面向强化学习的优化器学习

Qingfeng Lan *
计算机科学系，阿尔伯塔大学埃德蒙顿，阿尔伯塔省，加拿大
qlan3@ualberta.ca

A. Rupam Mahmood
阿尔伯塔大学计算机科学系
CIFAR人工智能讲席，Amii
armahmood@ualberta.ca

Shuicheng YanSea AI Lab、
Skywork AI
shuicheng.yan@gmail.com

Zhongwen XuSea AI Lab,
腾讯人工智能实验室
zhongwen.s.xu@gmail.com

arXiv:2302.01470v3 [cs.LG] 4 Jun 2024

摘要

近年来，学习型优化器通过利用更多数据、计算资源和多样化任务，在监督学习中取得了显著成功，表现优于经典手工设计的优化器。强化学习（RL）与监督学习本质上不同；在实践中，即使面对简单的强化学习任务，这些学习型优化器也无法发挥良好效果。我们研究了这一现象，并找出了两个问题。首先，智能体梯度分布是非独立同分布的，导致元训练效率低下。此外，由于智能体与环境的交互具有高度随机性，智能体梯度存在较大的偏差和方差，增加了为强化学习学习优化器的难度。为解决这些问题，我们提出了流水线训练方法和一种具有良好归纳偏置的新型优化器结构，使得从头开始为强化学习学习优化器成为可能。我们证明，尽管学习型优化器仅在玩具任务上训练，却能够泛化至 Brax 中未见过的复杂任务。 [1]

1 引言

深度学习在许多领域都取得了巨大成功（LeCun 等人，2015年），这在很大程度上归功于自动学习到的特征，它们优于手工设计的专家特征。梯度下降使模型中的参数能够自动调整，从而生成效果显著的特征。尽管取得了这些进展，作为深度学习的另一个重要组成部分，优化器在很大程度上仍由人工设计，并且高度依赖专家知识。为减轻手工设计优化器的负担，研究人员提出借助元学习来学习如何进行优化（Sutton，1992年；Andrychowicz 等人，2016年；Chen 等人，2017年；Wichrowska 等人，2017年；Maheswaranathan 等人，2021年）。与依靠人类专家知识设计优化器相比，学习优化器是一种数据驱动方法，能够减少对专家知识的依赖。在训练过程中，可以对学习型优化器进行优化，以加快学习速度并帮助获得更好的性能。

尽管学习型优化器取得了显著进展，但以往的工作仅提出了用于监督学习（SL）的学习型优化器。这些学习型优化器通常具有复杂的神经网络结构，并融入了大量人工设计的输入特征，因此需要投入大量计算资源和人力来设计和训练。此外，研究表明，它们在强化学习（RL）任务中表现不佳（Metz 等人，2020b；2022b）。面向强化学习的学习优化仍是一个开放且具有挑战性的问题。

*这项工作是在新加坡 Sea AI Lab 实习期间完成的。[1]代码可在 https://github.com/sail-sg/optim4rl/获取。

经典优化器通常针对监督学习任务进行设计，然后应用于强化学习任务。然而，强化学习任务具有经典优化器在很大程度上忽视的独特属性。例如，不同于监督学习，强化学习智能体的输入分布由于局部相关的状态转移动态而呈现非平稳性和非独立同分布特性（Alt 等人，2019年）。此外，由于策略和价值迭代，强化学习中的目标函数和损失景观在学习过程中不断变化，导致优化过程更加不稳定且复杂。在某些情况下，这些属性还使得直接将为监督学习设计的优化算法应用于强化学习变得不恰当，例如过时的累积梯度（Bengio 等人，2020a）或独特的干扰-泛化现象（Bengio 等人，2020b）。我们仍然缺乏专门为强化学习任务设计的优化器。

在本研究中，我们旨在为强化学习学习优化器。我们不通过研究强化学习优化来手动设计优化器，而是应用元学习，从智能体与环境的交互所生成的数据中学习优化器。我们首先研究了这一问题，发现复杂的智能体梯度分布阻碍了强化学习学习型优化器的训练。此外，智能体梯度分布的非独立同分布特性也妨碍了元训练。最后，高度随机的智能体与环境的交互可能导致具有较大偏差和方差的智能体梯度，从而进一步加剧学习强化学习优化器的难度。针对这些挑战，我们提出了一种新方法 *Optim4RL*：一种用于强化学习的学习型优化器，包含流水线训练和具有良好归纳偏置的专用优化器结构。与以往方法相比，Optim4RL 的训练更加稳定，在优化强化学习任务方面也更加有效，同时无需复杂的优化器结构或大量人工设计的输入特征。我们证明，Optim4RL 能够从头开始学习优化强化学习任务，并推广到未见过的任务。我们的工作首次提出了一种在实践中表现良好的、用于深度强化学习任务的学习型优化器。

2 背景

2.1 强化学习

强化学习（RL）的过程可以形式化为一个马尔可夫决策过程（MDP）。形式上，设 $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}\ r, \gamma)$ 为一个 MDP，它包含状态空间 $\mathcal{S}$、动作空间 $\mathcal{A}$、状态转移概率函数 $\mathrm{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$、奖励函数 $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$，以及折扣因子 $\gamma \in [0, 1)$。在每个时间步 t，智能体观察一个状态 $S_t \in \mathcal{S}$，并根据策略 $\pi(\cdot|S_t)$采样一个动作 A_t。随后，它根据 P 观察下一个状态 $S_{t+1} \in \mathcal{S}$，并获得标量奖励 $R_{t+1} = r(S_t, A_t)$。回报定义为奖励的加权和，即 $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1}$。状态价值函数 $v_\pi(s)$定义为从状态 s 开始时的期望回报。智能体旨在找到最优策略 π^*，以最大化期望回报。

近端策略优化（PPO）（Schulman 等人，2017年）和优势演员-评论家（优势演员-评论家）（Mnih 等人，2016年）是两种广泛用于连续控制的强化学习算法。近端策略优化通过使用裁剪代理目标，防止策略在每个时间步发生过大变化，从而提高训练稳定性。优势演员-评论家是演员-评论家方法的一种变体（Sutton 和 Barto，2018年），其特点是采用多智能体并行训练和同步梯度更新。在这两种算法中，v_π 由 v近似，后者通常由神经网络参数化。实践中，使用时序差分（TD）学习来近似 v_π:

$$v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)), \tag{1}$$

其中 α 是学习率，S_t 和 S_{t+1} 是两个连续的状态，而 $R_{t+1} + \gamma v(S_{t+1})$ 称为时序差分目标。由于状态值不断变化、状态转移复杂以及奖励信号带有噪声，时序差分目标通常存在偏差、非平稳且带有噪声（Schulman 等人，2016年）。它们通常会导致损失景观在训练过程中不断变化。因此，智能体梯度 [2] 通常具有较大的偏差和方差（Lan 等人，2022年），这可能导致性能不理想，甚至无法收敛。

[2]本文中，我们使用“智能体梯度”一词来指学习智能体中所有参数的梯度，其中可能包括策略梯度、价值函数的梯度以及其他超参数的梯度。

2.2 通过元学习学习优化

我们的目标是通过元学习来学习一个优化器。设 θ 为我们要优化的强化学习智能体的智能体参数。一个（学习得到的）优化器定义为更新函数 U，它将输入梯度映射为参数更新，并由元网络实现，其参数为元参数 ϕ。设 z 为该元网络的输入，其中可能包括梯度 g、损失 L、梯度的指数移动平均等。设 h 为用于存储历史值的优化器状态。由此，我们可以计算智能体参数更新 $\Delta\theta$ 以及更新后的智能体参数 θ':

$$\Delta\theta, h' = U_\phi(z, h) \text{ and } \theta' = \theta + \Delta\theta.$$

注意，所有经典的一阶优化器都可以写成这种形式，其中包含 $\phi = \emptyset$。例如，对于随机梯度下降，$h' = h = \emptyset$、$z = g$ 和 $U_{\text{SGD}}(g, \emptyset) = (-\alpha g, \emptyset)$，其中 α 是学习率。对于 RMSProp（Tieleman & Hinton，2012年），设 $z = g$；h 用于存储平方梯度的平均值。于是 $U_{\text{RMSProp}}(g, h) = (-\frac{\alpha g}{\sqrt{h'+\epsilon}}, h')$，其中 $h' = \beta h + (1-\beta)g^2$, $\beta \in [0, 1]$，而 ϵ 是一个很小的正数，用于保证数值稳定性。

与Xu 等人(2020年) 类似，我们采用双层优化来优化 θ 和 ϕ。首先，我们收集$M+1$轨迹$\mathcal{T} = \{\tau_i, \tau_{i+1}, \cdots \tau_{i+M-1}, \tau_{i+M}\}$。对于内层更新，我们固定 ϕ，并通过最小化内层损失，对 θ 执行多步梯度下降更新 L^{inner}。具体而言，对于每条轨迹 $\tau_i \in \mathcal{T}$，有

$$\Delta\theta_i \propto \nabla_\theta L^{\text{inner}}(\tau_i; \theta_i, \phi) \text{ and } \theta_{i+1} = \theta_i + \Delta\theta_i,$$

其中，$\nabla_\theta L^{\text{inner}}$ 是 θ的智能体梯度。将上述过程重复 M 次，我们得到$\theta_i \xrightarrow{\phi} \theta_{i+1} \cdots \xrightarrow{\phi} \theta_{i+M}$。这里，$\theta_{i+M}$是 ϕ的函数。为简单起见，我们滥用记号，仍使用 θ_{i+M}。接下来，我们使用 τ_{i+M} 作为验证轨迹，通过外层损失 L^{outer}来优化 ϕ：

$$\Delta\phi \propto \nabla_\phi L^{\text{outer}}(\tau_{i+M}; \theta_{i+M}, \phi) \text{ and } \phi' = \phi + \Delta\phi,$$

其中，$\nabla_\phi L^{\text{outer}}$ 是 ϕ的元梯度。由于 θ_{i+M} 是 ϕ的函数，我们可以应用链式法则，并借助自动微分软件包计算元梯度 $\nabla_\phi L^{\text{outer}}$。

3 相关工作

我们的工作与三个领域密切相关：强化学习中的优化、通用强化学习算法的发现，以及监督学习中的学习优化。

3.1 强化学习中的优化

Henderson 等人 (2018年) 在强化学习中测试了不同的优化器，并指出，经典的自适应优化器可能无法始终考虑强化学习算法与环境之间复杂的交互。 Sarigül 与 Avci (2018年) 对深度强化学习中的不同动量策略进行了基准测试，发现 Nesterov 动量的泛化能力更强。 Bengio 等人 (2020a) 更进一步指出，与监督学习不同，时序差分学习中的动量会因参数更新变化和自举而变得双重滞后。通过校正时序差分学习中的动量，可以提高样本效率。 这些工作共同表明，若不考虑强化学习的独特属性，直接将监督学习中的优化方法用于强化学习未必总是合适。与这些为强化学习手工设计新优化器的工作不同，我们采用数据驱动方法，应用元学习，从智能体与环境交互中生成的数据中学习强化学习优化器。

3.2 发现通用强化学习算法

数据驱动方法也被用于探索通用强化学习算法。例如，Houthooft 等人（2018年）提出对损失函数进行元学习，使其能够考虑智能体的历史记录，并大幅提升学习效率。同样，Kirsch 等人（2020年）提出了 MetaGenRL,

它使用离策略二阶梯度为确定性策略学习目标函数。Oh 等人（2020年）运用元学习，通过与一组环境交互，发现了完整的强化学习更新规则。不同于发现完整的更新规则，Lu 等人（2022年）专注于利用进化策略探索镜像学习空间，并展示了在未见过的设置中的泛化能力。Bechtle 等人（2021年）在元训练期间将额外信息纳入参数化损失函数，并将该方法应用于图像分类、行为克隆和基于模型的强化学习。Kirsch 等人（2022年）探索了对称性在发现新强化学习算法中的作用，并表明对称性能够提升泛化能力。Jackson 等人（2023年）考察了环境设计对强化学习中元学习更新规则的影响，并开发了一种自动化的对抗性环境设计方法，以提升所学习强化学习算法的分布内鲁棒性和泛化性能。沿着这些工作的思路，我们秉持数据驱动的精神，旨在学习优化器，而非通用强化学习算法。我们的训练和评估流程也在很大程度上受到这些工作的启发。

3.3 在监督学习中学习优化

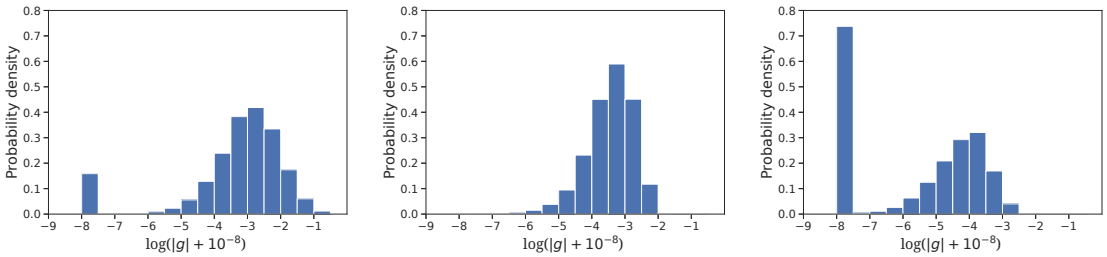
最初，学习优化仅用于调整学习率（Jacobs，1988年；Sutton，1992年；Mahmood 等人，2012年）。最近，研究人员开始从头开始学习一个完整的优化器。Andrychowicz 等人（2016年）使用长短期记忆网络实现了学习型优化器（Hochreiter 与 Schmidhuber，1997年），并表明学习型优化器能够泛化到未见过的任务。Li 与 Malik（2017年）采用引导式策略搜索方法来寻找良好的优化器。Wichrowska 等人（2017年）提出了一种分层循环神经网络（RNN）架构（Medsker 与 Jain，2001年），大幅降低了内存和计算开销，并且已证明能够泛化到不同的网络结构。Metz 等人（2022a）开发了基于多层感知器的学习型优化器，在内存、计算开销和性能之间实现了更好的平衡。

学习型优化器以难以训练而闻名。部分原因在于，它们通常通过截断时间反向传播进行训练，这会导致严重的有偏梯度或梯度爆炸。为了解决这些问题，Metz 等人(2019年提出了一种方法，动态地对重参数化梯度估计器和进化策略式梯度估计器赋予不同权重，从而稳定学习型优化器的训练。Vicol 等人(2021年)通过将计算图划分为截断展开，并利用进化策略和梯度偏差校正计算无偏梯度，解决了这些问题。Harrison 等人(2022年)研究了优化算法的训练稳定性，并提出通过添加来自 Adam（Kingma & Ba，2015年）和 AggMo（Lucas 等人，2019年）的自适应标称项，来提高学习型优化器的稳定性。Metz 等人(2020a)通过在大量任务上训练优化器，并投入大量计算资源，训练出了通用优化器。本着同样的理念，Metz 等人(2022b)继续开展大规模优化器训练，投入更多计算资源（4，000 TPU 月）并采用更多样化的监督学习任务。学习型优化器 VeLO 无需超参数调优，并且在广泛的监督学习任务中表现出色。VeLO 是学习优化领域长期研究的珍贵成果，凝聚了几代人的智慧与努力。尽管 VeLO 标志着学习型优化器在监督学习任务中取得了里程碑式的成功，但如《Metz 等人(2022b）》第4.4.4节所示，它在强化学习任务中的表现仍然不佳。

VeLO 在强化学习任务中的失败表明，为强化学习设计学习型优化器仍是一项艰巨的挑战。不同于以往专注于为监督学习学习优化器的工作，我们旨在面向强化学习进行学习优化。正如我们接下来将展示的，我们的方法简单、稳定且有效，无需使用复杂的神经网络结构，也不必融入大量人工设计特征。据我们所知，我们的工作首次证明了学习型优化器在深度强化学习任务中的成功。

4 面向强化学习的优化器学习中的问题

监督学习的学习型优化器出了名地难以训练，常受到严重的训练不稳定性困扰（Wichrowska 等人，2017年；Metz 等人，2019年；2020a）。为强化学习学习优化器则更加困难（Metz 等人，2022b）。接下来，我们将指出强化学习学习优化中的两个问题。



(a) 在训练开始时　(b) 在训练中期　(c) 训练结束时

图 1：智能体梯度分布的可视化：(a) 训练开始时，(b) 训练中期，以及 (c) 训练结束时。所有智能体梯度均在使用 RMSProp 优化器训练 big_密集_long 时收集。我们计算 $\log(|g| + 10^{-8})$，以避免对非正智能体梯度应用 log 函数时出错。

4.1 智能体梯度分布是非独立同分布的

在强化学习中，学习型优化器将智能体梯度 g 作为输入，并输出智能体参数更新 $\Delta\theta$。为探究在强化学习中学习优化器的难度，我们在网格世界（即大型_密集_长，详情见附录B ）中使用 RMSProp（Tieleman & Hinton，2012年）训练一个优势演员-评论家智能体，并收集不同训练阶段的智能体梯度。我们在图 x中以对数1轴绘制这些智能体梯度。 y轴表示概率密度。显然，智能体梯度分布是非独立同分布的，并且会在整个训练过程中发生变化。具体而言，在训练开始时，智能体梯度分布有两个峰值。训练中期，大多数智能体梯度都非零，并集中在 10^{-3}附近。训练结束时，很大一部分智能体梯度为零。众所周知，在许多设置中，非独立同分布的输入分布会使训练更加不稳定，并降低学习性能（Ma 等人，2022年；Wang 等人，2023年；Khetarpal 等人，2022年）。同样，违反独立同分布假设也会增加学习型优化器训练中的学习不稳定性并降低效率。请注意，这个问题在监督学习和强化学习的学习优化中都存在。然而，强化学习中的智能体梯度分布通常比监督学习中的梯度分布更加非独立同分布，因为强化学习任务本质上更具非平稳性。详情请参阅附录D。

4.2 双层优化的恶性螺旋

在优化模型参数的同时学习优化器属于双层优化，容易受到训练不稳定性严重影响（Wichrowska 等人，2017年；Metz 等人，2020a；Harrison 等人，2022年）。在强化学习中，由于智能体与环境的交互具有高度随机性，智能体梯度存在较大的偏差和方差，使双层优化更加不稳定。

具体而言，在监督学习中，通常假设训练集由独立同分布的样本组成。然而，强化学习中的输入数据分布是非独立同分布的，这使得整个训练过程更加不稳定和复杂，尤其是在涉及学习优化时。在大多数监督学习设置中，真实标签没有噪声且不随时间变化。例如，手写数字2 在 MNIST（Deng，2012年）中的真实标签是 $y = 2$，并且在训练期间不会改变。在强化学习中，时序差分学习（见公式（1））得到广泛应用，时序差分目标在其中发挥着类似于监督学习中标签的作用。与监督学习中的标签不同，由于智能体与环境之间的交互具有高度随机性，时序差分目标存在偏差、非平稳且带有噪声。这导致损失景观在训练过程中不断变化，并可能引发致命三元组（Van Hasselt 等人，2018年）和容量损失（Lyle 等人，2021年）。此外，在监督学习中，较低的损失通常意味着更好的性能（例如，更高的分类准确率）。但在强化学习中，由于损失景观不断变化，较低的外层损失并不一定能很好地表明性能更佳（即回报更高）。再加上有偏差的时序差分目标，以及状态转移、奖励信号和智能体与环境交互带来的随机性，智能体梯度的偏差和方差都相对较高。在学习

强化学习的优化方法时，元梯度会受到智能体梯度高偏差和高方差所引入的大量噪声影响。由于元梯度存在噪声且不准确，学习型优化器的改进过程既不稳定又缓慢。使用性能不佳的优化器，无法再保证策略改进。表现不佳的智能体不太可能收集到“高质量”数据来提升智能体和学习型优化器的性能。最终，这种双层优化陷入恶性循环：糟糕的优化器 $\rightarrow$ 糟糕的智能体策略 $\rightarrow$ 收集到低质量的数据 $\rightarrow$ 糟糕的优化器 $\rightarrow \cdots$。

5 Optim4RL：一种用于强化学习的学习型优化器

为解决第 4 节中的问题，我们提出了一种用于强化学习的学习型优化器，名为 *Optim4RL*，它结合了流水线训练和一种新颖的优化器结构。正如我们接下来将展示的，与以往方法相比，Optim4RL 的训练更加稳健且高效。

5.1 流水线训练

在图 1 中，我们展示了训练期间智能体梯度分布呈非独立同分布。一般而言，优秀的优化器应当在整个训练过程中，面对不同的智能体梯度分布都能良好运行。为了使智能体梯度分布更接近独立同分布，我们提出了流水线训练。

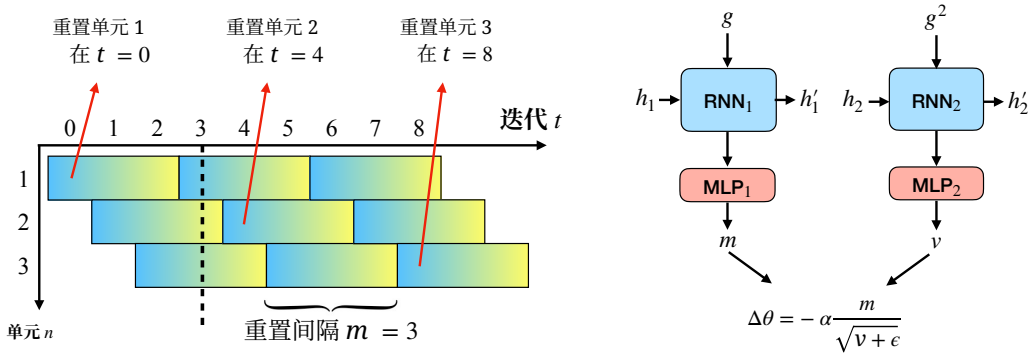
我们不再只训练一个智能体，而是并行训练 n 个智能体，每个智能体都有自己的任务和优化器状态。这三个要素共同构成一个训练单元（智能体、任务、优化器状态）；总共有 n 个训练单元。令 m 为我们称作重置间隔的正整数。一个完整的训练间隔持续 m 次训练迭代。在图 2 (a) 中，我们展示了使用 $m = n = 3$ 进行流水线训练的示例。为了有效地训练优化器，学习型优化器的输入包含来自所有 n 个训练单元的智能体梯度。训练前，我们选择 n 个整数 $\{r_1, \cdots \ r_n\}$，使它们在区间 $[0, m-1]$ 内均匀分布。然后，对于 $i \in \{1, \cdots \ n\}$，将 r_i 分配给训练单元 i。在训练迭代 t 时，如果 $r_i \equiv t \ (\text{mod } m)$，则重置训练单元 i。通过定期重置训练单元，可以确保在迭代 t 时，我们能够访问覆盖一个训练间隔的训练数据。例如，在 $t = 3$ 时，输入包含来自区间起始处单元 1 的智能体梯度、来自区间结束处单元 2 的智能体梯度，以及来自区间中间处单元 3 的智能体梯度，如图 2 (a) 中的虚线所示。通过流水线训练，输入的智能体梯度更加多样，并分布于整个训练间隔，使输入分布更接近独立同分布。理想情况下，我们希望 $m \leq n$，这样输入便包含来自所有训练阶段的智能体梯度。在我们的实验中，n 是训练环境的数量；m 取决于每个任务的训练步数，其数量级与 n 相近。

5.2 改进学习型优化器的归纳偏置

最近，Harrison 等人（2022年）证明，在学习型优化器中加入自适应项，可以提高优化含噪二次模型时的训练稳定性。实验中，Harrison 等人（2022年）表明，加入 Adam（Kingma& Ba，2015年）和 AggMo（Lucas 等人，2019年）中的项，也能提高学习型优化器的稳定性。然而，加入人工设计特征不仅会使优化器更加复杂，也违背了学习优化的理念——理想的学习型优化器应能够自动学习有用特征，尽可能减少对人类专家知识的依赖。我们不直接引入自适应优化器中的项，而是采用与自适应优化器类似的形式来设计参数更新函数：

$$\Delta\theta = -\alpha \frac{m}{\sqrt{v + \epsilon}}, \tag{2}$$

其中 α 是学习率，ϵ 是一个较小的正数，而 m 和 v 是双循环神经网络处理后的输出，如图2（b）所示。具体而言，对于每个输入梯度 g，我们生成两个标量 o_1 和 o_2。然后设置 $m = g_{sign} \exp(o_1)$ 和 $v = \exp(o_2)$，其中 $g_{sign} \in \{-1, \ 1\}$ 是 g的符号。更多细节见算法1。



(a) 流水线训练　　(b) Optim4RL 的优化器结构

图 2：（a）流水线训练示例，其中重置间隔为 $m = 3$，训练单元数量为 $n = 3$。所有训练单元都会定期重置，以使训练数据多样化。（b）Optim4RL 的网络结构。g 是输入的智能体梯度，h_i 和 h_i' 是隐藏状态，α 是学习率，ϵ 是一个较小的正数常量，$\Delta\theta$ 是参数更新。

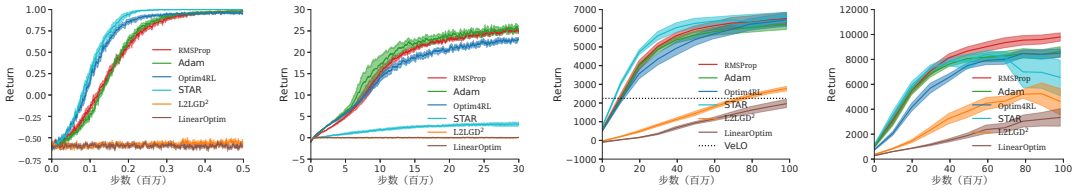
通过将参数更新函数参数化为公式(2)的形式，我们通过为学习型优化器选择合适的假设空间，减少神经网络近似平方根和除法的负担，从而改善学习型优化器的归纳偏置。一般来说，我们希望在合理的假设空间中学习一个良好的优化器。该空间应足够大，能够涵盖尽可能多的优良优化器，例如 Adam（Kingma & Ba，2015年）和 RMSProp（Tieleman&Hinton，2012年）。与此同时，它也应排除不良选择，以便高效地找到合适的候选项。采用公式（2）形式的优化器恰好满足这两项要求。此外，神经网络通常难以准确近似数学运算（Telgarsky，2017年；Yarotsky，2017年；Boullé 等人，2020年；Lu 等人，2021年）。借助公式（2），神经网络可以将全部表达能力和容量用于学习 m 和 v，从而减轻近似平方根和除法的负担。

最后，我们结合这两种技术，提出了我们的方法——一种用于强化学习的学习型优化器（Optim4RL）。遵循 Andrychowicz 等人（2016年），我们的优化器也以坐标为单位对智能体参数进行操作，因此所有智能体参数共享同一个优化器。除了梯度之外，许多先前为监督学习设计的学习型优化器还将人工设计特征作为输入，例如多个时间尺度上的梯度值移动平均、平方梯度移动平均，以及 Adafactor 风格的累加器（Shazeer& Stern，2018年）。理论上，这些特征可以在 Optim4RL 的循环神经网络隐藏状态中学习并存储。因此，为求简便，我们仅将智能体梯度作为输入。接下来我们将展示，尽管方法简单，我们的学习型优化器 Optim4RL 在许多强化学习任务中都取得了令人满意的性能，优于若干学习型优化器。

6 实验

本节首先验证 Optim4RL 能否从头开始学习强化学习优化。随后，我们展示如何训练通用的强化学习学习型优化器。更多实验结果见附录 E。

继 Oh 等人（2020年）的工作之后，我们设计了若干具有不同属性的网格世界，例如不同的时域、奖励函数或状态-动作空间。更多详情见附录 B。除网格世界外，我们还在 Catch（Osband 等人，2020年）和 Brax 任务（Freeman 等人，2021年）中测试了我们的方法。我们主要考虑两种强化学习算法——优势演员-评论家（Mnih 等人，2016年）和近端策略优化（Schulman 等人，2017年）。对于所有实验，我们在网格世界中训练优势演员-评论家，在 Brax 任务中训练近端策略优化。对于 Optim4RL，由于资源限制，我们选择了一个包含两个门控循环单元（Cho 等人，2014）的小型网络



(a) Catch (b) 大_密集_长 (c) Ant (d) Humanoid

图 3：不同优化器在四个强化学习任务中的优化性能。请注意，VeLO 的性能是根据Metz 等人 (2022b) 中图 11 (a) 估算得出的。所有其他结果均为10 次运行的平均值，阴影区域表示90% 置信区间。Optim4RL 是唯一一个在所有任务中都取得令人满意性能的学习型优化器。

，其隐藏层尺寸为 8；两个多层感知机均包含两个尺寸为 16 的隐藏层。我们使用 Adam 优化学习型优化器。更多实现细节见附录 C。

6.1 从头开始学习强化学习优化器

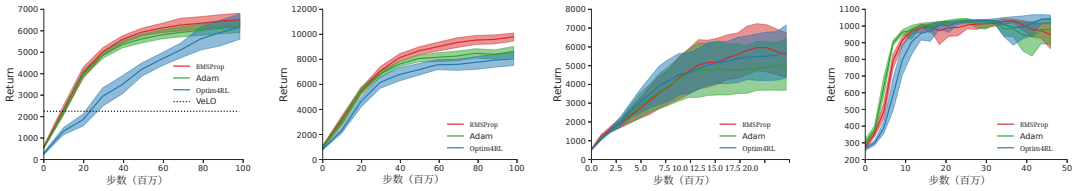
我们首先展示了在强化学习任务中从头开始训练 Optim4RL 是可行的，而用于监督学习的学习型优化器在强化学习任务中的表现并不总是理想。我们将经典优化器（Adam 和 RMSProp）与学习型优化器（L2LGD^2 (Andrychowicz 等人，2016年）、STAR（Harrison 等人，2022年）和 VeLO（Metz 等人，2022b）作为基线。除 VeLO 外，我们在一个任务中对优化器进行元学习，然后在该特定任务中测试固定的学习型优化器。优化器的优化性能以回报衡量，回报取 10 次运行的平均值，如图10 所示，见图3。总体而言，L2LGD^{2}在全部四个任务中均告失败。在 Catch 中，STAR 和 Optim4RL 的表现都优于经典优化器（Adam 和 RMSProp），收敛速度也更快。在 Ant 中，Optim4RL 和 STAR 表现相当不错，与 Adam 和 RMSProp 不相上下，同时显著优于最先进的优化器 VeLO。然而，STAR 无法在大型_密集_长任务中有效优化；在 Humanoid 中，STAR 的性能不稳定，最终还会崩溃。*Optim4RL* 是唯一一个在所有任务中都能实现稳定且令人满意性能的学习型优化器，这本身就是一项重大成就，因为它证明了我们的方法有效，并展现了其在实际应用中的潜力。

Optim4RL 归纳偏置的优势作为一项消融研究，我们将 OptimRL 与$LinearOptim$进行比较，以展示 OptimRL 归纳偏置的优势。LinearOptim 采用“线性”参数更新函数： $\Delta\theta = -\alpha(a*g+b)$，其中 α是学习率， a和 b是循环神经网络模型的输出。LinearOptim 与 Optim4RL 的唯一区别在于归纳偏置——LinearOptim 的参数更新函数是线性函数。相比之下，Optim4RL 的参数更新函数受到自适应优化器的启发（见公式（2））。如图3所示，LinearOptim 在所有任务中都未能实现优化，这验证了 Optim4RL 归纳偏置的优势。

流水线训练的有效性通过使输入的智能体梯度分布更接近独立同分布且更少依赖时间，流水线训练可以提升训练稳定性和效率。为验证这一观点，我们在表1中比较了采用和未采用流水线训练时 Optim4RL 的优化性能。我们观察到，两个网格世界任务（小型_密集_长程和大型_密集_长程）中的性能提升较小，而两个 Brax 任务（Ant 和 Humanoid）中的提升更为显著，这证实了流水线训练的有效性。

6.2 迈向通用型强化学习学习型优化器

通用优化器即使面对不同尺度的输入梯度，也应表现出良好的性能。为了元训练一个通用学习型优化器，我们首先设计了六个网格世界，使这些任务中生成的智能体梯度涵盖广泛的范围。为了展示其泛化



(a) Ant (b) Humanoid (c) 摆 (d) Walker2D

图4：尽管 Optim4RL 仅在六个简单的网格世界中从头开始训练，却展现出很强的泛化能力，并在 Brax 任务中取得了良好的性能。相比之下，VeLO（Metz 等人，2022b）经过 4,000 张量处理单元月的训练，任务数量达数千个，但在 Ant 上的性能仍未达到最优。这些结果展现了 Optim4RL 在复杂且未见过的任务中的泛化能力，这本身就是一项重大成就，证明了我们方法的有效性。

方法 \ Task	小_密集_长	大_密集_长	Ant	Humanoid
采用流水线训练	32.22±0.52	23.10±0.31	6421±355	8440±364
不进行流水线训练	30.64±0.69	22.47±0.51	5038±235	6557±1055

表 1：采用和不采用流水线训练时 Optim4RL 的性能。所有结果均为 10 次运行的平均值，并附有 90% 置信区间。

能力，接下来我们使用优势演员-评论家在这些网格世界中对 Optim4RL 进行元训练，并使用近端策略优化在 Brax 任务中对其进行测试。如图 4 所示，Optim4RL 在这些任务中取得了令人满意的性能，展现出很强的泛化能力。值得注意的是，Optim4RL 在 Ant 中显著超越了 VeLO（最先进的学习型优化器）。这是一项了不起的成果，因为 VeLO 在数千个任务上使用数千个张量处理单元月进行训练，而 Optim4RL 仅在六个玩具任务上训练了数个图形处理单元小时。最后，尽管 Optim4RL 完全从头开始训练，但与经典的人工设计优化器（Adam 和 RMSProp）相比仍具有竞争力。为强化学习任务训练一个普遍适用的学习型优化器，本质上是一项极具挑战性的任务。我们的结果展现了 *Optim4RL* 在复杂且未见过的任务中的泛化能力，这本身就是一项重大成就，证明了我们方法的有效性。

7 结论与未来工作

在这项工作中，我们分析了强化学习中的学习优化难度，并研究了学习型优化器在强化学习中的失效问题。我们的研究发现，强化学习中的智能体梯度具有非独立同分布特性，且偏差和方差都很高。为缓解这些问题，我们引入了流水线训练和一种新颖的优化器结构。结合这些技术，我们提出了一种用于强化学习的学习型优化器 Optim4RL，它可以通过元学习，从头开始优化强化学习任务。尽管 Optim4RL 仅在玩具任务上训练，却展现出对未见过的复杂任务的强大泛化能力。

针对强化学习进行学习优化是一项具有挑战性的问题。受内存和计算限制影响，我们目前的成果有限，因为只能在少量玩具任务上训练 Optim4RL。未来，借助更多计算资源和内存，我们希望将这一方法扩展到更大规模，并通过在更多任务上使用多样化的强化学习智能体进行训练，提升 Optim4RL 的性能。此外，从理论上分析学习型优化器的收敛性也是一个有趣的课题。我们希望我们的分析和提出的方法能够启发并助益未来研究，为强化学习领域更优秀的学习型优化器铺平道路。

参考文献

Bastian Alt、Adrian Šošić 和 Heinz Koeppl。《用于强化学习的相关性先验》。《神经信息处理系统进展》，第 32 卷，2019年。

Marcin Andrychowicz、Misha Denil、Sergio Gomez、Matthew W Hoffman、David Pfau、Tom Schaul、Brendan Shillingford 和 Nando De Freitas。通过梯度下降学习梯度下降。《神经信息处理系统进展》，2016年。

Sarah Bechtle、ArtemMolchanov、Yevgen Chebotar、Edward Grefenstette、Ludovic Righetti、Gaurav Sukhatme 和 Franziska Meier。通过学习得到的损失进行元学习。载于《*2020*年 第*25*届国际模式识别会议（*ICPR*）》，2021年。

Emmanuel Bengio、Joelle Pineau 和 Doina Precup。纠正时序差分学习中的动量。《*NeurIPS* 深度强化学习研讨会》，2020a。

Emmanuel Bengio、Joelle Pineau 和 Doina Precup。时序差分学习中的干扰与泛化。载于《国际机器学习会议》，2020b。

Nicolas Boullé、Yuji Nakatsukasa 和 Alex Townsend。有理神经网络。神经信息处理系统进展，33:14243–14253，2020年。

James Bradbury、Roy Frostig、Peter Hawkins、Matthew James Johnson、Chris Leary、Dougal Maclaurin、GeorgeNecula、AdamPaszke、JakeVanderPlas、SkyeWanderman-Milne 和 Qiao Zhang。JAX：Python+NumPy 程序的可组合变换，2018年。网址 http://github.com/google/jax。

Yutian Chen、Matthew W Hoffman、Sergio Gomez Colmenarejo、Misha Denil、Timothy P Lillicrap、Matt Botvinick 和 Nando de Freitas。通过梯度下降学习如何在不使用梯度下降的情况下学习。国际机器学习会议，2017年。

Kyunghyun Cho、Bart van Merriënboer、Dzmitry Bahdanau 和 Yoshua Bengio。关于神经机器翻译的性质：编码器—解码器方法。载于*SSST-8* 论文集：统计机器翻译中的句法、语义与结构第八届研讨会，2014年。

Li Deng。用于机器学习研究的手写数字图像 MNIST 数据库。*IEEE* 信号处理杂志，2012年。

C. Daniel Freeman、Erik Frey、Anton Raichuk、Sertan Girgin、Igor Mordatch 和 Olivier Bachem。Brax——用于大规模刚体仿真的可微物理引擎，2021年。网址 http://github.com/google/brax。

James Harrison、Luke Metz 和 Jascha Sohl-Dickstein。深入探究学习型优化：稳定性、鲁棒性与归纳偏差。载于神经信息处理系统进展，2022年。

Peter Henderson、Joshua Romoff 和 Joelle Pineau。我的最优解去哪儿了？策略梯度方法中梯度下降优化的实证分析。载于第*14*届欧洲强化学习研讨会，2018年。

Sepp Hochreiter 和 Jürgen Schmidhuber。长短期记忆。神经计算，1997年。

Rein Houthooft、Yuhua Chen、Phillip Isola、Bradly Stadie、Filip Wolski、OpenAI 的 Jonathan Ho 和 Pieter Abbeel。进化策略梯度。神经信息处理系统进展，2018年。

Matthew Thomas Jackson、Minqi Jiang、Jack Parker-Holder、Risto Vuorio、Chris Lu、Gregory Far-quhar、Shimon Whiteson 和 Jakob Nicolaus Foerster。通过对抗性环境设计发现通用强化学习算法。神经信息处理系统进展，2023年。

Robert A Jacobs。通过学习率自适应提高收敛速度。《神经网络》，1988年。

Khimya Khetarpal、Matthew Riemer、Irina Rish 和 Doina Precup。《迈向持续强化学习：综述与展望》。人工智能研究杂志，2022年。

Diederik P Kingma 和 Jimmy Ba。Adam：一种随机优化方法。载于国际学习表征会议，2015年。

Louis Kirsch、Sjoerd van Steenkiste 和 Juergen Schmidhuber。使用学习得到的目标函数提升元强化学习中的泛化能力。载于国际学习表征会议，2020年。

Louis Kirsch、Sebastian Flennerhag、Hado van Hasselt、Abram Friesen、Junhyuk Oh 和 Yutian Chen。将对称性引入黑盒元强化学习。载于*AAAI*人工智能会议论文集，2022年。

Qingfeng Lan、Samuele Tosatto、Homayoon Farrahi 和 Rupam Mahmood。使用奖励梯度进行无模型策略学习。载于国际人工智能与统计会议，2022年。

Yann LeCun、Yoshua Bengio 和 Geoffrey Hinton。深度学习。《自然》，2015年。

Ke Li 和 Jitendra Malik。学习优化。载于 国际学习表征会议，2017年。

Chris Lu、Jakub Kuba、Alistair Letcher、Luke Metz、Christian Schroeder de Witt 和 Jakob Foerster。发现式策略优化。神经信息处理系统进展，2022年。

Lu Lu、Pengzhan Jin、Guofei Pang、Zhongqiang Zhang 和 George Em Karniadakis。基于算子普适逼近定理，通过 DeepONet 学习非线性算子。《自然·机器智能》，2021年。

James Lucas、Shengyang Sun、Richard Zemel 和 Roger Grosse。聚合动量：通过被动阻尼实现稳定性。载于国际学习表征会议，2019年。

Clare Lyle、Mark Rowland 和 Will Dabney。理解并防止强化学习中的容量损失。载于 国际学习表征会议，2021年。

Xiaodong Ma、Jia Zhu、Zhihao Lin、Shanxuan Chen 和 Yangjie Qin。《联邦学习中解决非独立同分布数据的最新研究综述》。*Future Generation Computer Systems*，2022年。

Niru Maheswaranathan、David Sussillo、Luke Metz、Ruoxi Sun 和 Jascha Sohl-Dickstein。逆向工程学习型优化器，揭示已知和新颖的机制。神经信息处理系统进展，2021年。Ashique Rupam Mahmood、Richard S Sutton、Thomas Degris 和 Patrick M Pilarski。无需调参的步长自适应。发表于*2012*年*IEEE*声学、语音与信号处理国际会议 *(ICASSP)*，2012年。Larry R Medsker 和 LC Jain。循环神经网络。设计与应用，2001年。

Luke Metz、Niru Maheswaranathan、Jeremy Nixon、Daniel Freeman 和 Jascha Sohl-Dickstein。理解并纠正学习型优化器训练中的病态问题。发表于国际机器学习会议，2019年。

Luke Metz、Niru Maheswaranathan、C Daniel Freeman、Ben Poole 和 Jascha Sohl-Dickstein。《任务、稳定性、架构与计算：训练更有效的学习型优化器，并用它们训练自身》。*arXiv* 预印本 *arXiv:2009.11243*，2020a。

Luke Metz、Niru Maheswaranathan、Ruoxi Sun、C Daniel Freeman、Ben Poole 和 Jascha Sohl-Dickstein。《使用一千个优化任务学习超参数搜索策略》。*arXiv* 预印本 *arXiv:2002.11887*，2020b。

Luke Metz、C Daniel Freeman、James Harrison、NiruMaheswaranathan 和 Jascha Sohl-Dickstein。在<style id='1'>终身学习智能体会议上发表：《学习型优化器中的内存、计算与性能之间的实际权衡》，2022a。

Luke Metz、James Harrison、C Daniel Freeman、Amil Merchant、Lucas Beyer、James Bradbury、Naman Agrawal、Ben Poole、Igor Mordatch、Adam Roberts 等。VeLO：通过扩展规模训练通用型学习型优化器。*arXiv* 预印本 *arXiv:2211.09760*，2022b。

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。发表于国际机器学习会议，2016年。

Junhyuk Oh、Matteo Hessel、Wojciech M Czarnecki、Zhongwen Xu、Hado P van Hasselt、Satinder Singh 和 David Silver。发现强化学习算法。神经信息处理系统进展，2020年。

Ian Osband、Yotam Doron、Matteo Hessel、John Aslanides、Eren Sezener、Andre Saraiva、Katrina McKinney、Tor Lattimore、Csaba Szepesvari、Satinder Singh 等。强化学习行为套件。载于国际学习表征会议，2020年。

Mehmet Sarigül 和 Mutlu Avci。深度强化学习中不同动量技术的性能比较。信息与电信期刊，2018年。

John Schulman、Philipp Moritz、Sergey Levine、Michael Jordan 和 Pieter Abbeel。使用广义优势估计进行高维连续控制。载于国际学习表征会议，2016年。

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年。

Noam Shazeer 和 Mitchell Stern。Adafactor：具有次线性内存开销的自适应学习率。发表于国际机器学习会议，2018年。

Richard S Sutton。通过梯度下降调整偏差：delta-bar-delta 的增量版本。发表于*AAAI*人工智能会议论文集，1992年。

Richard S Sutton 和 Andrew G Barto。《强化学习：导论》。麻省理工学院出版社，第二版，2018年。

Matus Telgarsky。神经网络与有理函数。发表于国际机器学习会议，2017年。

Tijmen Tieleman 和 Geoffrey Hinton。第 6.5 讲——RMSProp：将梯度除以其近期幅度的滑动平均值。*Coursera*《神经网络：用于机器学习的神经网络》，2012年。

Hado Van Hasselt、Yotam Doron、Florian Strub、Matteo Hessel、Nicolas Sonnerat 和 Joseph Modayil。《深度强化学习与致命三元组》。*arXiv* 预印本 *arXiv:1812.02648*，2018年。

Paul Vicol、Luke Metz 和 Jascha Sohl-Dickstein。《使用持久化进化策略对展开的计算图进行无偏梯度估计》。发表于 国际机器学习会议，2021年。

Liyuan Wang、Xingxing Zhang、Hang Su 和 Jun Zhu。《持续学习综述：理论、方法与应用》。*arXiv* 预印本 *arXiv:2302.00487*，2023年。

Olga Wichrowska、Niru Maheswaranathan、Matthew W Hoffman、Sergio Gomez Colmenarejo、Misha Denil、Nando Freitas 和 Jascha Sohl-Dickstein。《可扩展且具备泛化能力的学习型优化器》。载于国际机器学习会议，2017年。

Zhongwen Xu、Hado P van Hasselt、Matteo Hessel、JunhyukOh、SatinderSingh 和 David Silver。通过在线发现的目标进行元梯度强化学习。神经信息处理系统进展，2020年。

Dmitry Yarotsky。深度 ReLU 网络近似的误差界。《神经网络》，2017年。

Optim4RL 的伪代码

Optim4RL 的伪代码展示于算法 1 中。具体而言，在我们的所有实验中，我们使用隐藏尺寸为 Cho 等人, 2014 ）的门控循环单元（ 8 ），以及隐藏尺寸为 $[16, 16]$ 的多层感知机。

算法 1 Optim4RL：用于强化学习的学习型优化器

需要： 循环神经网络$_1$ 和循环神经网络$_2$, 多层感知机$_1$ 和多层感知机$_2$, 隐藏状态 h_1 和 h_2，输入梯度 $g, \epsilon = 10^{-8}$, 学习率 α。

$g \leftarrow \perp g$ ▷ $\perp$ 表示停止梯度操作

$h_1,\ x_1 \leftarrow$ 循环神经网络$_1(h_1,\ g)$ 和 $o_1 =$ 多层感知机$_1(x_1)$

$m =$ 符号(g) 指数(o_1) ▷ 计算 m：一阶伪矩估计

$h_2,\ x_2 \leftarrow$ 循环神经网络$_2(h_2,\ g^2)$ 和 $o_2 =$ 多层感知机$_2(x_2)$

$v =$指数(o_2) ▷ 计算 v：二阶伪矩估计

$\Delta\theta \leftarrow -\alpha \frac{m}{\sqrt{v+\epsilon}}$ ▷ 计算参数更新

B 网格世界

我们遵循 Oh 等人 (2020年) 的做法，设计了 6 个网格世界。每个网格世界中都有 N 个物体。每个物体表示为 $[r, \epsilon_{\text{term}}, \epsilon_{\text{respawn}}]$。每回合开始时，物体的位置都是随机确定的；物体被收集后，会以 $\epsilon_{\text{respawn}}$ 的概率在每个时间步重新出现在随机位置。观测由一个张量 $\{0,1\}^{N\times H\times W}$组成，其中 N表示物体的数量， $H \times W$ 表示网格的尺寸。智能体有 9 种移动动作，可前往相邻位置，也可以留在原地。智能体收集到物体时，会获得相应的奖励（r ×奖励缩放系数），并以与该物体相关的 ϵ_{term}概率结束回合。默认奖励缩放系数为 1。表 2详细介绍了各个网格世界的设置。

Task \ 设置	尺寸 $(H \times W)$	物体	时域
大_稀疏_短	10×12	$2 \times [1.0, 0.0, 0.05]$, $2 \times [-1.0, 0.5, 0.05]$	50
大_稀疏_长	12×10	$2 \times [1.0, 0.0, 0.05]$, $2 \times [-1.0, 0.5, 0.05]$	500
大_密集_短	9×13	$2 \times [1.0, 0.0, 0.5]$, $2 \times [-1.0, 0.5, 0.5]$	50
大_密集_长	13×9	$2 \times [1.0, 0.0, 0.5]$, $2 \times [-1.0, 0.5, 0.5]$	500
小_密集_长	6×4	$[1.0, 0.0, 0.5]$, $[-1.0, 0.5, 0.5]$	500
小 密集 短_ _	4×6	$[1.0, 0.0, 0.5]$, $[-1.0, 0.5, 0.5]$	50

表 2：网格世界的详细设置。

C 实验细节

在本研究中，我们使用 Jax（Bradbury 等人，2018年）进行自动微分。对于网格世界中的优势演员-评论家训练，特征网络是一个隐藏层尺寸为32 的多层感知机，用于“小型”网格世界。对于“大型”网格世界，特征网络是一个卷积神经网络（CNN），具有16 个特征，卷积核尺寸为2，后接一个输出尺寸为32的多层感知机。除非另有明确说明，我们使用ReLU 作为激活函数。我们将 $\lambda = 0.95$ 设为计算 λ回报。折扣因子为 $\gamma = 0.995$。一次轨迹展开包含20 个步骤。评论家损失权重为0.5，熵权重为0.01。对于 Brax 游戏中的近端策略优化训练，我们采用 Brax 示例中的相同设置 [3]。为元学习优化器，我们在所有实验中将$M = 4$ 设为该值；也就是说，每次外层更新都进行4 次内层更新。较大的 M 可能会带来更具前瞻性的学习，但会增加内存和计算开销

[3] https://github.com/google/brax/blob/main/notebooks/training.ipynb

要求。我们将 $M = 4$ 设为一个折中值，该值在实践中也表现良好。按照常见做法（Lu 等人，2022年），我们报告10 次运行的平均结果。其他细节将在以下章节中介绍。

C.1 计算资源

我们的所有实验均可使用 V100 GPU 进行训练。由于 GPU 内存需求较大，部分实验使用了 4 个 V100 GPU。重复本研究中的所有实验结果所需的计算量应少于 1 个 GPU 年，但实际使用的计算量很难估算。

C.2 第 4 节的实现细节

我们通过使用 RMSProp 优化器，以学习率$3e-3$训练 A2C，在 big_密集_long 中运行 $30M$ 步，从而收集智能体梯度。我们按时间步将收集到的所有智能体梯度分为 30 份。随后，我们将第一份、第十六份和最后一份中的智能体梯度分别绘制为训练开始、中期和结束时的智能体梯度分布。

C.3 第 6.1 节的实现细节

对于 LinearOptim 和 L2LGD^2, ，模型由一个隐藏尺寸为 8 的门控循环单元组成，后接一个隐藏层尺寸为 $[16, 16]$ 的多层感知机。对于 STAR，我们使用 learned_optimization [4] 提供的官方实现。与监督学习设置不同，我们将权重衰减设为 0，因为在 STAR 中，正的权重衰减会导致性能大幅下降。为公平比较，我们采用流水线训练来训练所有学习型优化器。

对于 Catch，智能体的学习率为 $1e-3$；环境/训练单元的数量 n 为 64；重置间隔 m 从 {32,64} 中选取。对于 big_dense_long，智能体的学习率为 $3e-3$；环境/训练单元的数量 n 为 512；重置间隔 m 从 $\{72, 144, 288, 576\}$ 中选取。对于 Ant 和 Humanoid，智能体的学习率为 $3e-4$；环境/训练单元的数量 n 为 2048；重置间隔 m 从 $\{32, 64, 128, 256, 512\}$ 中选取。此外，为了降低内存需求，我们将小批次数量设为 8；并将价值网络的隐藏层尺寸从 $[256, 256, 256, 256, 256]$ 改为 $[64, 64, 64, 64, 64]$。

我们使用 Adam 作为元优化器，并从$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$中选择元学习率。

C.4 第 6.2 节的实现细节

Optim4RL 在 6 个网格世界中进行元训练，然后在 Brax 任务中进行测试。我们使用 Adam 作为元优化器，并从以下集合中选择元学习率：$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3\}$。环境/训练单元的数量 n 为 512。重置间隔 m 从以下集合中选择：$\{72, 144, 288, 576\}$。所有网格世界的奖励尺度见表 3。

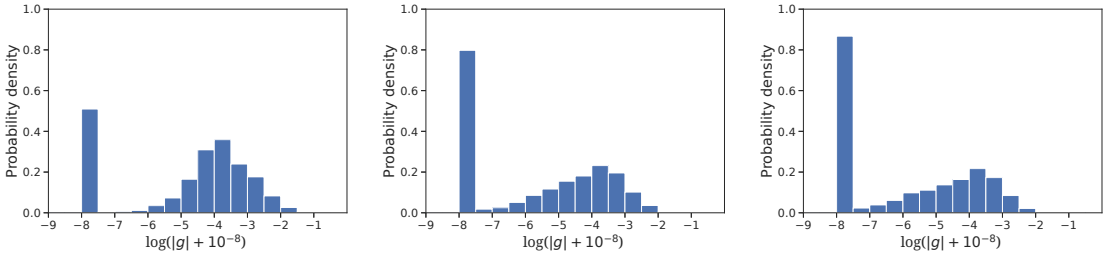
网格世界	奖励缩放系数
小_密集_长	1000
小_密集_短	100
大_稀疏_短	100
大_密集_短	10
大_稀疏_长	10
大_密集_长	1

表3：用于学习通用优化器的网格世界奖励尺度。

[4] https://github.com/google/learned_optimization/blob/main/learned_optimization/learned_optimizers/adafac_nominal.py

D Ex监督学习中梯度分布的实验 ng

本节中，我们展示监督学习中的梯度分布也具有非独立同分布特性，但其独立同分布程度高于强化学习中的智能体梯度分布。具体而言，我们使用 RMSProp 在 MNIST（Deng，2012年）上训练一个神经网络，共训练 10 轮，并收集不同训练阶段的梯度。请注意，该网络与用于在 big_密集_long 中训练优势演员-评论家的演员网络相同，唯一不同的是输出层。我们在图 5中以对数刻度绘制这些梯度。与图 1类似，梯度分布也具有非独立同分布特性，并在训练过程中不断变化。



（a）训练开始时　　（b）训练中期　　（c）训练结束时

图 5：梯度分布的可视化结果：（a）训练开始时，（b）训练中期，以及（3）训练结束时。所有梯度均采集自使用 RMSProp 优化的 MNIST 训练过程。

为了说明，MNIST 训练得到的梯度分布比在大型_密集_长篇训练中得到的智能体梯度分布更接近独立同分布，我们在表 4 中计算了不同训练阶段的（智能体）梯度分布与训练期间所有（智能体）梯度的分布之间的 Wasserstein 距离（WD）。请注意，距离越小，独立同分布程度越高。因此，这些结果支持了上述观点。

WD 值 / Task	WD（开始阶段，全部）	WD（中间阶段，全部）	WD（结束阶段，全部）
MNIST	4.0642×10^{-4}	0.7205×10^{-4}	1.4204×10^{-4}
大_密集_长	7.1583×10^{-4}	1.1726×10^{-4}	6.7967×10^{-4}

表 4：不同训练阶段的（智能体）梯度分布与训练期间所有（智能体）梯度分布之间的 Wasserstein 距离。

E 不同超参数设置下的鲁棒训练与强泛化

一般而言，我们发现学习型优化器很难训练，部分原因是在训练过程中会出现非数（NaN）错误，即使采用了梯度裁剪或梯度归一化也不例外。例如，在所有元训练超参数设置中，我们在 Humanoid 和 Ant 上分别有超过 80% 和 50% 的设置因 NaN 错误而无法训练 STAR。然而，在 Humanoid 和 Ant 上对 Optim4RL、LinearOptim 和 L2LGD^2 进行元训练时，很少遇到 NaN 错误；并且其中只有 Optim4RL 达到了令人满意的性能。

接下来，我们将展示 Optim4RL 不仅能够泛化到未见过的任务，还能迁移到不同的超参数设置。具体而言，我们在默认超参数设置下训练学习型优化器 Optim4RL，然后在两个网格世界——小型_密集_短程和大型_密集_长程——的不同超参数设置下进行测试。我们报告训练结束时的回报，并取平均值

（基于 10 次运行）。如表 5、表6和表 7所示，Optim4RL 在不同的超参数设置下都很稳健，例如广义优势估计 λ、熵权重和折扣因子。

Task	参数值	回报
小_密集_短	0.9	11.51±0.19
小_密集_短	0.95	11.25±0.16
小_密集_短	0.99	10.81±0.17
小_密集_短	0.995	10.66±0.17
大_密集_长	0.9	23.35±0.76
大_密集_长	0.95	23.57±0.60
大_密集_长	0.99	21.31±0.64
大_密集_长	0.995	20.55±0.54

表5：Optim4RL 在两个网格世界中使用不同广义优势估计 广义优势估计 λ 值 时的性能。所有结果均为 10 次运行的平均值，并报告 90%置信区间。

Task	参数值	回报
小_密集_短	0.005	11.01±0.16
小_密集_短	0.01	11.13±0.09
小_密集_短	0.02	11.29±0.12
小_密集_短	0.04	11.25±0.16
大_密集_长	0.005	22.41±0.59
大_密集_长	0.01	22.45±0.79
大_密集_长	0.02	22.59±0.43
大_密集_长	0.04	19.96±1.30

表6：Optim4RL 在两个网格世界中使用不同熵权重 时的性能。所有结果均为 10 次运行的平均值，并报告了 90%置信区间。

Task	参数值	回报
小_密集_短	0.9	12.47±0.14
小_密集_短	0.95	12.32±0.09
小_密集_短	0.99	11.48±0.09
小_密集_短	0.995	11.01±0.17
大_密集_长	0.9	18.13±3.27
大_密集_长	0.95	25.01±1.42
大_密集_长	0.99	25.45±0.47
大_密集_长	0.995	22.07±0.81

表7：Optim4RL 在两个网格世界中使用不同折扣因子时的性能。所有结果均为 10 次运行的平均值，并附有 90%置信区间。