Badia、Mehdi Mirza、Alex Graves、Timothy Lillicrap、Tim Harley、David Silver 和 Koray Kavukcuoglu。深度强化学习的异步方法。载于国际机器学习会议，第 1928–1937 页。PmLR，2016。

Long Ouyang、Jeffrey Wu、Xu Jiang、Diogo Almeida、Carroll Wainwright、PamelaMishkin、Chong Zhang、Sandhini Agarwal、Katarina Slama、Alex Ray 等。利用人类反馈训练语言模型遵循指令。神经信息处理系统进展，35：27730–27744，2022。

MichaelPsenka、Alejandro Escontrela、PieterAbbeel 和 YiMa。通过 q分数匹配从奖励学习扩散模型策略。*arXiv* 预印本 *arXiv:2312.11752*，2023。

Allen Z Ren、Justin Lidard、Lars L Ankile、Anthony Simeonov、Pulkit Agrawal、Anirudha Majumdar、Benjamin Burchfiel、Hongkai Dai 和 Max Simchowitz。扩散策略策略优化。*arXiv* 预印本 *arXiv:2409.00588*，2024。

Nikita Rudin、David Hoeller、Philipp Reist 和 Marco Hutter。利用大规模并行深度强化学习，在几分钟内学会行走。发表于机器人学习会议，第 91–100 页。机器学习研究论文集，2022。

John Schulman、Sergey Levine、Pieter Abbeel、Michael Jordan 和 Philipp Moritz。信赖域策略优化。发表于国际机器学习会议，第 1889–1897 页。机器学习研究论文集，2015。

John Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford 和 Oleg Klimov。近端策略优化算法。*arXiv* 预印本 *arXiv:1707.06347*，2017。

Clemens Schwarke、Mayank Mittal、Nikita Rudin、David Hoeller 和 Marco Hutter。Rsl-rl：面向机器人研究的学习库。*arXiv* 预印本 *arXiv:2509.10771*，2025年。

Antonio Serrano-Mu˜noz、Dimitrios Chrysostomou、Simon Bøgh 和 Nestor Arana-Arexolaleiba。skrl：用于强化学习的模块化灵活库。机器学习研究期刊，24(254):1–9，2023年。网址 http://jmlr.org/papers/v24/23-0112.html。

Yang Song、Jascha Sohl-Dickstein、Diederik P Kingma、Abhishek Kumar、Stefano Ermon 和 Ben Poole。通过随机微分方程进行基于分数函数的生成建模。*arXiv* 预印本 *arXiv:2011.13456*，2020年。

Yifeng Tian、Nishant Panda 和 Yen Ting Lin。刘维尔流重要性采样器。*arXiv* 预印本 *arXiv:2405.06672*，2024。

Alexander Tong、Nikolay Malkin、Guillaume Huguet、Yanlei Zhang、Jarrid Rector-Brooks、Kilian Fatras、Guy Wolf 和 Yoshua Bengio。条件流匹配：无模拟动态最优传输。*arXiv* 预印本 *arXiv:2302.00482*，2(3)，2023。

Yinuo Wang、Likun Wang、Yuxuan Jiang、Wenjun Zou、Tong Liu、Xujie Song、Wenxuan Wang、Liming Xiao、Jiang Wu、Jingliang Duan 等。带熵调节器的扩散演员-评论家。神经信息处理系统进展，37：54183–54204，2024。

Zhendong Wang、Jonathan J Hunt 和 Mingyuan Zhou。扩散策略：一种用于离线强化学习的高表达能力策略类别。*arXiv* 预印本 *arXiv:2208.06193*，2022。

Kevin Zakka、Baruch Tabanpour、Qiayuan Liao、Mustafa Haiderbhai、Samuel Holt、Jing Yuan Luo、Arthur Allshire、Erik Frey、Koushil Sreenath、Lueder A. Kahrs、Carlo Sferrazza、Yuval Tassa 和 Pieter Abbeel。《MuJoCo Playground：一个用于 GPU 加速机器人学习和仿真到现实迁移的开源框架》，2025年。网址 https://github.com/google-deepmind/mujoco_playground。

Yanjie Ze、Gu Zhang、Kangning Zhang、Chenyuan Hu、Muhan Wang 和 Huazhe Xu。《3D 扩散策略：通过简单的 3D 表征实现可泛化的视觉运动策略学习》。*arXiv* 预印本 *arXiv:2403.03954*，2024。

Simon Zhai、HaoBai、Zipeng Lin、Jiayi Pan、Peter Tong、Yifei Zhou、Alane Suhr、Saining Xie、Yann LeCun、Yi Ma 等。通过强化学习将大型视觉语言模型微调为决策智能体。神经信息处理系统进展，37:110935–110971，2024。

Shiyuan Zhang、Weitong Zhang 与 Quanquan Gu。面向离线强化学习的能量加权流匹配。*arXiv* 预印本 *arXiv:2503.04975*，2025年。

大语言模型的使用

大语言模型（LLMs），具体为 ChatGPT-5，被用于两个目的：（1）辅助语法检查、语言润色并确保格式一致性；（2）用于检索与发现任务，例如查找相关工作。任何内容、想法或科学主张均未由大语言模型生成或改动。

PolicyFlow 目标近似的误差分析

我们从策略的流参数化出发，推导终端偏移的变分表达式，并引入插值路径近似。随后，我们严格分析由此产生的误差，证明其为一阶误差，并在 PPO 框架内论证该近似的有效性。

A.1 终端偏移的变分公式

设由随机向量生成的两个流满足

$$\dot{\varphi}_t(\mathbf{z};\mathbf{s}) = v_t(\varphi_t(\mathbf{z};\mathbf{s});\mathbf{s}), \tag{17}$$

$$\dot{\hat{\varphi}}_t(\mathbf{z};\mathbf{s}) = \hat{v}_t(\hat{\varphi}_t(\mathbf{z};\mathbf{s});\mathbf{s}), \tag{18}$$

其初始条件为 $\varphi_0(\mathbf{z};\ \mathbf{s}) = \hat{}\varphi_0(\mathbf{z};\ \mathbf{s}) = \mathbf{z}$。将终端偏移设为 $\delta_{\varphi_t}(\mathbf{z};\ \mathbf{s}) :=\ \varphi_t(\mathbf{z};\ \mathbf{s}) - \hat{}\varphi_t(\mathbf{z};\ \mathbf{s})$，速度场变化设为 $\delta_{v_t}(\mathbf{x};\ \mathbf{s}) :=\ v_t(\mathbf{x};\ \mathbf{s}) - \hat{}v_t(\mathbf{x};\ \mathbf{s})$。
在参考轨迹 $\hat{}\varphi_t$ 附近对 v_t 进行线性化，可得变分方程

$$\dot{\delta}_{\varphi_t} = \boldsymbol{J}_t\delta_{\varphi_t} + \delta_{v_t}(\hat{\varphi}_t;\mathbf{s}), \quad \text{with} \quad \delta_{\varphi_0} = \mathbf{0}, \tag{19}$$

其中，雅可比矩阵定义为 $\boldsymbol{J}_t := \partial_{\mathbf{x}}\hat{}v_t(\mathbf{x};\ \mathbf{s})\Big|_{\mathbf{x}=\hat{\varphi}_t}$。

设 $\boldsymbol{\Phi}(1,\ t)$ 是 $\boldsymbol{\Phi}(\tau,\ t) = \boldsymbol{J}_\tau\boldsymbol{\Phi}(\tau,\ t)$ 的基本矩阵，其中 $\boldsymbol{\Phi}(t,\ t) = \boldsymbol{I}$。那么，精确终端偏移为

$$\delta_{\varphi_1}(\mathbf{z};\mathbf{s}) = \int_0^1 \boldsymbol{\Phi}(1,t)\delta_{v_t}\big(\hat{\varphi}_t(\mathbf{z};\mathbf{s});\mathbf{s}\big)\mathrm{d}t. \tag{20}$$

A.2 插值路径近似与误差分析

我们使用更简单的线性插值路径来近似式（20）中的复积分：

$$\mathbf{x}_t := (1-t)\mathbf{z} + t\hat{\varphi}_1(\mathbf{z};\mathbf{s}), \quad t \in [0,1]. \tag{21}$$

终端偏移的近似为 $\tilde{\delta}_{\varphi_1} := \int_0^1 \delta_{v_t}(\mathbf{x}_t;\mathbf{s})\mathrm{d}\mathbf{t}$。该近似的误差为 $E = \delta_{\varphi_1} - \delta_{\varphi_1}$。我们将此误差分解为两个组成部分：

$$E = \underbrace{\int_0^1 (\boldsymbol{\Phi}(1,t) - \boldsymbol{I})\delta_{v_t}(\hat{\varphi}_t)\mathrm{d}t}_{E_1} + \underbrace{\int_0^1 \big(\delta_{v_t}(\hat{\varphi}_t) - \delta_{v_t}(\mathbf{x}_t)\big)\mathrm{d}t}_{E_2}. \tag{22}$$

假设如下：

（A1）速度场 $\hat{v}_t$ 在 $\mathbf{x}$ 中为 $\mathcal{C}^1$，其雅可比矩阵一致有界，$\|\boldsymbol{J}_t\| \le L_v$。（A2）速度变化 δ_{v_t} 一致有界，$\|\delta_{v_t}(\mathbf{x};\mathbf{s})\| \le \epsilon$，并且关于 $\mathbf{x}$ 满足 Lipschitz 条件，其常数为 $L_\delta = \mathcal{O}(\epsilon)$，且在 $(t,\mathbf{s})$ 上一致成立。关于 L_δ 的假设是合理的，因为该变化本身源于规模为 ϵ 的小幅策略更新。

第一项误差 E_1 源于用单位矩阵 $\boldsymbol{I}$ 近似基本矩阵 $\boldsymbol{\Phi}(1,t)$。由于 $\|\boldsymbol{\Phi}(1,t) - \boldsymbol{I}\| = \mathcal{O}(1)$ 且 $\|\delta_{v_t}\| = \mathcal{O}(\epsilon)$，该项的大小为：

$$\|E_1\| \le \int_0^1 \|\boldsymbol{\Phi}(1,t) - \boldsymbol{I}\| \cdot \|\delta_{v_t}(\hat{\varphi}_t)\| \mathrm{d}t = \int_0^1 \mathcal{O}(1) \cdot \mathcal{O}(\epsilon) \mathrm{d}t = \mathcal{O}(\epsilon). \tag{23}$$

第二项误差 E_2 源于用线性路径 $\mathbf{x}_t$ 替代真实轨迹 $\hat{\varphi}_t$。路径偏差 $\|\hat{\varphi}_t - \mathbf{x}_t\|$ 通常为 $\mathcal{O}(1)$。根据关于 δ_{v_t} 的利普希茨假设：

$$\|E_2\| \le \int_0^1 L_\delta \|\hat{\varphi}_t - \mathbf{x}_t\| \mathrm{d}t = \int_0^1 \mathcal{O}(\epsilon) \cdot \mathcal{O}(1) \mathrm{d}t = \mathcal{O}(\epsilon). \tag{24}$$

由于两个误差分量都是一阶的，因此总近似误差也是一阶的。于是，我们修正正文中的结论：

$$\delta_{\varphi_1}(\mathbf{z};\mathbf{s}) = \int_0^1 \delta_{v_t}(\mathbf{x}_t;\mathbf{s}) \mathrm{d}t + \mathcal{O}(\epsilon). \tag{25}$$

A.3 似然比的一阶误差

令 $\mathrm{n} := \mathrm{a} - \hat{\varphi}_1(\mathrm{z};\mathrm{s})$，$\mathrm{u} := \delta_{\varphi_1}(\mathrm{z};\mathrm{s})$，且 $\tilde{\mathrm{u}} := \int_0^1 \delta_{v_t}(\mathrm{x}_t;\mathrm{s})\mathrm{d}t$。根据式（25），我们有 $\mathrm{u} = \tilde{\mathrm{u}} + \mathcal{O}(\epsilon)$。

对高斯对数密度进行泰勒展开可知，均值的一阶误差会导致对数似然的一阶误差：

$$\log p_n(\mathbf{n};\tilde{\mathbf{u}},\boldsymbol{\sigma}^2) - \log p_n(\mathbf{n};\mathbf{u},\boldsymbol{\sigma}^2) = \mathbf{n}^\top \left(\frac{\tilde{\mathbf{u}} - \mathbf{u}}{\boldsymbol{\sigma}}\right) + \mathcal{O}(\epsilon^2) = \mathbf{n}^\top \left(\frac{\mathcal{O}(\epsilon)}{\boldsymbol{\sigma}}\right) = \mathcal{O}(\epsilon). \tag{26}$$

利用詹森不等式，可得

$$\log p_n(\mathbf{n};\tilde{\mathbf{u}},\boldsymbol{\sigma}^2) \ge \int_0^1 \log p_n(\mathbf{n};\delta_{v_t}(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2)\,\mathrm{d}t = \mathbb{E}_{p(t)}\left[\log p_n(\mathbf{n};\delta_{v_t}(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2)\right]. \tag{27}$$

因此，

$$\mathbb{E}_{p(t)}\left[\log p_n(\mathbf{n};\delta_{v_t}(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2)\right] - \log p_n(\mathbf{n};\mathbf{u},\boldsymbol{\sigma}^2) = \mathcal{O}(\epsilon). \tag{28}$$

因此，重要性比率的对数误差为一阶：

$$\left|\mathbb{E}_{p(t)}\left[\log \frac{p_n(\mathbf{n};\delta_{v_t}(\mathbf{x}_t;\mathbf{s}),\boldsymbol{\sigma}^2)}{p_n(\mathbf{n};\mathbf{0},\hat{\boldsymbol{\sigma}}^2)}\right] - \log \frac{p_n(\mathbf{n};\mathbf{u},\boldsymbol{\sigma}^2)}{p_n(\mathbf{n};\mathbf{0},\hat{\boldsymbol{\sigma}}^2)}\right| = \mathcal{O}(\epsilon). \tag{29}$$

尽管这是对重要性比率误差的一个较宽松的界，但在使用 PPO 式代理目标函数进行策略改进时，它足以确保算法在实践中的有效性。一阶近似误差意味着，对于较小的更新，利用近似目标函数计算出的梯度与精确目标函数的梯度十分接近。PPO 式裁剪机制会自然限制更新幅度 ϵ，从而最大限度地减小这一线性误差项的影响，并保持训练的稳定性。因此，这种近似仍是优化连续归一化流策略的一种有效且计算高效的方法。

B 随机插值中分数函数与速度场的关系

考虑流匹配中使用的随机插值器：

$$\mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0 + \gamma(t)\mathbf{z}, \qquad \mathbf{x}_0 \sim p_0,\ \mathbf{x}_1 \sim p_1,\ \mathbf{z} \sim \mathcal{N}(0,I). \tag{30}$$

Albergo 等人（2023）推导了沿插值路径的得分函数 $\nabla_{\mathbf{x}}$ 对数 p_t（$\mathbf{x}_t$）与条件流匹配中使用的速度场 v_t（$\mathbf{x}_t$）之间的关系（见其论文中的公式（2.27））：

$$v_t(\mathbf{x}_t) = \bar{v}_t(\mathbf{x}_t) - \dot{\gamma}(t)\,\gamma(t)\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t), \tag{31}$$

其中

$$\bar{v}_t(\mathbf{x}_t) = \mathbb{E}_t[\mathbf{x}_1 - \mathbf{x}_0 \,|\, t\mathbf{x}_1 + (1-t)\mathbf{x}_0 + \gamma(t)\mathbf{z} = \mathbf{x}_t]\,. \tag{32}$$

一般而言，无法为 $\bar{v}_t(\mathbf{x}_t)$ 求得闭式表达式，因此无法精确解析随机插值器下得分函数与速度场之间的关系。然而，当 $\gamma(t) = 0$时，插值器会退化为整流流的确定性路径，此时两者之间的关系便显式可得。因此，我们采用确定性恒等式近似 $\bar{v}_t(\mathbf{x}_t)$：

$$\bar{v}_t(\mathbf{x}_t) \approx \frac{(1-t)\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) + \mathbf{x}_t}{t}. \tag{33}$$

Sub将这一近似代入 $v_t(\mathbf{x}_t)$ 的表达式，可得一个近似关系 p
bet随机速度场与得分函数之间的关系：

$$v_t(\mathbf{x}_t) \approx \frac{(1-t)\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) + \mathbf{x}_t}{t} - \dot{\gamma}(t)\,\gamma(t)\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t). \tag{34}$$

最后，选择常用的随机性调度

$$\gamma(t) = \sqrt{2(1-t)t}, \tag{35}$$

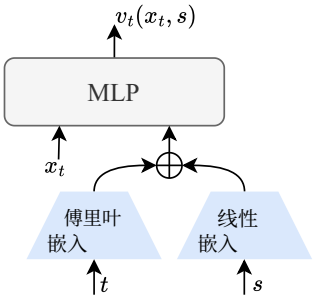
即可得到选项卡4中汇总的表达式。

C 实验细节

我们的算法基于开源框架`SKRL` (Serrano-Mu˜noz等人，2023年)和`RSL-RL` (Schwarke等人，2025年)。具体而言，我们继承了来自`SKRL`的灵活回放缓冲区实现，并将其与`RSL-RL`所提供的近端策略优化实现框架相集成。

C.1 模型架构

PolicyFlow中使用的模型基于一个流网络，该网络将噪声输入映射为动作，并以状态或其他上下文为条件。该模型包含四个主要组成部分：



流网络（多层感知机）：一个多层感知机，以加噪动作以及时间和观测嵌入作为输入来预测速度。

时间步嵌入（傅里叶嵌入）：使用一组固定的频率，将标量噪声/时间步编码为高维表示。该嵌入的计算方式为

$$\mathbf{t}_{\text{emb}} = \text{MLP}([\cos(2\pi f_i t), \sin(2\pi f_i t)]_{i=1}^{d/2})$$

从而使模型能够更好地捕捉时间依赖关系。

观测嵌入（线性层）：将观测向量嵌入固定维度空间的线性层，该嵌入会与时间步嵌入相加，从而调制流网络输出。

可学习方差：与近端策略优化中的高斯策略参数化相同。

C.2 多目标设置

MultiGoal 环境是一个二维方形工作空间，旨在评估智能体学习多模态且均衡的目标到达行为的能力。六个固定目标均匀分布在以原点为中心的圆上，且与起始位置的距离相等。在每个回合开始时

，智能体会在工作空间内随机初始化，并且必须尽可能高效地学会到达最近的目标。

该环境被建模为一个二阶系统：状态同时包含位置和速度，动作对应一个二维加速度向量。奖励由两个部分组成：鼓励智能体接近最近目标的基于距离的项，以及抑制过度控制输入的动作惩罚。通过结合这两项，该环境能够一致地评估目标到达准确度和控制效率。

表 5 和表 6 总结了主要的环境配置参数，以及状态、观测和奖励函数的定义。

MultiGoal 任务的智能体配置总结于表 7 和表8中。PolicyFlow 和近端策略优化共享学习率、折扣因子、GAE 参数和裁剪设置等常见超参数。与近端策略优化相比，PolicyFlow 引入了一个额外项，即布朗正则化损失，而近端策略优化采用标准的高斯熵正则项。

表 5：MultiGoal 环境配置

参数	**值**
最大回合长度	50
动作裁剪范围	[-5.0, 5.0]
积分时间步长 (δt)	0.1
检查墙壁碰撞	True
工作空间 x-限制	[-8.0, 8.0]
工作空间 y-限制	[-8.0, 8.0]
目标数量	6
目标半径	5.0
目标中心	[0.0, 0.0]
距离奖励系数	1.0
动作惩罚系数	0.01

表 6：MultiGoal 环境：状态/观测和奖励函数

组成部分	**描述 / 公式**
观测 / 状态	
状态向量 $\mathbf{s}$	$\mathbf{s} = [x, y, v_x, v_y]$
演员 / 评论家观测	$\mathbf{o}_{\text{actor}} = \mathbf{o}_{\text{critic}} = \mathbf{s}$
动作	
动作 $\mathbf{a}$	二维加速度：$\mathbf{a} = [a_x, a_y]$ （限制在 $[-5, 5]$ 内）
奖励函数	
距离奖励	$r_{\text{distance}} = \exp(-0.3\, d_{\text{min}}^2) + \exp(-0.1\, d_{\text{min}}^2)$, 其中 $d_{\text{min}} = \min_i \lVert\mathbf{p} - \mathbf{g}_i\rVert$ 是到最近目标的距离 $\mathbf{g}_i$
动作惩罚	$r_{\text{action}} = -0.01\, \lVert\mathbf{a}\rVert^2$
总奖励	$r_{\text{total}} = r_{\text{distance}} + r_{\text{action}}$

表 7：MultiGoal 的近端策略优化与 PolicyFlow 超参数配置

参数	PolicyFlow	PPO
目标 KL 散度	0.01	0.01
学习率	2×10^{-4}	2×10^{-4}
折扣因子 γ	0.99	0.99
广义优势估计 λ	0.95	0.95
时间限制自举	True	True
小批量	4	4
学习轮数	5	5
高斯熵损失缩放	0.001	0.001
布朗正则化损失缩放	0.25	—
比率裁剪 ϵ	0.2	0.2
裁剪预测值	True	True
价值裁剪	0.2	0.2
价值损失缩放	1.0	1.0
二阶龙格-库塔法的时间步积分	12	
演员隐藏层	[256, 128, 64]	[256, 128, 64]
评论家隐藏层	[256, 128, 64]	[256, 128, 64]
Actor/Critic 激活函数	Mish	Mish
并行环境数量	1024	1024
每个环境的轨迹采样数	24	24

表 8：流匹配策略优化（FPO）在 MultiGoal 环境上的超参数设置。

参数	值
欧拉积分步数	12
时间嵌入维度	8
每个动作的样本数	8
指数运算前的平均损失	True
在训练中使用离散时间步	True
比率更新的裁剪 epsilon	0.05
训练批次大小	1024
折扣因子 γ	0.99
每回合最大步数	50
学习率	1×10^{-4}
并行环境数量	4096
每次更新的小批量数	32
环境总步数	1.2×10^{8}
学习轮数	16
GAE 参数 λ	0.95
价值损失缩放	0.25
演员隐藏层	[256, 128, 64]
评论家隐藏层	[256, 128, 64]
Actor/Critic 激活函数	Mish

C.3 ISAACLAB

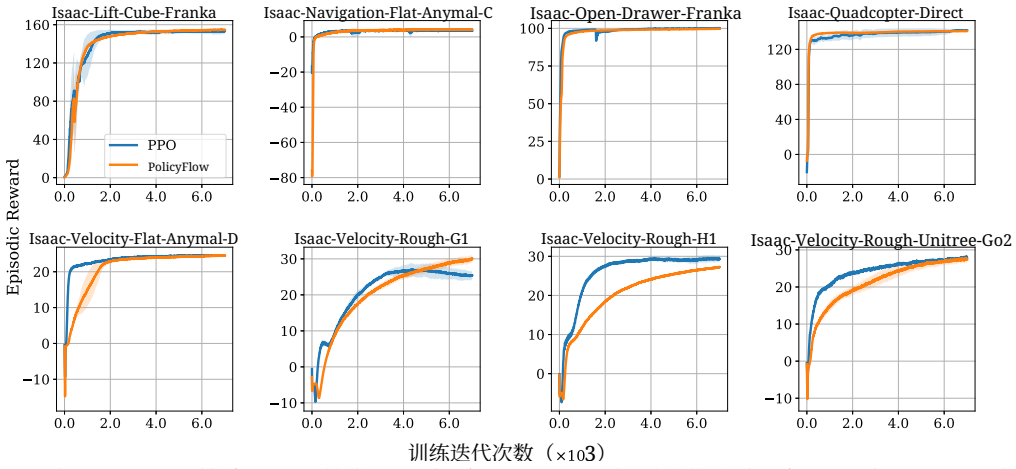


图5：在 IsaacLab 基准测试上的学习曲线（PolicyFlow 与近端策略优化）。图中展示了训练迭代次数（x 轴）对应的平均回合奖励（y 轴）及其标准误，结果取 5 个随机种子的平均值。

表9：在所有 IsaacLab 基准测试中，近端策略优化和 PolicyFlow 使用的通用超参数设置

参数	PPO	PolicyFlow
二阶 Runge–Kutta 积分的时间步数	–	12
积分		
并行环境数量	4096	4096
自适应学习率	True	True
期望KL目标	0.01	0.01
GAE 参数 λ	0.95	0.95
时间限制自举	True	True
每次更新的小批量数	4	4
学习轮数	5	5
比率裁剪 ϵ	0.2	0.2
裁剪预测值	True	True
价值裁剪	0.2	0.2
价值损失缩放	1.0	1.0
梯度范数裁剪	1.0	1.0
Actor/Critic 激活函数	ELU	ELU

Tab表 10：三个 IsaacLab 基准测试中近端策略优化和 PolicyFlow 的可变超参数设置 s.

参数	Lift-Cube-Franka		Navigation-ANYmal-C		Quadcopter-Direct	
	PPO	PolicyFlow	PPO	PolicyFlow	PPO	PolicyFlow
演员/评论家隐藏层	[256,128,64]	[256,128,64]	[128,128]	[128,128]	[64,64]	[128,128]
初始学习率	1×10^{-4}	1×10^{-4}	1×10^{-3}	1×10^{-3}	5×10^{-4}	5×10^{-4}
折扣因子 γ	0.98	0.98	0.99	0.99	0.99	0.99
高斯熵系数 w_g	0.006	0.004	0.005	0.0025	0.01	0.005
布朗正则化系数 w_b	–	0.002	–	0.0025	–	0.005
时间/观测嵌入维度	–	64	–	64	–	64
每个环境的轨迹采样数	24	24	8	8	24	24

表 11：PPO 和 PolicyFlow 在三个 IsaacLab 基准测试中的可变超参数设置。

参数	Flat-ANYmal-D		Rough-H1		Rough-Unitree-Go2	
	PPO	PolicyFlow	PPO	PolicyFlow	PPO	PolicyFlow
演员/评论家隐藏层	[128,128,128]	[128,128,128]	[512,256,128]	[512,256,128]	[512,256,128]	[512,256,128]
初始学习率	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
折扣因子 γ	0.99	0.99	0.99	0.99	0.99	0.99
高斯熵系数 w_g	0.005	0.0025	0.01	0.005	0.01	0.008
布朗正则化系数 w_b	–	0.0025	–	0.005	–	0.002
时间/观测嵌入维度	–	64	–	512	–	512
每个环境的轨迹采样数	24	24	24	24	24	24

Tab表 12：PPO 和 PolicyFlow 在 IsaacLab 两个基准测试中的可变超参数设置 .

参数	Open-Drawer-Franka		Rough-G1	
	PPO	PolicyFlow	PPO	PolicyFlow
演员/评论家隐藏层	[256,128,64]	[256,128,64]	[512,256,128]	[512,256,128]
初始学习率	5×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}
折扣因子 γ	0.99	0.99	0.99	0.99
高斯熵系数 w_g	0.001	0.0008	0.008	0.002
布朗正则化系数 w_b	–	0.0002	–	0.006
时间/观测嵌入维度	–	64	–	256
每个环境的轨迹采样数	96	96	24	24

C.4 MUJOCO PLAYGROUND

表 13：MuJoCo Playground 基准测试中 PolicyFlow 的常见超参数 ks.

参数	值
折扣因子 γ	0.995
GAE 参数 λ	0.95
时间限制自举	True
自适应学习率	否
每次更新的小批量数	4
学习轮数	16
裁剪预测值	否
价值损失缩放	0.25
二阶 Runge–Kutta 积分的时间步数	12
演员隐藏层	[32, 32, 32, 32]
评论家隐藏层	[256, 256, 256, 256, 256]
Actor/Critic 激活函数	SiLU
并行环境数量	1024
每个环境的轨迹采样数	24

表14：MuJoCo Playground 基准测试中 PolicyFlow 的可变超参数（1/2).

参数	WalkerWalk	FingerSpin	BallInCup	CartpoleBalance	FishSwim
学习率	1×10^{-3}	5×10^{-5}	3×10^{-4}	5×10^{-5}	3×10^{-4}
高斯熵系数 w_g	0.001	0.01	0.002	0.001	0.008
布朗正则化系数 w_b	0.001	0.001	0.001	0.0001	0.002
比率裁剪 ϵ	0.2	0.2	0.2	0.01	0.2
梯度范数裁剪	1.0	10.0	1.0	0.5	5.0

表 15：MuJoCo Playground 基准测试上 PolicyFlow 的可变超参数（2/2）。

参数	FingerTurnHard	FingerTurnEasy	CheetahRun	ReacherEasy	ReacherHard
学习率	4×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
高斯熵系数 w_g	0.006	0.002	0.002	0.002	0.002
布朗正则化系数 w_b	0.001	0.001	0.001	0.001	0.001
比率裁剪 ϵ	0.2	0.05	0.05	0.05	0.05
梯度范数裁剪	-1.0	10.0	10.0	5.0	5.0