arXiv:2302.05745v2 [cs.LG] 9 May 2023

深度学习中的泛化验证

Guy Amir*、Osher Maayan*、Tom Zelazny、Guy Katz 和 Michael Schapira

耶路撒冷希伯来大学，以色列耶路撒冷{guyam、osherm、tomz、guykatz、schapiram}@cs.huji.ac.il

摘要。 深度神经网络（深度神经网络）是深度学习的核心工具，而深度学习在众多应用领域中代表着最先进的水平。然而，基于深度神经网络的决策规则众所周知容易出现泛化不佳的问题，也就是说，在训练过程中未遇到的输入上可能表现不足。这一局限性严重阻碍了深度学习在任务关键型任务中的应用，也妨碍了它在高度多变的现实环境中的应用。我们提出了一种新颖的、由验证驱动的方法，用于识别能够很好地泛化到新输入域的基于深度神经网络的决策规则。我们通过考察 *inde-pendently trained* 深度神经网络对该定义域内输入所作决策的一致程度，来量化其对输入域的泛化能力。我们展示了如何借助深度神经网络验证的强大能力，高效而有效地实现我们的方法。我们在三个深度强化学习基准测试上评估了这种基于验证的方法，其中包括一个用于互联网拥塞控制的系统。结果证明了我们方法的实用性。更广泛地说，我们的工作提出了形式化验证的一项新颖目标，有望降低在实际环境中部署基于深度神经网络的系统所伴随的风险。

1 引言

在过去十年中，深度学习 [41] 在自然语言处理、图像识别、游戏对弈、计算生物学及许多其他领域取得了最先进的结果 [4,18,24,51,56,93,94]。然而，尽管取得了令人瞩目的成功，深度学习仍存在严重缺陷，限制了它在涉及任务关键型任务或输入高度可变的定义域中的应用。

其中一个至关重要的局限性，是深度神经网络（DNNs）难以泛化到新的输入域，即它们往往无法在与训练期间遇到的输入存在显著差异的输入上表现良好。在训练过程中，一个DNN会接收从某个输入域上的特定分布中采样得到的输入数据（“分布内”输入）。由于以下原因，所诱导的基于DNN的规则可能无法泛化到训练期间未遇到的输入：（1）DNN被置于“分布外”（OOD）环境中，即训练数据中的输入分布与DNN运行数据中的输入分布不匹配；（2）有限训练数据中对某些输入的表示不充分

[*] 两位作者贡献相同。

（例如各种低概率的边界情况）；以及（3）对训练数据中的决策规则产生“过拟合”。

近期提出的深度强化学习（DRL）应用于现实世界系统的案例，凸显了验证基于深度神经网络的决策能否泛化的重要性。[62] 在深度强化学习中，智能体以深度神经网络实现，通过反复与环境交互进行训练，从而学习一种决策策略，使其在某个目标函数（“奖励”）上达到较高性能。近期，深度强化学习已应用于许多现实世界的挑战 [22, 50, 60, 61, 70–73, 105, 117]。在许多应用领域，人们期望学到的策略能够在种类繁多的运行环境中表现良好，而这些环境的多样性不可能全部体现在训练数据中。此外，错误决策可能造成严重后果。我们对基于深度强化学习的互联网拥塞控制的讨论（见第 4.3 节）阐释了这一点。

在此，我们提出一种方法，用于识别能够泛化到感兴趣输入域上所有可能分布的基于深度神经网络的决策规则。我们的方法基于以下关键观察：一般而言，深度神经网络训练，尤其是深度强化学习策略训练，包含多种随机性因素，例如深度神经网络权重的初始化，以及训练过程中观察输入的顺序。因此，即使使用相同架构的深度神经网络，在相同数据上训练以执行完全相同的任务，通常也会学到略有不同的决策规则。借用托尔斯泰《安娜·卡列尼娜》中的说法 [102], ，我们认为：“成功的决策规则大同小异；而每条不成功的决策规则，都各有各的不成功之处。”换言之，考察多个独立训练的深度神经网络对某个输入所作的决策时，只有当它们（相似的）决策能带来高性能时，这些决策才可能一致。

鉴于上述内容，我们提出以下启发式方法，用于生成能够良好泛化到给定输入定义域的整个范围的基于深度神经网络的决策规则：独立训练多个深度神经网络，然后从中找出一个子集，使这些网络在所考虑输入定义域的所有可能输入上高度一致（根据我们的假设，这意味着这些深度神经网络学到的决策规则能够良好泛化到该定义域上的所有概率分布）。我们的评估表明（见第 4 节），这种方法极为有效，能够从一组决策规则中筛选出少数几个确实能更好地泛化到该定义域内输入的规则。由于我们的启发式方法旨在寻找对特定定义域内每一个输入都做出一致决策的深度神经网络，因此以这种方式得到的决策规则能够在该定义域内输入的不同可能分布上实现稳健的高泛化能力。

由于我们的方法需要对比不同深度神经网络在可能是无限 的输入域上的输出，因此采用形式化验证是顺理成章的。为此，我们借鉴了近期深度神经网络形式化验证领域的进展 [2, 11, 13,15, 31, 66, 84, 95, 111]。深度神经网络验证领域的研究主要聚焦于确立深度神经网络的局部对抗鲁棒性，即寻找会导致深度神经网络错误分类的微小输入扰动 [37, 42, 67]。据我们所知，我们的方法首次展示了如何拓展深度神经网络验证的适用范围，

（尽我们所知），还可以利用它来识别能够良好泛化的基于深度神经网络的决策规则。具体而言，我们展示了，对于给定的输入域，深度神经网络验证器如何为深度神经网络评分，以反映其在整个输入域上与其他深度神经网络的一致程度。这使得我们能够迭代地筛除候选深度神经网络，最终只保留彼此高度一致的网络，而这类网络往往泛化能力良好。

为了评估我们的方法，我们聚焦于三个热门的深度强化学习基准：(i) *Cartpole*，该任务要求控制一辆小车，同时保持摆锤平衡；(ii) *Mountain Car*，该任务要求控制一辆需要驶出山谷的汽车；以及 (iii) *Aurora*，一个互联网拥塞控制器。

Aurora 是一个特别适合展示我们方法的例子。Aurora 旨在应对多种多样的真实互联网环境中的网络拥塞，但它仅使用合成数据进行训练。因此，要在现实世界中部署 Aurora，确保其策略在训练输入未涵盖的众多场景下都可靠至关重要。

我们的评估结果表明，在这三种设置中，我们的验证驱动方法都能成功地根据基于深度神经网络的深度强化学习策略对分布外输入的泛化能力，对这些策略进行排序。我们的实验还表明，形式化验证优于基于梯度的方法和预测不确定性方法。这些结果展现了我们方法的潜力。我们的代码和基准测试作为随本文附带的工件公开提供 [7]。

本文其余部分安排如下。第 2 节介绍深度神经网络、深度强化学习和深度神经网络验证的背景。第 3 节介绍我们基于验证的方法，用于识别能够成功泛化到分布外输入的深度神经网络。第 4 节介绍评估结果。第 5 节讨论相关工作，而我们将在第 6 节总结。

2 背景

深度神经网络（深度神经网络） [41]是由多个层组成的有向图。在接收到其第一层（输入层）各节点的赋值后，深度神经网络会传播这些值，

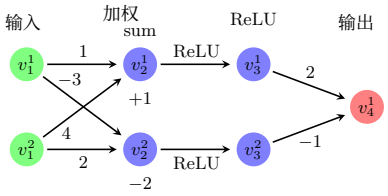


图 1：一个简单的深度神经网络。

逐层计算，直至最终得到最后（输出）层的赋值。每个节点的值根据其所在层的类型计算。例如，在加权求和层中，节点的值是与其相连的前一层节点值的仿射组合。在修正线性单元 (*ReLU*)层中，每个节点 y 计算值 $y = \text{ReLU}(x) = \max(x,0)$，其中 x 是前一层中的单个节点。有关深度神经网络及其训练的更多详情，请参见 [41]。图 1 展示了一个简单的深度神经网络。对于输入 $V_1 = [1,2]^T$，第二层计算结果（加权求和）为 $V_2 = [10,-1]^T$。随后在第三层应用 ReLU 函数，得到结果 $V_3 = [10,0]^T$。最后，网络的单个输出为 $V_4 = [20]$。

深度强化学习 (DRL) [62]是一种机器学习范式，其中以深度神经网络实现的深度强化学习智能体会在离散时间步 $t \in 0$，1，2...中与环境交互。在每个时间步，智能体都会接收环境的状态 $s_t \in \mathcal{S}$，并选择一个动作 $N(s_t) = a_t \in \mathcal{A}$。随后，环境转移到下一个状态 s_{t+1}，并向智能体提供其上一个动作所获得的奖励 r_t。智能体通过与环境反复交互进行训练，以最大化预期累积折扣奖励 $R_t = \mathbb{E}\big[\sum_t \gamma^t \cdot r_t\big]$（其中 $\gamma \in \big[0,1\big]$ 称为折扣因子） [44,91,99,100,106,116]。

深度神经网络和深度强化学习验证。 一个可靠的深度神经网络验证器 [52] 接收以下输入：(i) 一个经过训练的 深度神经网络 N；(ii) 针对深度神经网络输入的前置条件 P，用于将可能的赋值限制在感兴趣的定义域内；以及 (iii) 针对深度神经网络输出的后置条件 Q，用于限制深度神经网络的可能输出。验证器可以通过以下两种方式之一进行响应：(i) SAT，并给出一个具体输入 x'，使得 $P(x') \wedge Q(N(x'))$ 得到满足；或 (ii) UNSAT，表明不存在这样的 x'。通常， Q 编码了对于满足 P的输入而言， N期望行为的否定 。因此，SAT 结果表明深度神经网络出现错误，并且 x' 触发了一个漏洞；而 UNSAT 结果表明深度神经网络按照预期运行。附录的 B.1 小节给出了这一过程的示例。迄今为止，针对通用前馈深度神经网络 $[3,37,47,52,67,108]$, 以及在反应式环境中运行的基于深度强化学习的智能体 $[5,8,14,25,34]$，已经提出了大量验证方法。

3 通过验证量化泛化能力

我们评估深度神经网络在分布外输入上预期泛化效果的方法，依赖于“卡列宁假说”：尽管产生错误结果的方式有很多（甚至可能无限多种），但正确输出往往相当相似。因此，为了找出能够良好泛化到新输入域的基于深度神经网络的决策规则，我们主张训练多个深度神经网络，并根据其学习到的决策模型在所考察输入域上的输出与其他模型的输出对齐程度进行评分。这些分数可以通过后端深度神经网络验证器计算。我们将展示，通过迭代剔除那些倾向于与其余模型意见不一致的模型，可以有效提炼出泛化良好的深度神经网络。

首先，我们引入以下定义，以便推理两个基于深度神经网络的决策规则在某一输入域上达成一致的程度。

定义 1（距离函数）。设 $\mathcal{O}$ 为深度神经网络的可能输出空间。 $\mathcal{O}$ 的距离函数是一个函数 $d\colon \mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}^+$。

直观地说，距离函数（例如， L_1 范数）可以量化两个深度神经网络对同一输入所做决策之间的一致或分歧程度。我们在附录B中详细介绍了在不同领域可能适用的一些距离函数。

定义 2（成对分歧阈值）。设 N_1、N_2 是具有相同输出空间 $\mathcal{O}$ 的深度神经网络，d 是距离函数，Ψ 是输入域。我们将 N_1 和 N_2 的成对分歧阈值（*PDT*）定义为：

$$\alpha = PDT_{d,\Psi}(N_1, N_2) \triangleq \min\left\{\alpha' \in \mathbb{R}^+ \mid \forall x \in \Psi \colon d(N_1(x), N_2(x)) \leq \alpha'\right\}$$

该定义体现了这样一种概念：对于 Ψ、N_1 和 N_2 中的任意 输入，产生的输出之间的距离至多为 α。较小的 α 值表示，对于 Ψ 中的所有输入，N_1 和 N_2 的输出都很接近；而较大的值则表示，存在一个 Ψ 中的输入，使得决策模型出现显著分歧。

为了计算 PDT 值，我们的方法采用验证，通过二分搜索求出两个深度神经网络输出之间的最大距离；见算法1。

算法1 成对分歧阈值

输入： 深度神经网络 (N_i, N_j)，距离函数 d，输入域 Ψ，最大分歧 M > 0**输出：** PDT(N_i, N_j)

1：low $\leftarrow 0$，high $\leftarrow$ M 2：**当** (low $<$ high) **执行** 3：$\alpha \leftarrow \frac{1}{2} \cdot$ (low $+$high) 4：查询 $\leftarrow$SMT求解器 $\langle P \leftarrow \Psi, [N_i;\ N_j],\ Q \leftarrow d$ (N_i, N_j) $\geq^{-} \alpha\rangle$5：**如果** 查询结果为可满足 **则**：low $\leftarrow \alpha$6：**否则,** **如果** 查询结果为不可满足 **则**：high $\leftarrow \alpha$7：**结束循环**8：**返回** α

成对不一致阈值可以进行汇总，以衡量一个决策模型与其他决策模型集合之间的不一致程度，定义如下。

定义 3（不一致分数）。设 $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ 为一组由k 深度神经网络生成的决策模型，设 d 为距离函数，设 Ψ 为输入域。模型相对于 $\mathcal{N}$ 的不一致分数（*DS*）定义为：

$$DS_{\mathcal{N},d,\Psi}(N_i) = \frac{1}{|\mathcal{N}| - 1} \sum_{j \in [k], j \neq i} PDT_{d,\Psi}(N_i, N_j)$$

直观而言，不一致分数衡量的是单个决策模型平均而言与其余模型存在多大程度的不一致。

利用分歧分数，我们的启发式方法采用迭代方案，选取能够泛化到分布外场景的模型子集——这些场景由 Ψ 中的输入编码（见算法 2）。首先，一组 k深度神经网络 $\{N_1, N_2, \ldots, N_k\}$ 在训练数据上独立训练。接下来，调用后端验证器计算

每一对 $\binom{k}{2}$ 基于深度神经网络的模型各自的成对不一致阈值（达到一定的 ϵ 准确率）。接下来，我们的算法迭代执行：(i) 计算剩余模型子集中每个模型的分歧分数；(ii) 找出分数相对最高的模型；以及 (iii) 将其移除（算法 2 第 9 行）。算法在迭代次数超过用户定义的上限（算法 2 第 3 行），或剩余模型在输入域上“达成一致”时终止；后者由几乎相同的分歧分数表明（算法 2 第 7 行）。我们注意到，该算法还会根据用户的领域知识，获得最大差异的上界（M）。

算法2 模型选择

输入： 模型集合 $\mathcal{N} = \{N_1, \ldots, N_k\}$，最大分歧数 M，迭代次数
输出： $\mathcal{N}' \subseteq \mathcal{N}$

1: PDT $\leftarrow$ Pairwise Disagreement Thresholds($\mathcal{N}, d, \Psi$, M) ▷包含所有 PDT 的表格
2: $\mathcal{N}' \leftarrow \mathcal{N}$
3: **对** $l = 1 \ldots$次迭代 **执行**
4: **对** $N_i \in \mathcal{N}'$ **执行**
5: 当前分歧分数$[N_i] \leftarrow DS_{\mathcal{N}'}(N_i,$ PDT) ▷根据定义 3
6: **结束循环**
7: **如果** 模型分数相似(当前分歧分数)**则**：跳出循环
8: 待移除模型 $\leftarrow$ 查找分歧分数最高的模型
9: $\mathcal{N}' \leftarrow \mathcal{N}' \setminus$ 要移除的模型 ▷ 移除那些容易产生分歧的模型
10: **结束循环**
11: **返回** $\mathcal{N}'$

DS 移除阈值。确定上述用于移除模型的 DS 阈值，以及每次迭代移除多少模型（算法 2 第 8 行），可以采用不同的标准。我们在评估中采用了一种自然且简单的方法：移除具有最高 分歧分数的 p% 模型，其中 p 取某个值（在我们的评估中为 25%）。由于篇幅所限，我们将附加过滤标准的详细讨论（这些标准均证明有效）放在附录 C 中。

4 评估

我们使用三个深度强化学习基准对我们的方法进行了广泛评估。正如引言中所讨论的，验证基于深度强化学习的系统的泛化能力十分重要，因为人们通常期望此类系统在各种环境中都能稳定地保持高性能，而这些环境的多样性并未体现在训练数据中。我们的评估涵盖了两种经典的深度强化学习场景——Cartpole [16] 和 Mountain Car [74]，以及近期提出的、用于互联网流量的 Aurora 拥塞控制器 [50]。Aurora 是一个尤其有说服力的案例：它是一个相当复杂的基于深度强化学习的系统，旨在应对一项关键的现实挑战，并且必须泛化到训练数据中未涵盖的现实条件。

设置。 对于这三个深度强化学习基准中的每一个，我们首先使用相同的架构训练多个深度神经网络，训练过程之间唯一的差别是所用的随机种子。随后，我们从这些深度神经网络中剔除除那些在分布内获得较高奖励值的模型之外的所有模型（以排除决策模型仅因训练不佳而泛化能力较差的可能性）。接下来，我们为每个特定基准定义感兴趣的分布外输入域，并根据我们的框架使用算法 2 选出最有可能在这些输入域上良好泛化的模型。为了确立不同模型在实践中的实际泛化表现这一真实情况，我们随后将这些模型应用于从所考察领域中抽取的分布外输入，并根据其经验性能（平均奖励）进行排名。为考察结果的鲁棒性，我们针对该领域输入上的不同概率分布重复了最后一步。所有使用的深度神经网络均采用由两个隐藏层组成的前馈架构，使用 ReLU 激活；第一隐藏层包含 32–64 个神经元，第二隐藏层包含 16 个神经元。

结果表明，通过我们的方法选出的模型很可能表现显著更好。下面简要介绍我们的评估；附录中提供了大量补充信息。

4.1 Cartpole

Cartpole [39] 是一项广为人知的强化学习基准测试，其中智能体控制一辆小车的运动，小车顶部连接着一根倒立摆（“杆”）。小车在平台上移动，智能体的目标是尽可能长时间地保持杆平衡（见图2）。



图2：*Cartpole*：分布内设置（蓝色）和分布外设置（红色）。

智能体与环境。智能体的输入为 $s = (x,\ v_x,\ \theta,\ v_\theta)$，其中 x表示小车在平台上的位置，θ 表示杆的角度（即$|\theta| \approx 0$ 表示杆处于平衡状态，$|\theta| \approx 90°$ 表示杆处于不平衡状态），v_x 表示小车的水平速度，v_θ 表示杆的角速度。

分布内输入。训练期间，智能体的目标是在保持小车位于平台边界内的同时平衡杆。每次迭代中，智能体的单个输出表示小车在下一步的加速度（符号和幅度）。训练期间，我们将平台边界设为 $[-2.4, 2.4]$，并将小车的初始位置设为近乎静止且靠近平台中心（图 2 左侧）。具体做法是从范围 $[-0.05, 0.05]$ 内均匀抽取小车初始状态向量的各个值。

（分布外）输入域。 我们考虑一个平台比训练时使用的平台更长的输入域。具体而言，现在允许输入向量的 x 坐标覆盖更大的范围 $[-10, 10]$。对于其他输入，我们使用与训练期间相同的边界。更多细节请参见附录 A 和 B。

评估。我们训练了 $k=16$ 个模型，它们在短平台上训练时都获得了很高的奖励。接下来，我们在上述输入域上运行算法 2，直至收敛（在我们的实验中为 7 次迭代），得到一组包含 3 个模型的集合。随后，我们使用从新域抽取的分布外输入测试全部 16 个原始模型；这些输入生成的分布代表了一种新情境：小车现在位于一块更长的移位平台中央（见图 2 中的红色小车）。

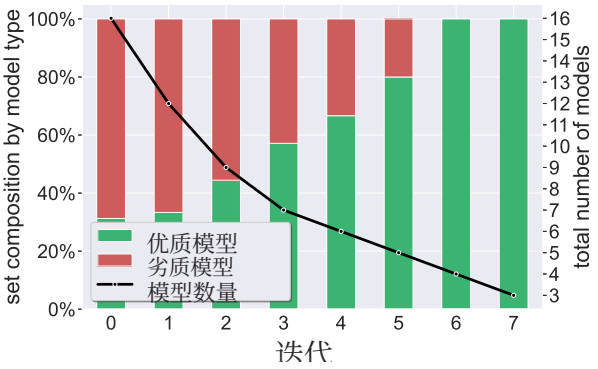


图 3：*Cartpole*：算法 *2* 每次迭代的结果：柱形表示存活模型集合中优质模型与劣质模型的比例（左侧 y 轴），曲线表示存活模型的数量（右侧 y 轴）。

其他所有参数均在分布外环境与原始训练所用的环境完全相同。图 9（见附录）展示了使用 20,000 个分布外样本评估模型的结果。在原有的 16 个模型中，11 个模型的平均奖励较低至中等，表明它们泛化到这一新分布的能力较差。只有 5 个模型获得了较高的奖励值，其中包括算法 2 找出的 3 个模型；这说明我们的方法能够有效剔除在这一分布外设置下表现不佳的其余 11 个模型（见图 3）。更多信息请参见附录 D。

4.2 Mountain Car

在第二个实验中，我们在 Mountain Car [86]基准测试上评估了我们的方法。在该基准测试中，智能体控制一辆汽车，需要学习如何逃出山谷并到达目标。与 Cartpole 实验一样，我们选取了一组在分布内表现良好的模型，并应用我们的方法找出模型子集，这些模型在预定义输入域内会做出相似的决策。我们再次在该定义域内生成分布外输入（相对于训练而言），并观察到，我们的算法所选出的模型确实比迭代剔除的其他模型泛化能力显著更强。有关这一基准测试的详细信息，请参见附录 E。

4.3 Aurora拥塞控制器

在第三个基准测试中，我们将我们的方法应用于一个复杂的真实世界系统，该系统实现了互联网拥塞控制策略。拥塞控制的目标是确定通信网络中的每个流量源应以何种速率向网络发送数据包。拥塞控制是

计算机网络领域一个众所周知的难题，也是其核心挑战 [65,75]；发送数据包过快可能导致网络拥塞，进而造成数据丢失和延迟。反之，发送速率过低则可能无法充分利用可用的网络带宽。$Aurora$ [50] 是一种基于深度强化学习的拥塞控制器，也是近期深度强化学习验证研究的对象 [8, 34]。在每个时间步，Aurora 智能体都会观察网络的统计信息，并决定下一时间步的分组发送速率。例如，如果智能体观察到网络状况极佳（如没有分组丢失），我们预期它会提高分组发送速率，以更充分地利用网络。值得注意的是，Aurora 处理的任务比经典强化学习基准测试（如 Cartpole 和 Mountain Car）难得多：拥塞控制器必须根据细微的信号，灵活应对各种可能发生的事件，而 Aurora 的输入便体现了这一点。与前述基准测试不同，在这里，要描述最优策略并不简单。

智能体与环境。 Aurora 的输入是 t向量 v_1、.....、 v_t，表示来自 t 前几个时间步的观测。智能体的单个输出值表示下一个时间步内分组发送速率的变化。每个向量 $v_i \in \mathbb{R}^3$ 包含三个不同的值，表示反映网络状况的统计数据（详情见附录 F）。按照先前的工作 [8,34, 50], ，我们将 $t = 10$ 时间步设为，因此 Aurora 的输入大小为 $3t = 30$。奖励函数是数据发送端的吞吐量、延迟和分组丢失的线性组合，这些指标由智能体观测得到（更多详情见 [50] ）。

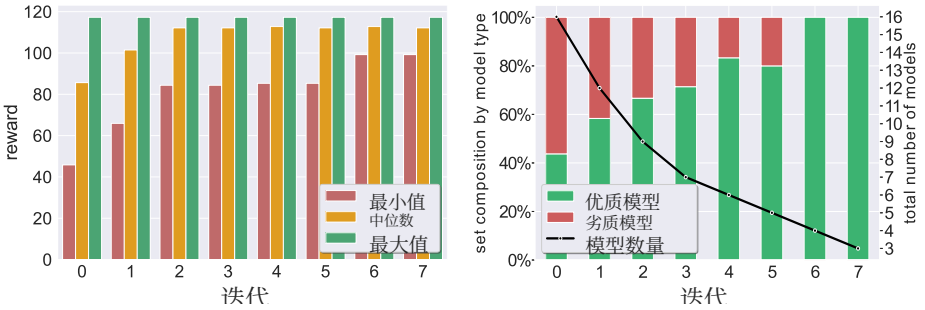
分布内输入。 Aurora 的训练是在简单的网络场景中，将拥塞控制器应用于以下情形：单个 发送端通过单条 网络链路，向单个接收端发送流量。Aurora 在不同的初始发送速率、链路带宽、链路分组丢失率、链路时延以及链路分组缓冲区大小的设定下进行训练。在训练期间，Aurora 最初以相当于链路带宽 $0.3 - 1.5$ 倍的速率发送数据包。

（分布外）输入域。 在我们的实验中，输入域描述了一条具有浅分组缓冲区的链路，这意味着网络中只能积累少量数据包（而大部分多余流量都会被丢弃），导致链路表现出波动行为。初始发送速率最高可达链路带宽的 8 倍，以此模拟可用带宽大幅下降的可能性（例如，由于竞争、流量变化等原因）。更多详情请参阅附录 F。

评估。 我们运行了算法，并根据各模型在这个大定义域上的意见分歧对其进行评分。该定义域包含模型在训练期间未曾遇到过的输入，代表了前述的新型链路条件。

实验（1）：高分组丢失。 在本实验中，我们在原始（分布内）环境中训练了 100 多个 Aurora 智能体。从中，我们选出了 $k = 16$ 个在分布内取得较高平均奖励的智能体（见图 20a）。接下来，我们在先前描述的定义域内包含的分布外输入上评估了这些智能体。训练分布与新的（分布外）分布之间的主要区别在于，初始化时可能出现极高的分组丢失率。

我们在定义域内对分布外输入进行的评估表明，尽管所有 16 个模型在分布内表现良好，但只有 7 个智能体能够成功处理此类分布外输入（见图 20b）。当我们在这 16 个模型上运行算法 2 时，它成功筛除了在分布外输入上泛化能力较差的全部 9 个模型（见图 4）。具体而言，我们的方法返回了模型 $\{16\}$，根据我们的模拟，它是表现最好的模型。值得注意的是，在最初几次迭代中，被筛除的四个模型是模型 $\{1, 2, 6, 13\}$，它们确实是在分布外输入上表现最差的四个模型（见附录 F）。



(a) *Reward statistics of remaining models* (b) *Ratio between good/bad models*

图 4: *Aurora*：算法 *2* 的逐次迭代结果。

实验（2）：分布外输入上的更多分布。 为了进一步证明，在指定的输入域中，我们的方法确实可能保留表现更好的模型，同时移除劣质模型，我们针对分布外输入上的其他分布（概率密度函数）重新进行了此前的 Aurora 实验。评估结果表明，算法 2 移除的所有模型在这些额外分布下也都取得了较低的奖励值（见附录 F）。这些结果凸显了我们方法的一项重要优势：它适用于所考虑定义域内的所有输入，因此适用于这些输入上的所有分布。

补充实验。 我们还通过调整训练过程，使其包含显著更长的交互，生成了一组新的 Aurora 模型。随后，我们重复了上述实验。结果（汇总于附录 F）表明，我们的方法再次成功选出了一组模型子集，该子集能够很好地泛化到分布外输入域上的各种分布。

4.4 与其他方法的比较

基于梯度的方法 [46, 59, 68, 69] 是能够找到满足指定约束的 DNN 输入的优化算法，与验证方法类似。这些算法因其简单性和可扩展性而极受欢迎。然而，这种优势的代价是算法本质上并不完备，且不如深度神经网络验证 [10, 110] 精确。事实上，当我们将算法修改为使用基于梯度的方法计算 PDT 分数时，结果（汇总于附录 G）表明，在我们的场景中，基于验证的方法

优于基于梯度的方法。由于基于梯度的方法并不完备， [110], 它们经常计算出次优的 PDT 值，导致泛化能力较差的模型得以保留。

预测不确定性方法 [1, 80] 是用于评估相对于观测输入的不确定性的在线方法，以确定所遇到的输入是否来自训练分布。我们进行了一项实验，将我们的方法与基于不确定性预测的模型选择进行比较：我们生成了原始模型的集成 [27,36,57]，并使用基于方差的度量（受 [64]启发）来识别在分布外采样输入上输出方差较低的模型子集。与基于梯度的方法类似，预测不确定性技术速度快且可扩展，但缺乏验证驱动的模型选择所具备的精度，也无法剔除泛化能力较差的模型。例如，在根据分布外输入上的不确定性对 Cartpole 模型进行排序时，不确定性最低的三个模型中也包括“劣质”模型，而这些模型已被我们的方法过滤掉。

5 相关工作

近年来，人们提出了大量用于确保深度神经网络正确性的方案和工具 [2, 14, 20, 28–31、35、37、38、40、42、43、47–49、52–55、58、63、67, 76, 82, 89, 92, 95, 96, 98, 101, 103, 104, 107, 109, 111, 113, 115], 包括用于深度神经网络屏蔽 [66], 、优化 [13, 97], 、定量验证 [15], 、抽象 [11, 12, 79, 84, 95, 114], 、规模缩减 [83], 等方面的技术。包括运行时监控 [45],、集成 [77, 78, 88, 112] 和其他方法 [81] 在内的非验证技术，已被用于分布外输入检测。

与上述方法不同，我们旨在针对整个输入域（涵盖该定义域上的所有分布）建立泛化保证。此外，据我们所知，我们首次尝试利用模型间的差异性，从中提炼出一个具有更强泛化能力的子集。具体而言，这也是首次将形式化验证用于这一目的的方法。

6 结论

本文介绍了一种新颖的验证驱动方法，用于识别能够良好泛化到目标输入域的深度神经网络模型。我们提出了一种迭代方案，该方案借助后端深度神经网络验证器，根据模型在给定定义域上生成相似输出的能力为模型评分。我们通过大量实验表明，该方法确实能够提炼出泛化能力良好的模型。随着 DNN验证技术日趋成熟，我们的方法将变得越来越可扩展，也能够应用于种类更加丰富的深度神经网络。

致谢。 Amir、Zelazny 和 Katz 的工作部分得到了以色列科学基金会（资助编号683/18）的支持。Amir 的工作得到了 Clore以色列基金会奖学金的支持。Maayan 和 Schapira 的工作部分得到了华为的资助支持。

参考文献

1. M. Abdar、F. Pourpanah、S. Hussain、D. Rezazadegan、L. Liu、M. Ghavamzadeh、P. Fieguth、X. Cao、A. Khosravi、U. Acharya、V. Makarenkov 和 S. Nahavandi。《深度学习中的不确定性量化综述：技术、应用与挑战》。 *Information Fusion*，76:243–297，2021。2. P. Alamdari、G. Avni、T. Henzinger 和 A. Lukina。《带有一丝魔法的形式化方法》。载于 *Proc. 20th Int. Conf. on Formal Methods in Computer-Aided*，页码 –。3. A. *Design (FMCAD)* 138 147, 2020
Albarghouthi。《 *Introduction to Neural Network Verification*》。verifieddeeplearn-ing.com, 2021。4. M. AlQuraishi。《AlphaFold at CASP13》。 *Bioinformatics*，35(22):4862–4865，2019。5. G. Amir、D. Corsi、R. Yerushalmi、L. Marzari、D. Harel、A. Farinelli 和 G. Katz。《验证基于学习的机器人导航系统》。载于 *Proc. 29th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*，页码607–627，2023。6. G. Amir、Z. Freund、G. Katz、E. Mandelbaum 和 I. Refaeli。《veriFIRE：验证工业级基于学习的野火探测系统》。载于 *Proc. 25th Int.*，页码 –。7. G. *Symposium on Formal Methods (FM)* 648 656, 2023
Amir、O. Maayan、T. Zelazny、G. Katz 和 M. Schapira。《验证深度学习中的泛化：工件》，2023。https://zenodo.org/record/7884514# .ZFAz_3ZBy3B。8. G. Amir、M. Schapira 和 G. Katz。《迈向可扩展的深度强化学习验证》。载于 *Proc. 21st Int. Conf. on Formal Methods in Computer-*，页码 –。9. G. *Aided Design (FMCAD)* 193 203, 2021
Amir、H. Wu、C. Barrett 和 G. Katz。《一种用于验证二值化神经网络的基于SMT的方法》。载于 *Proc. 27th Int. Conf. on Tools and Algorithms for*，页码 –。10. G. *the Construction and Analysis of Systems (TACAS)* 203 222, 2021
Amir、T. Zelazny、G. Katz 和 M. Schapira。《验证辅助的深度集成选择》。载于 *Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design*，页码 –。11. G. *(FMCAD)* 27 37, 2022
Anderson、S. Pailoor、I. Dillig 和 S. Chaudhuri。《优化与抽象：分析神经网络鲁棒性的协同方法》。载于 *Proc.*，页码 –。*40th ACM SIGPLAN Conf. on Programming Languages Design and Implementations (PLDI)* 731 744, 2019
12. P. Ashok、V. Hashemi、J. Kretinsky 和 S. Mohr。《DeepAbstract：用于加速验证的神经网络抽象》。载于 *Proc. 18th Int. Symp. on Automated*，页 *Technology for Verification and Analysis (ATVA)* 92 107, 2020
码 –。13. G. Avni、R. Bloem、K. Chatterjee、T. Henzinger、B. K¨onighofer 和 S. Pranger。《通过定量博弈对学习型控制器进行运行时优化》。载于 *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*，页码630 649，2019 –。14. E. Bacci、M. Giacobbe 和 D. Parker。《将强化学习验证至无穷》。载于 *Proc. 30th Int. Joint Conf. on Artificial Intelligence (IJCAI)*，2021。15. T. Baluta、S. Shen、S. Shinde、K. Meel 和 P. Saxena。《神经网络的定量验证及其安全应用》。载于 *Proc. ACM SIGSAC Conf.*，页码 –。16. A. *on Computer and Communications Security (CCS)* 1249 1264, 2019
Barto、R. Sutton 和 C. Anderson。《能够解决困难学习控制问题的类神经元自适应元件》。载于 *Proc. of IEEE Systems Man and*，页码 –。*Cybernetics Conference (SMC)* 834 846, 1983

17. S. Bassan 和 G. Katz。面向神经网络的形式化近似最小解释。载于 *Proc. 29th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*，页码 187–207, 2023
18. M. Bojarski、D. Del Testa、D. Dworakowski、B. Firner、B. Flepp、P. Goyal、L. Jackel、M. Monfort、U. Muller、J. Zhang、X. Zhang、J. Zhao 和 K. Zieba。端到端学习自动驾驶汽车，2016年。技术报告。http://arxiv.org/abs/1604.07316。19. G. Brockman、V. Cheung、L. Pettersson、J. Schneider、J. Schulman、J. Tang 和 W. Zaremba。OpenAI Gym，2016年。技术报告。https://arxiv.org/abs/1606.01540。20. R. Bunel、I. Turkaslan、P. Torr、P. Kohli 和 P. Mudigonda。分段线性神经网络验证的统一视角。载于 *Proc. 32nd Conf. on Neural Information Processing Systems (NeurIPS)*，页码 4795–4804, 2018
21. M. Casadio、E. Komendantskaya、M. Daggitt、W. Kokke、G. Katz、G. Amir 和 I. Refaeli。神经网络鲁棒性作为验证属性：一个有原则的案例研究。载于 *Proc. 34th Int. Conf. on Computer Aided Verification (CAV)*，页码 219–231，2022年。
22. W. Chen、Y. Xu 和 X. Wu。用于多资源多机器作业调度的深度强化学习，2017。技术报告。http://arxiv.org/abs/1711.07440。23. E. Cohen、Y. Elboher、C. Barrett 和 G. Katz。神经网络验证中的更紧致抽象查询。载于 *Proc. 24th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)* 2023，。24. R. Collobert、J. Weston、L. Bottou、M. Karlen、K. Kavukcuoglu 和 P. Kuksa。几乎从零开始的自然语言处理。*Journal of Machine Learning Research (JMLR)* 12 2493–2537, 2011，：–。25. D. Corsi、E. Marchesini 和 A. Farinelli。深度强化学习中安全关键任务的神经网络形式化验证。载于 *Proc. 37th Conf. on Uncertainty in Artificial Intelligence (UAI)*，页码 333–343, 2021
26. D. Corsi、R. Yerushalmi、G. Amir、A. Farinelli、D. Harel 和 G. Katz。通过基于场景的编程实现机器人学中的约束强化学习，2022年。技术报告。https://arxiv.org/abs/2206.09603。
27. T. Dietterich。机器学习中的集成方法。载于 *Proc. 1st Int. Workshop on Multiple Classifier Systems (MCS)*，页码 1–15, 2020
28. G. Dong、J. Sun、J. Wang、X. Wang 和 T. Dai。面向正确修复神经网络，2020年。技术报告。http://arxiv.org/abs/2012.01872。29. S. Dutta、X. Chen 和 S. Sankaranarayanan。利用回归多项式规则推理对神经反馈系统进行可达性分析。载于 *Proc. 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*，页码 157–168，2019年。30. S. Dutta、S. Jha、S. Sankaranarayanan 和 A. Tiwari。利用前馈神经网络对反馈控制系统进行学习与验证。*IFAC-PapersOnLine* 51 16 151–156, 2018，（）：–。
31. R. Ehlers。分段线性前馈神经网络的形式化验证。载于 *Proc. 15th Int. Symp. on Automated Technology for Verification and Analysis (ATVA)*，页码 269–286, 2017
32. Y. Elboher、E. Cohen 和 G. Katz。利用残差推理进行神经网络验证。载于 *Proc. 20th Int. Conf. on Software Engineering and Formal Methods (SEFM)*，页码 173–189, 2022

33. Y. Elboher、J. Gottschlich 和 G. Katz。基于抽象的神经网络验证框架。载于 *Proc. 32nd Int. Conf. on Computer Aided Veri-*，页码 –。 *fication (CAV)* 43 65, 2020
34. T. Eliyahu、Y. Kazak、G. Katz 和 M. Schapira。验证学习增强系统。载于 *Proc. Conf. of the ACM Special Interest Group on Data Communica-*
tion on the Applications Technologies Architectures and Protocols for Computer 、 、 、，页 *Communication (SIGCOMM)* 305 318, 2021
码 –。35. N. Fulton 和 A. Platzer。通过形式化方法实现安全强化学习：通过证明与学习迈向安全控制。载于 *Proc. 32nd AAAI Conf. on*，。36. M. *Artificial Intelligence (AAAI)* 2018
Ganaie、M. Hu、A. Malik、M. Tanveer 和 P. Suganthan。集成深度学习：综述。 *Engineering Applications of Artificial Intelligence*, 115 105151，2022：。37. T. Gehr、M. Mirman、D. Drachsler-Cohen、E. Tsankov、S. Chaudhuri 和 M. Vechev。AI2：利用抽象解释进行神经网络安全性与鲁棒性认证。载于 *Proc. 39th IEEE Symposium on Security and Privacy*，。38. C. *(S&P)* 2018
Geng、N. Le、X. Xu、Z. Wang、A. Gurfinkel 和 X. Si。迈向可靠的神经网络规范，2022年。技术报告。https://arxiv.org/abs/2210.16114。39. S. Geva 和 J. Sitte。可训练控制器倒立摆实验基准。 *IEEE Control Systems Magazine*, 13(5):40–51，1993年。40. B. Goldberger、Y. Adi、J. Keshet 和 G. Katz。利用验证对深度神经网络进行最小修改。载于 *Proc. 23rd Int. Conf. on Logic for Pro-*，，页码 –。 *gramming Artificial Intelligence and Reasoning (LPAR)* 260 278, 2020
41. I. Goodfellow、Y. Bengio 和 A. Courville。 *Deep Learning*。麻省理工学院出版社，2016年。42. D. Gopinath、G. Katz、C. P˘as˘areanu 和 C. Barrett。DeepSafe：一种评估神经网络鲁棒性的数据驱动方法。载于 *Proc. 16th. Int. Sym-*
posium on Automated Technology for Verification and Analysis (ATVA)，页码 3–19，2018年。43. E. Goubault、S. Palumby、S. Putot、L. Rustenholz 和 S. Sankaranarayanan。利用热带多面体对 ReLU 神经网络进行静态分析。载于 *Proc. 28th*，页码 –。 *Int. Symposium on Static Analysis (SAS)* 166 190, 2021
44. T. Haarnoja、A. Zhou、P. Abbeel 和 S. Levine。软演员-评论家：带有随机演员的离策略最大熵深度强化学习。载于 *Int.*，页码 –。 *Conf. on Machine Learning* 1861 1870 2018
PMLR，。45. V. Hashemi、J. K˘ret´ınsky、S. Rieder 和 J. Schmidt。用于目标检测神经网络分布外检测的运行时监控，2022年。技术报告。http://arxiv.org/abs/2212.07773。46. S. Huang、N. Papernot、I. Goodfellow、Y. Duan 和 P. Abbeel。针对神经网络策略的对抗攻击，2017。技术报告。https://arxiv. org/abs/1702.02284。47. X. Huang、M. Kwiatkowska、S. Wang 和 M. Wu。深度神经网络的安全验证。载于 *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*，页码 3–29，2017年。48. O. Isac、C. Barrett、M. Zhang 和 G. Katz。带证明生成的神经网络验证。载于 *Proc. 22nd Int. Conf. on Formal Methods in Computer-*，页码 –。 *Aided Design (FMCAD)* 38 48, 2022
49. Y. Jacoby、C. Barrett 和 G. Katz。利用不变式推断验证循环神经网络。载于 *Proc. 18th Int. Symposium on Automated Technology for*，页码 –。 *Verification and Analysis (ATVA)* 57 74, 2020

50. N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar. 互联网拥塞控制的深度强化学习视角. 收录于

Proc. 36th Int.，第 – 页。

Conf. on Machine Learning (ICML) 3050 3059, 2019

51. K. Julian, J. Lopez, J. Brush, M. Owen, and M. Kochenderfer. 飞机碰撞规避系统的策略压缩. 收录于 *Proc. 35th Digital Avionics Systems*，

Conf. (DASC) 1 10, 2016

第 – 页。 52. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex：用于验证深度神经网络的高效SMT求解器. 收录于

Proc. 29th Int.，第 – 页。

Conf. on Computer Aided Verification (CAV) 97 117, 2017

53. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex：一种用于推理深度神经网络的演算. *Formal Methods in System Design*

(FMSD) 2021 。 54. G.

Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zelji´c, D. Dill, M. Kochenderfer, and C. Barrett. 用于深度神经网络验证与分析的 Marabou 框架. 收录于 *Proc. 31st Int.*，

Conf. on Computer Aided Verification (CAV) 443 452, 2019

第 – 页。 55. B. K¨onighofer, F. Lorber, N. Jansen, and R. Bloem. 强化学习的屏蔽器综合. 收录于 *Proc. Int. Symposium on Leveraging Applications of*，，第 – 页。

Formal Methods Verification and Validation (ISoLA) 290 306, 2020

56. A. Krizhevsky, I. Sutskever, and G. Hinton. 使用深度卷积神经网络进行 ImageNet 分类 . 收录于 *Proc. 26th Conf. on Neural Information Pro-*，第 –

cessing Systems (NeurIPS) 1097 1105, 2012

页。 57. A. Krogh and J. Vedelsby. 神经网络集成、交叉验证与主动学习. 收录于

Proc. 7th Conf. on Neural Information Processing Systems，第 – 页。

(NeurIPS) 231 238, 1994

58. L. Kuper, G. Katz, J. Gottschlich, K. Julian, C. Barrett, and M. Kochenderfer. 面向可扩展的安全关键深度网络验证，2018年。技术报告。https://arxiv.org/abs/1801.05950。 59. A. Kurakin, I. Goodfellow, and S. Bengio. 物理世界中的对抗样本，2016年。技术报告。http://arxiv.org/abs/1607.02533。 60. A. Lekharu, K. Moulii, A. Sur, and A. Sarkar. 基于深度学习的自适应视频流媒体预测模型. 收录于

Proc. 12th Int. Conf. on Communication，第 – 页。

Systems & Networks (COMSNETS) 152 159 2020

IEEE，。 61. W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis. QTCP：基于强化学习的自适应拥塞控制. *IEEE Transactions on Network Science and Engineering* 6 3 445 458, 2018 ，

（）：–。 62. Y. Li. 深度强化学习：概述，2017年。技术报告。http://arxiv.org/abs/1701.07274。 63. A. Lomuscio and L. Maganti. 一种针对前馈 ReLU神经网络的可达性分析方法，2017年。技术报告。http://arxiv.org/abs/1706.07351。 64. A. Loquercio, M. Segu, and D. Scaramuzza. 深度学习中不确定性估计的通用框架. 收录于

Proc. Int. Conf. on Robotics and Automation，第 – 页。

(ICRA) 3153 3160, 2020

65. S. Low, F. Paganini, and J. Doyle. 互联网拥塞控制. *IEEE Control Systems Magazine* 22 1 28 43, 2002 ，（）：–。

66. A. Lukina, C. Schilling, and T. Henzinger. 走进未知：神经网络的主动监测. 收录于 *Proc. 21st Int. Conf. on Runtime Verification (RV)*，第42–61页，2021年。

67. Z. Lyu, C. Y. Ko, Z. Kong, N. Wong, D. Lin, and L. Daniel. Fastened Crown：收紧的神经网络鲁棒性证书. 收录于 *Proc. 34th AAAI Conf. on Artificial Intelligence (AAAI)* 5037 5044, 2020 第 – 页。 68. J. Ma, S. Ding, and Q. Mei. 朝着更实用的图神经网络对抗攻击迈进. 收录于 *Proc. 34th Conf. on Neural Information Processing Systems (NeurIPS)* 2020 。 69. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 面向能够抵御对抗攻击的深度学习模型，2017年。技术报告。http: //arxiv.org/abs/1706.06083。 70. R. Mammadli, A. Jannesari, and F. Wolf. 通过深度强化学习进行静态神经编译器优化. 收录于 *Proc. 6th IEEE/ACM Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC) and Workshop on Hierarchical Parallelism for Exascale Computing (HiPar)* 1 11, 2020 , 第 – 页。 71. H. Mao, M. Alizadeh, I. Menache, and S. Kandula. 利用深度强化学习进行资源管理. 收录于 *Proc. 15th ACM Workshop on Hot Topics in Networks (HotNets)* 50 56, 2016 , 第 – 页。 72. H. Mao, R. Netravali, and M. Alizadeh. 使用 Pensieve 实现神经自适应视频流. 收录于 *Proc. Conf. of the ACM Special Interest Group on Data Communication on the Applications Technologies Architectures and Protocols for Computer Communication (SIGCOMM)* 197 210, 2017 ,,,, 第 – 页。 73. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 使用深度强化学习玩 Atari，2013年。技术报告。https://arxiv.org/abs/1312.5602。 74. A. Moore. 用于机器人控制的高效基于记忆的学习，1990年。剑桥大学。 75. J. Nagle. IP/TCP 互联网络中的拥塞控制. *ACM SIGCOMM Computer Communication Review* 14 4 11 17, 1984 , （）：–。 76. T. Okudono, M. Waga, T. Sekiyama, and I. Hasuo. 通过对状态空间进行回归，从循环神经网络中提取加权自动机. 收录于 *Proc. 34th AAAI Conf. on Artificial Intelligence (AAAI)* 5037 5044, 2020 , 第 – 页。 77. L. Ortega, R. Caba˜nas, and A. Masegosa. 神经网络集成的多样性与泛化. 收录于 *Proc. 25th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)* 11720 11743, 2022 , 第 – 页。 78. I. Osband, J. Aslanides, and A. Cassirer. 用于深度强化学习的随机化先验函数. 收录于 *Proc. 31st Int. Conf. on Neural Information Processing Systems (NeurIPS)* 8617 8629, 2018 , 第 – 页。 79. M. Ostrovsky, C. Barrett, and G. Katz. 一种用于验证卷积神经网络的抽象-细化方法. 收录于 *Proc. 20th. Int. Symposium on Automated Technology for Verification and Analysis (ATVA)* 391 396, 2022 , 第 – 页。 80. Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lak- shminarayanan, and J. Snoek. 你能信任模型的不确定性吗？评估数据集偏移下的预测不确定性. 收录于 *Proc. 33rd Conf. on Neural Information Processing Systems (NeurIPS)* 14003 14014, 2019 , 第 – 页。 81. C. Packer, K. Gao, J. Kos, P. Kr¨ahenb¨uhl, V. Koltun, and D. Song. 评估深度强化学习中的泛化，2018年。技术报告。https: //arxiv.org/abs/1810.12282。 82. E. Polgreen, R. Abboud, and D. Kroening. 反例引导的神经网络综合，2020年。技术报告。https://arxiv.org/abs/2001.09245。 83. P. Prabhakar. 神经网络约简的双模拟. 收录于 *Proc. 23rd Int.Conf. Verification on Model Checking, and Abstract Interpretation (VMCAI)*，第285–300页，2022年。

84. P. Prabhakar 和 Z. Afzal。基于抽象的神经网络输出范围分析，2020年。技术报告。https://arxiv.org/abs/2007.09527。85. A. Raffin、A. Hill、A. Gleave、A. Kanervisto、M. Ernestus 和 N. Dormann。Stable-Baselines
3 *Journal of*：可靠的
Machine Learning Research (JMLR) 22 1 8, 2021
强化学习实现。，：–。86. M. Riedmiller。神经拟合Q迭代——基于数据的高效神经强化学习方法的初步经验。载于
Proc. 16th European Conf. on，第–页。
Machine Learning (ECML) 317 328, 2005
87. T. Rockafellar。拉格朗日乘子与最优性。 *SIAM Review*，35(2)：183–238，1993年。
88. N. Rotman、M. Schapira 和 A. Tamar。深度强化学习的在线安全保障。载于
Proc. 19th ACM Workshop on Hot Topics in Networks，第–页。
(HotNets) 88 95, 2020
89. W. Ruan、X. Huang 和 M. Kwiatkowska。具有可证明保证的深度神经网络可达性分析。载于 *Proc. 27th Int. Joint Conf. on Artificial*，。
Intelligence (IJCAI) 2018
90. S. Ruder。梯度下降优化算法概述，2016年。技术报告。
https://arxiv.org/abs/1609.04747。91. J. Schulman、F. Wolski、P. Dhariwal、A. Radford 和 O. Klimov。近端策略优化算法，2017。技术报告。http://arxiv.org/abs/1707.06347。92. S. Seshia、A. Desai、T. Dreossi、D. Fremont、S. Ghosh、E. Kim、S. Shivakumar、M. Vazquez-Chanlatte 和 X. Yue。深度神经网络的形式化规范。载于
Proc. 16th Int. Symposium on Automated Technology for Verification，第–页。
and Analysis (ATVA) 20 34, 2018
93. D. Silver、A. Huang、C. Maddison、A. Guez、L. Sifre、G. van Den Driessche、J. Schrittwieser、I. Antonoglou、V. Panneershelvam、M. Lanctot 和 S. Dieleman。掌握围棋：利用深度神经网络和树搜索。 *Nature*，529 7587 484 489，2016（）：–。94. K. Simonyan 和 A. Zisserman。用于大规模图像识别的超深卷积网络，2014年。技术报告。http://arxiv.org/abs/1409. 1556。95. G. Singh、T. Gehr、M. Puschel 和 M. Vechev。用于认证神经网络的抽象域。载于
Proc. 46th ACM SIGPLAN Symposium on Principles，。96.
of Programming Languages (POPL) 2019
M. Sotoudeh 和 A. Thakur。使用小型泛化补丁校正深度神经网络。载于
Workshop on Safety and Robustness in Decision Making，2019年。97. C. Strong、H. Wu、A. Zelji´c、K. Julian、G. Katz、C. Barrett 和 M. Kochender-fer。由 ReLU 网络表示的目标函数的全局优化。*Journal of Machine Learning*，第1–28页，2021。98. X. Sun、H. Khedr 和 Y. Shoukry。神经网络控制自主系统的形式化验证。载于
Proc. 22nd ACM Int. Conf. on Hybrid Systems:，。99.
Computation and Control (HSCC) 2019
R. Sutton 和 A. Barto。 *Reinforcement Learning: An Introduction*。麻省理工学院出版社，2018年。100. R. Sutton、D. McAllester、S. Singh 和 Y. Mansour。基于函数逼近的强化学习策略梯度方法。载于
Proc. 12th Conf.，。
on Neural Information Processing Systems (NeurIPS) 1999

101. V. Tjeng、K. Xiao 和 R. Tedrake。使用混合整数规划评估神经网络的鲁棒性，2017。技术报告。http://arxiv.org/ abs/1711.07356。102. L. Tolstoy。 *Anna Karenina*。《俄罗斯信使》，1877年。103. C. Urban、M. Christakis、V. W¨ustholz 和 F. Zhang。神经网络的完全并行公平性认证。载于 *Proc. ACM Int. Conf. on Object Oriented* ，第 – 页。*Programming Systems Languages and Applications (OOPSLA)* 1 30, 2020
104. M. Usman、D. Gopinath、Y. Sun、Y. Noller 和 C. P˘as˘areanu。NNrepair：基于约束的神经网络分类器修复，2021。技术报告。http://arxiv.org/abs/2103.12535。105. A. Valadarsky、M. Schapira、D. Shahaf 和 A. Tamar。学习使用深度强化学习进行路由。载于 *NeurIPS Deep Reinforcement Learning Symposium*，2017。106. H. van Hasselt、A. Guez 和 D. Silver。结合双重Q学习的深度强化学习。载于 *Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI)*，2016。107. M. Vasi´c、A. Petrovi´c、K. Wang、M. Nikoli´c、R. Singh 和 S. Khurshid。Mo¨ET：专家树混合及其在可验证强化学习中的应用。*Neural Networks*，151：34–47，2022。108. S. Wang、K. Pei、J. Whitehouse、J. Yang 和 S. Jana。使用符号区间对神经网络进行形式化安全分析。载于 *Proc. 27th USENIX Security* ，第 – 页。*Symposium* 1599 1614, 2018
109. H. Wu、A. Ozdemir、A. Zelji´c、A. Irfan、K. Julian、D. Gopinath、S. Fouladi、G. Katz、C. P˘as˘areanu 和 C. Barrett。验证神经网络的并行化技术。载于 *Proc. 20th Int. Conf. on Formal Methods in Computer-Aided* ，第 – 页。*Design (FMCAD)* 128 137, 2020
110. H. Wu、A. Zelji´c、K. Katz 和 C. Barrett。使用不可行性总和进行高效的神经网络分析。载于 *Proc. 28th Int. Conf. on Tools and Algorithms for* ，第 – *the Construction and Analysis of Systems (TACAS)* 143 163, 2022 页。111. W. Xiang 、 H. Tran 和 T. Johnson。多层神经网络的输出可达集估计与验证。*IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 2018 ，。112. J. Yang、X. Zeng、S. Zhong 和 S. Wu。用于提升泛化性能的有效神经网络集成方法。*IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 24 6 878 887, 2013 ，（）：–。
113. X. Yang、T. Yamaguchi、H. Tran、B. Hoxha、T. Johnson 和 D. Prokhorov。基于可达性分析的神经网络修复。载于 *Proc. 20th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS)* 221 236, ，第 – 页。2022
114. T. Zelazny、H. Wu、C. Barrett 和 G. Katz。关于减少神经网络验证中的过近似误差。载于 *Proc. 22nd Int. Conf. on Formal* ，第 – *Methods in Computer-Aided Design (FMCAD)* 17 26, 2022 页。115. H. Zhang 、 M. Shinn 、 A. Gupta 、 A. Gurfinkel 、 N. Le 和 N. Narodytska。通过可达性分析验证认知任务中的循环神经网络。载于 *Proc. 24th European Conf. on Artificial Intelligence (ECAI)*，第1690 1697页，2020 –。
116. J. Zhang、J. Kim、B. O'Donoghue 和 S. Boyd。使用 REINFORCE 的样本高效强化学习，2020。技术报告。https://arxiv.org/abs/ 2010.11364。117. J. Zhang、Y. Liu、K. Zhou、G. Li、Z. Xiao、B. Cheng、J. Xing、Y. Wang、T. Cheng、L. Liu 等。使用深度强化学习的端到端自动云数据库调优系统。载于 *Proc. of the 2019 Int. Conf. on Management* ，第 – 页。*of Data (SIGMOD)* 415 432, 2019

附录

A 训练与评估

在本附录中，我们详细说明了超参数和训练过程，以便复现全部三个基准测试中的所有模型和环境。我们还全面概述了各种实现细节。代码基于 *Stable-Baselines 3* [85] 和 *OpenAI Gym* [19] 软件包。除非另有说明，训练和评估期间使用的各项参数均为默认值（按训练算法、环境等区分）。我们的原始代码和模型已在网上公开 [7]。

A.1 训练算法

我们使用 *Actor-Critic* 算法训练模型。这些是最先进的强化学习训练算法，通过迭代优化两个神经网络：

– 一个评论家网络，用于学习价值函数 [73]（也称为*Q*-函数），为每个 ⟨状态、动作⟩ 对赋予一个值；以及– 一个演员网络，也就是由该算法训练的基于深度强化学习的智能体。该网络通过迭代最大化评论家学得的价值函数，从而改进学得的策略。

具体而言，我们使用了两种演员-评论家算法的实现：近端策略优化 (PPO) [91] 和软演员-评论家 (SAC) [44]。

演员-评论家算法被认为具有很大优势，因为它们通常只需相对较少的样本即可学习，而且能够让智能体学习适用于连续状态、动作⟩配对空间的策略。

在每个训练过程中，所有模型均使用相同的超参数进行训练，唯一的例外是伪随机数生成器（*PRNG*）的种子。每个训练阶段包含 10 个检查点，每个检查点都包含固定数量的环境步数，如下文所述。评估模型时，我们使用每个训练过程（针对每个基准测试）的最后一个检查点。

A.2 架构

在所有基准测试中，我们都使用了采用前馈架构的深度神经网络。每项基准测试所选架构的摘要请参见表 1。

基准测试	隐藏层	层大小	激活函数	训练算法
Cartpole	2	[32, 16]	ReLU	PPO
Mountain Car	2	[64, 16]	ReLU	SAC
Aurora	2	[32, 16]	ReLU	PPO

表 1：各基准测试对应的深度神经网络架构和训练算法。

A.3 Cartpole 参数

架构与训练

1. **架构**
   – 隐藏层:2– 隐藏层大小: 分别为 32 和 16– 激活函数:ReLU

2. **训练**
   – 算法：近端策略优化（PPO）– 伽马（γ）：0.95– 批量大小：128– 检查点数量：10– 总时间步数 （每个检查点的训练步数）：50，000– 伪随机数生成器种子 （每个种子用于训练一个不同的模型）：$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$

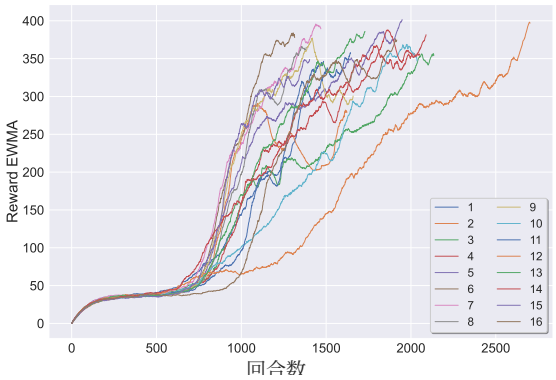


图 5： *Cartpole*：模型的奖励指数加权移动平均（*EWMA*）。所有模型在训练结束时都获得了较高的奖励。

环境
我们使用了可配置的$CartPoleContinuous\text{-}v0$ 环境。给定x轴位置的下界和上界，分别记为 $[low, high]$和 $mid = \frac{high+low}{2}$，初始x位置将从区间 $[mid - 0.05, mid + 0.05]$中随机均匀地抽取。

一个回合 是智能体与环境交互的一系列过程，当达到终止状态时结束。在Cartpole环境中，回合会在以下情况中最先发生的一种出现后终止：

1. 小车的位置超出平台边界（以x轴位置表示）；或 2. 小车无法平衡杆，导致杆倒下（以θ值表示）；或 3. 已经过了 500 个时间步

领域

1. **（训练）分布内**
   - 动作最小幅度：True
   - x轴下界 （x阈值下限）：-2.4
   - x轴上界 （x阈值上限）：2。-4
   - -
   - -
2. **（分布外）输入域**
   我们评估了两种对称的分布外场景：小车的 x 位置表示沿单一方向显著延伸的平台，因此涵盖了训练期间未曾见过的区域。具体而言，我们生成了一个输入点定义域，其特征是 x轴边界以相同概率从 $[-10, -2.4]$ 或 $[2.4, 10]$ 中选取（而非采用 $[-2.4, 2.4]$的分布内范围）。小车的初始位置从该范围的中心 ± 0.05中均匀抽取：分别为 $[-6.4 - 0.05, -6.4 + 0.05]$ 和 $[6.4 - 0.05, 6.4 + 0.05]$。

   所有其他参数均与分布内使用的参数相同。

   分布外场景1
   - x轴下界 （x阈值下限）：-10.0
   - x轴上界 （x阈值上限）：-2.4
   - -
   - -

   分布外场景2
   - x 轴下界 （x阈值下限）：2.4
   - x 轴上界 （x阈值上限）：10.0
   - -
   - -

A.4 Mountain Car 参数

架构与训练

1. **架构**

   – 隐藏层：2– 隐藏层大小：分别为 64 和 16–
   激活函数：ReLU– 均值裁剪参数：5.0– 对数
   标准差初始化参数：-3.6

2. **训练**

   – 算法： 软演员-评论家（SAC）– 伽马系数（γ）： 0.9999– 批量大小：
   512– 缓冲区大小： 50,000– 梯度步数：32– 学习率： 3×10^{-4}– 学习
   开始步数：0– *tau*（τ）： 0.01– 训练频率：32– 使用状态依赖探索：是–
   检查点数量：10– 总时间步数 （每个检查点的训练步数）： 5,000– 伪
   随机数生成器种子 （每个种子用于训练不同的模型）：
   $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$

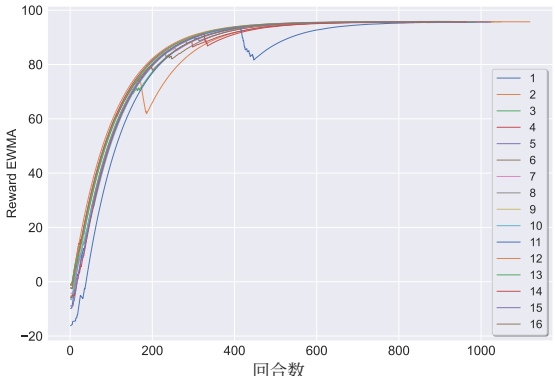


图 6： *Mountain Car*：各模型的奖励指数加权移动平均（*EWMA*）。所有模型（在训练结束时）都获得了较高的奖励。

环境
我们使用了 *MountainCarContinuous-v1* 环境。

领域

1. **（训练）分布内 –** 最小位置: -1.2– 最大位置: -0.6– 目标位置: 0.45– 最小动作 （如果智能体的动作小于 0 且低于此值，则使用此值）： -2– 最大动作 （如果智能体的动作为正且高于此值，则使用此值）： 2– 最大速度: 0.4– 初始位置范围 （初始位置在此范围内均匀抽取）：$[-0.9, -0.6]$– 初始速度范围 （初始速度在此范围内均匀抽取）：$[0, 0]$ （即此情景中的初始速度始终为 0）– *x* 轴缩放因子 （用于缩放 x 轴）： 1.5

2. **分布外输入域**
   输入与分布内使用的输入相同，但以下情况除外:

   – 最小位置: -2.4– 最大位置:1.2– 目标位置:0.9– 初始位置范围: $[0.4, 0.5]$– 初始位置速度 : $[-0.4, -0.3]$

A.5 Aurora 参数

架构与训练

1. **架构**
   – 隐藏层：2– 隐藏层大小：分别为 32 和 16– 激活函数：ReLU

2. **训练**
   – 算法：近端策略优化（PPO）– *gamma (γ)*：0.99– 每次更新时每个环境运行的步数 (n steps)：8,192– 优化代理损失时的轮数 (n epochs)：4– 学习率： 1×10^{-3}

– 价值函数损失系数 (vf coef)：1– 熵函数损失系数 (ent coef)： 1×10^{-2}– 检查点数量：6– 总时间步数 (每个检查点对应的训练步数)：656，000（与原论文 [50] 中使用的一致）– 伪随机数生成器种子 (每个种子用于训练一个不同的模型)：
$\{4, 52, 105, 666, 850, 854, 857, 858, 885, 897, 901, 906, 907, 929, 944, 945\}$为简化起见，我们将这些种子映射到索引 $\{1 \ldots 16\}$，依此类推（例如，$\{4\} \to \{1\}$、$\{52\} \to \{2\}$ 等）。

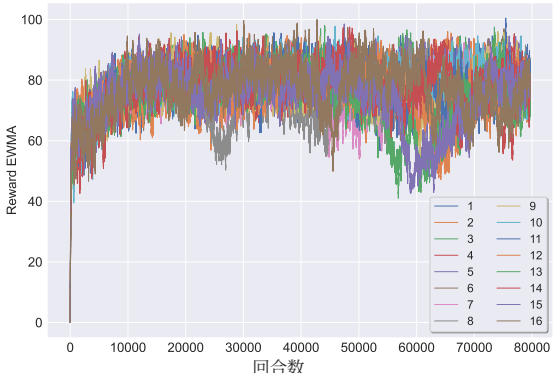


图7：*Aurora*（短期训练）：训练过程中各模型奖励的指数加权移动平均（*EWMA*）。所有模型都获得了较高的奖励（在训练结束时）。

环境

我们使用了可配置版本的*PccNs-v0* 环境。对于实验1中的模型（采用短期训练），每个回合包含50步。对于实验3中的模型（采用长期训练），每个回合包含400步。

领域

1. **（训练）分布内**
   – 最小初始发送速率比（相对于链路带宽）（最小初始发送速率带宽比）：0.3– 最大初始发送速率比（相对于链路带宽）（最大初始发送速率带宽比）：1.5

2. **分布外输入域**
   为了界定输入中的延迟梯度和延迟比元素，我们采用了浅缓冲区配置，并设置界定参数 $\delta > 0$，使得延迟梯度 $\in [-\delta, \delta]$ 且延迟比 $\in [1.0, 1.0 + \delta]$。

– 最小初始发送速率比（相对于链路带宽）（最小初始发送速率带宽比）：2.0– 最大初始发送速率比（相对于链路带宽）（最大初始发送速率带宽比）：8.0– 使用浅缓冲区：是– 浅缓冲区 δ 界限参数：1×10^{-2}

B 验证查询

B.1 深度神经网络验证查询：玩具示例

让我们重新考察图 1 中的深度神经网络。假设我们希望验证：对于所有非负输入，该深度神经网络的输出都严格小于 25；也就是说，对于所有输入 $x = \langle v_1^1, v_1^2 \rangle \in \mathbb{R}^2_{\geq 0}$，都有 $N(x) = v_4^1 < 25$。我们可以通过选择一个将输入限制为非负数的前置条件，即 $P = (v_1^1 \geq 0 \wedge v_1^2 \geq 0)$，并设置 $Q = (v_4^1 \geq 25)$ 来将其编码为验证查询；后者是对我们所期望性质的否定 。对于这个特定的验证查询，可靠验证器将返回可满足，并给出一个可行反例，例如 $x = \langle 0, 4 \rangle$，它会产生 $v_4^1 = 28 \geq 25$。因此，图 1 所示的深度神经网络不满足这一性质。所有查询都已发送至*Marabou* [54]——一个可靠且完备的验证引擎，曾用于其他深度神经网络验证相关工作 [5, 6, 9, 10, 17, 21, 23, 26, 32, 33, 49, 79, 95, 108, 110]。

B.2 输入域查询

接下来，我们将详细说明如何对查询进行编码，随后将其输入后端验证引擎，以计算一对深度神经网络的 PDT 分数。给定一对深度神经网络 N_1 和 N_2，我们执行以下阶段：

1. 我们将 N_1 和 N_2 拼接成一个新的深度神经网络 $N_3 = [N_1; N_2]$，其规模大约是每个原始深度神经网络的两倍（因为 N_1 和 N_2 具有相同的架构）。N_3 的输入与每个单独深度神经网络的原始输入大小相同，并连接到每个深度神经网络的第二层，从而使相同的输入能够在整个网络中流向每个原始深度神经网络的输出层（N_1，N_2）。因此，N_3 的输出层是 N_1 和 N_2输出的拼接。图 8 展示了拼接的深度神经网络的构建示意图。

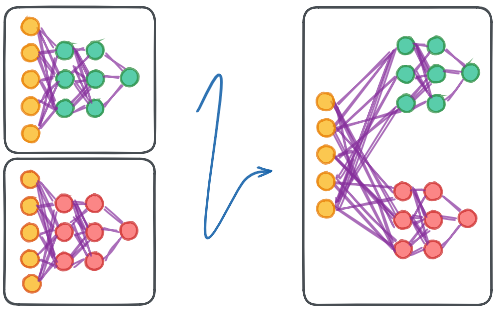


图 8：为了计算 *PDT* 分数，我们构建了一个新的深度神经网络，它由每对深度神经网络拼接而成，并共享相同的输入。

2. 对表示输入变量取值范围的前置条件 P进行编码。我们使用这些输入，将相关值纳入所关注的输入域中。在某些情况下，这些值是预先定义的，以匹配所评估的分布外设置；在其他情况下，我们根据模型训练后的经验模拟提取这些值。正如我们之前所提到的，边界由系统设计者根据对输入域的先验知识提供。在我们的实验中，我们使用了以下输入域边界：

(a) **Cartpole：**
– x 位置： $x \in [-10, -2.4]$ 或 $x \in [2.4, 10]$ 这两个场景的 PDT 均设为各自的最大 PDT 分数。– x 速度： $v_x \in [-2.18, 2.66]$– 角度： $\theta \in [-0.23, 0.23]$– 角速度： $v_\theta \in [-1.3, 1.22]$

(b) **Mountain Car：**
– x 位置： $x \in [-2.4, 0.9]$– x 速度： $v_x \in [-0.4, 0.134]$

(c) **Aurora：**
– 延迟梯度： $x_t \in [-0.007, 0.007]$，对于所有满足 $(t \bmod 3) = 0$ 的 t – 延迟比： $x_t \in [1, 1.04]$，对于所有满足 $(t \bmod 3) = 1$ 的 t – 发送比： $x_t \in [0.7, 8]$，对于所有满足 $(t \bmod 3) = 2$ 的 t

3. 编码一个后置条件 Q，它编码了（针对固定的松弛量 α 和给定的距离函数 d ）：$\mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}^+$，即对于输入 $x' \in \Psi$，以下条件成立： $d(N_1(x'), N_2(x')) \geq \alpha$

距离函数示例：
(a) L_1**范数：**

$$d(N_1, N_2) = \arg\max_{x \in \Psi}(|N_1(x) - N_2(x)|)$$

此距离函数用于 Aurora 基准测试。
(b) 条件 – 距离(“c –距离”)： </style}该函数返回两个深度神经网络在所有满足约束 c 的输入 $x \in \Psi$ 上的最大值 L_1 范数，其中两个输出 $N_1(x)$、 $N_2(x)$ 均符合该约束。

$$c - distance(N_1, N_2) \triangleq \max_{x \in \Psi \text{ s.t. } N_1(x), N_2(x) \vDash c}(|N_1(x) - N_2(x)|)$$

对于 Cartpole 和 MountainCar 基准测试，我们将距离函数定义为：

$$d(N_1, N_2) = \min_{c, c'}(c - distance(N_1, N_2), c' - distance(N_1, N_2))$$

我们选择 $c := N_1(x) \geq 0 \wedge N_2(x) \geq 0$ 和 $c' := N_1(x) \leq 0 \wedge N_2(x) \leq 0$ 这个距离函数旨在找出给定输入类别（在我们的案例中为非正或非负）下，两个模型输出（动作）之间的最大差异。其背后的直觉是，在某些基准测试中，优劣模型的动作可能在符号上有所不同（而不仅仅是幅度不同）。例如，考虑

Cartpole 基准测试中的一种情形：小车位于平台的“边缘”，朝一个方向（离开平台）采取动作会导致回合结束，而朝另一个方向采取动作则能让智能体继续回合并增加奖励。

C 算法变体与超参数

在本附录中，我们详细说明了整个评估过程中使用的算法的额外超参数和筛选标准。正如结果所示，我们的方法在各种设置下都具有很强的稳健性。

C.1 精度

对于每个基准测试和每项实验，我们任意选择了 $k = 16$ 个达到分布内数据奖励阈值的模型。随后，我们使用这些模型进行实证评估。PDT 分数的计算采用有限精度 $0.5 \leq \epsilon \leq 1$，具体取值取决于基准测试（Cartpole 和 Aurora 为 1，Mountain Car 为 0.5）。

C.2 筛选标准

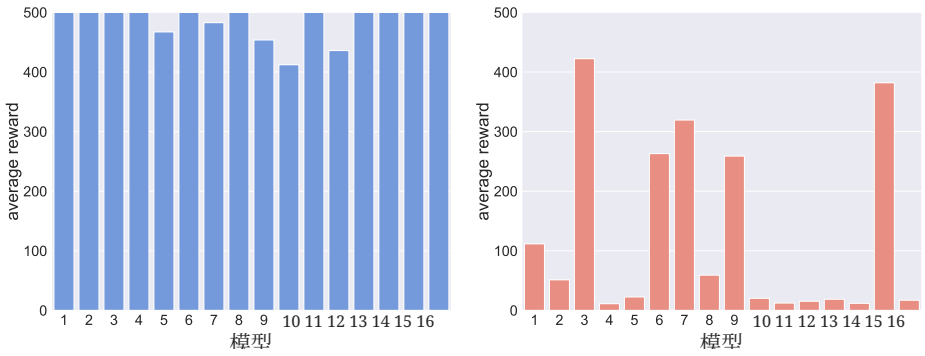
如第 3 节所述，我们的算法会迭代地筛选掉（算法 2 的第 8 行）不一致分数相对较高的模型，即可能与输入域中的其他模型存在分歧的模型。在根据其 DS 分数对模型进行排序后，我们提出了三种不同的标准，用于选择在给定迭代中要移除的模型：

1. 百分位数：移除不一致分数最高的 p%模型，其中 p为预先设定的值。在实验中，我们选择了$p = 25$%。2. 最大值：具体步骤如下：(a) 按降序排列所有模型的 DS 分数；(b) 计算每对相邻分数之间的差值；(c) 找出任意两个连续 DS 分数之间的最大差值；(d) 将该差值对应的较大 DS 作为阈值；(e) 移除 DS 大于或等于此阈值的所有模型。3. 组合：在每次迭代中，根据最大值或百分位数哪种标准能剔除更多模型，按照相应标准移除模型。

D Cartpole：补充结果

在对此基准测试进行评估的整个过程中，我们使用**250** 作为阈值，以区分优质模型和劣质模型——该阈值与表现不佳模型的奖励值（通常低于 100）之间存在较大差距。

请注意，如图 10 所示，我们的算法最终也会移除一些表现较好的模型。不过，最终结果仅包含表现良好的模型，与其他基准测试中的情况一样。



(a) *In-distribution* (b) *OOD*

图 9：*Cartpole*：模型在不同分布中的平均奖励。

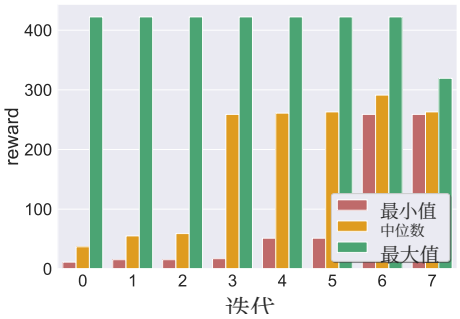
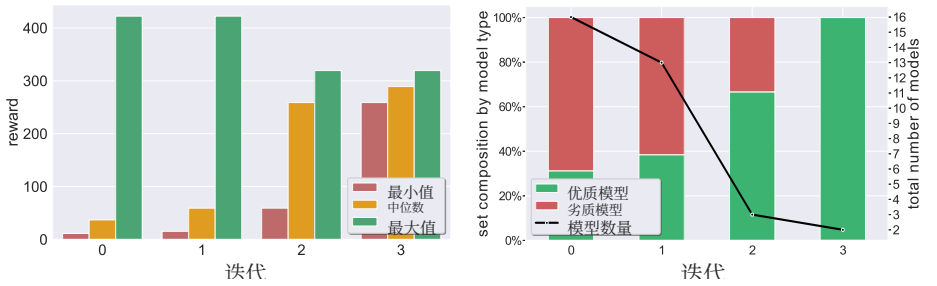


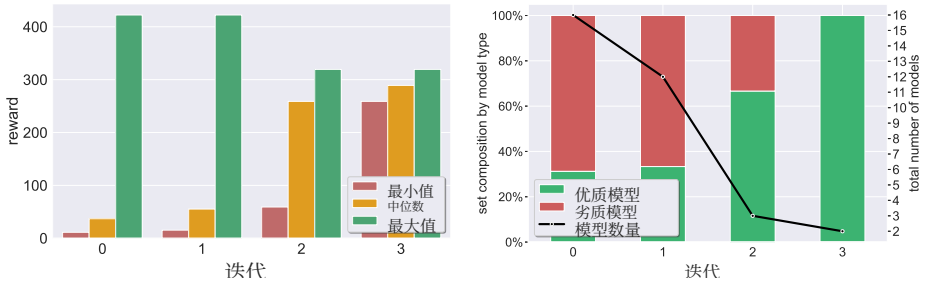
图 10：*Cartpole*：模型选择结果：每次迭代后选出的模型的最小值、中位数和最大值奖励。

D.1 各筛选标准的结果



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 1 1: *Cartpole*: 使用 `MAX` 筛选标准的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

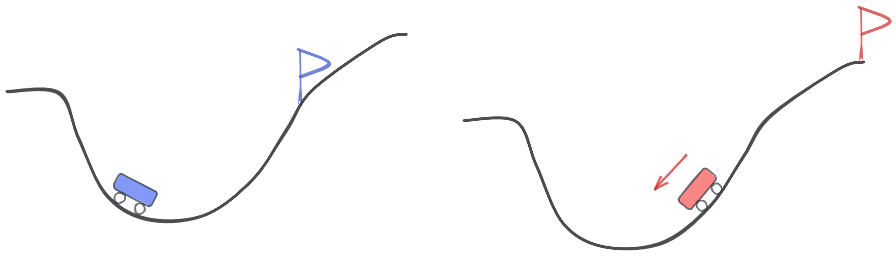
图 1 2: *Cartpole*: 使用 `COMBINED` 筛选标准的结果。

E MountainCar：补充结果

E.1 Mountain Car 基准测试

Mountain Car 是一个经典且相对简单的强化学习基准测试。在此基准测试中，一辆汽车（智能体）被放置在两座山丘之间的山谷中（位于 $x \in [-1.2, 0.6]$），需要抵达其中一座山丘顶上的旗帜。状态 $s = (x,\ v_x)$ 表示汽车的位置（沿 x 轴）和速度。智能体的动作（输出）是施加的力：一个连续值，表示智能体希望移动的幅度和方向。训练期间，智能体会受到激励去抵达旗帜（旗帜最初放置在山谷顶端，位于 $x = 0.45$）。在抵达旗帜之前的每个时间步，智能体都会获得少量负奖励；若抵达旗帜，则会获得大量正奖励。抵达旗帜或步数超过预先设定的值（我们实验中为 300）时，回合即告结束。好模型和劣质模型通过平均奖励阈值 90 加以区分。

训练期间（分布内），汽车最初被放置在山谷底部的左 侧，速度较低且随机（见图 13a）。我们训练了$k = 16$ 个智能体（记为 $\{1, 2, \ldots 16\}$），它们在分布内均表现良好（即平均奖励高于我们的阈值）。此项评估进行了超过 10，000 个回合。



(a) *Training (in-distribution) setting: the agent's initial point is on the left side, and the goal is at a nearby point*

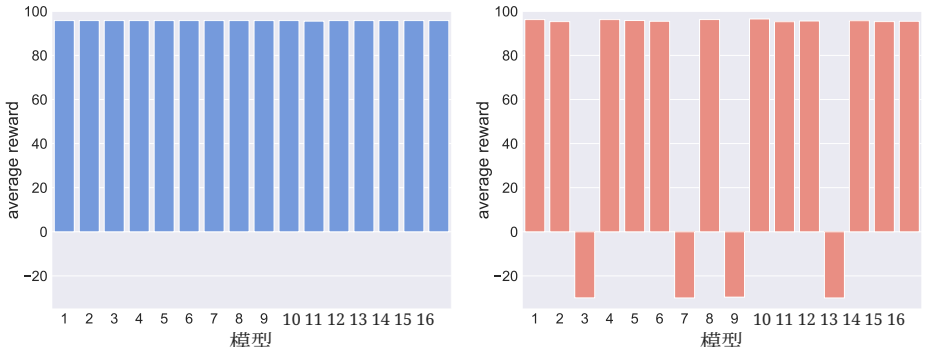
(b) *OOD setting: the agent's initial point is farther away, with initial negative velocity; the goal is significantly farther up than in training*

图 13：*Mountain Car*：图 *(a)* 展示了智能体接受训练的设置，图 *(b)* 展示了更难的分布外设置。

根据训练环境所用的场景，我们将分布外输入域指定为：(i) 将 x 轴从 $[-1.2, 0.6]$ 延伸至 $[-2.4, 0.9]$；(ii) 将旗帜进一步向右移动，从 $x = 0.45$ 移至 $x = 0.9$；以及 (iii) 将汽车的初始位置设在山谷底部更靠右处，并赋予较大的初始负 速度（向左）。图 13b 展示了示意图。这些新设置代表一种新颖的状态分布，导致

智能体需要对训练期间未曾观察到的状态作出响应：不同的位置、更大的速度，以及位置与速度方向的不同组合。

在分布内表现良好的模型中，有 4 个模型在分布外场景中失败了（即未能到达旗帜，并以负的平均奖励结束回合），而其余 12 个模型则成功了，也就是说，它们在分布外数据上进行模拟时获得了较高的平均奖励（见图 14）。成功模型的比例很高并不令人意外，因为 Mountain Car 是一个相对简单的基准测试。

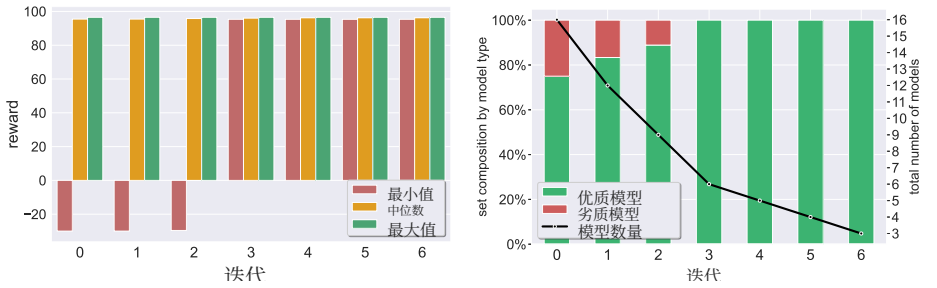


(a) *In-distribution* (b) *OOD*

图 14：*Mountain Car*：模型在不同分布中的平均奖励。

为了评估我们的算法，我们在这些模型和上述分布外输入域上运行了该算法，并检查它是否剔除了那些（尽管在分布内表现成功）却无法适应新的、更困难场景的模型。确实，我们的方法成功筛除了所有未成功的模型，仅留下 5 个模型的子集（$\{2, 4, 8, 10, 15\}$），它们在分布外场景中均表现良好。

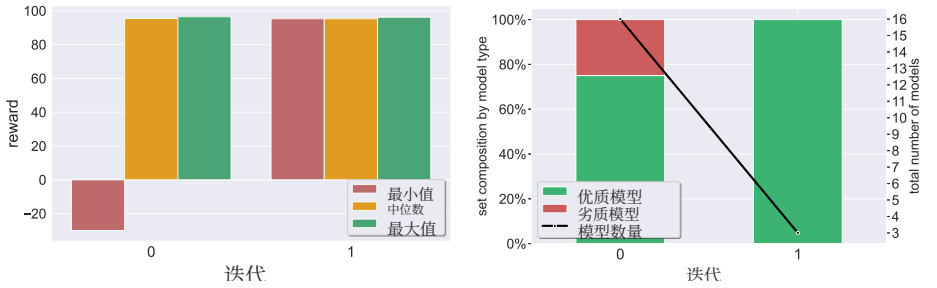
我们还注意到，我们的算法对各种超参数都具有鲁棒性，如图 15、图 16 和图 17 所示。这些图展示了在采用各种筛选标准（详见附录 C）时，我们的算法每次迭代的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

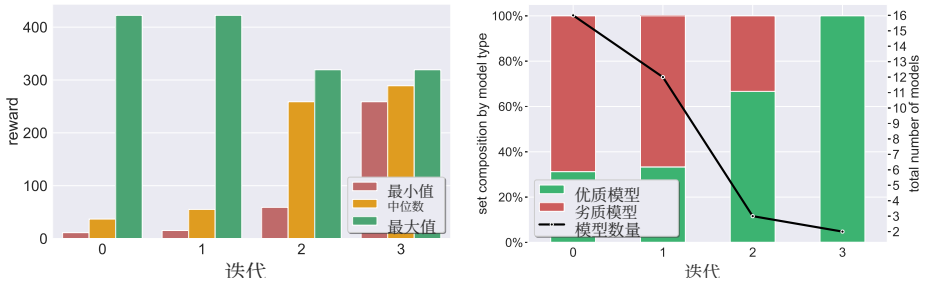
图 15： *Mountain Car*：使用 `PERCENTILE` 筛选标准的结果。

E.2 其他筛选标准



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 16： *Mountain Car*：使用 `MAX` 筛选标准的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 17： *Mountain Car*：使用 `COMBINED` 筛选标准的结果。

E.3 组合实验

由于初始候选集合存在固有偏差——原始的 16 个模型中有 12 个在分布外设置下表现优异——我们着手验证：算法成功返回的全是优质模型，确实是因为算法正确，而非模型集合本身倾向于包含优质模型。在我们的实验中（总结如下），我们人为生成了新的模型集合，其中优质模型的比例特意设得低于原始集合。随后，我们对初始子集的所有可能组合重新运行算法，并计算在这种新设置下，对于每个子集，从经过筛选过程后存活的模型中选出优质模型的概率。正如我们所展示的，我们的方法显著提高了选出优质模型的机会即使优质模型在原始集合中占少数时也是如此。例如，图 18 最左侧一列显示，在包含 4 个劣质模型、仅有 2 个优质模型的集合中，运行算法后选出优质模型的概率超过 60%（！）——几乎是运行算法前从原始集合中随机选出优质模型概率的两倍。这些结果在不同子集大小以及各种筛选标准下均保持一致。我们指出，在计算选出优质模型的概率时，我们假定从模型子集中随机选择：在 应用算法之前，子集就是原始模型集合；而在应用算法之后 ，子集则根据筛选过程的结果更新。该概率根据筛选过程后仍存活的劣质模型组合数量，以及这些情况下返回的所有模型中劣质模型所占的比例计算得出（假定每个子集被选中的概率均匀）。

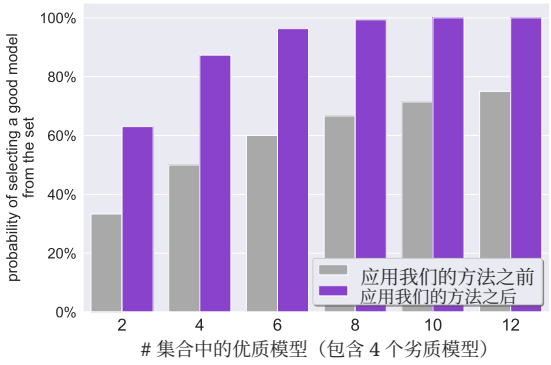


图 18： *Mountain Car*：由于进行了有效筛选，我们的算法有效提高了选择优质模型的概率。该图对应表 *3*。

组成		总数 # 实验	# 次实验，使用		选择的概率 良好的 模型	
good	bad	子群 组合 的良好模型	**all** 存活 模型良好	**some** 存活 模型 劣质	朴素	我们的方法
12	4	$\binom{12}{12} = 1$	1	0	75 %	100%
10	4	$\binom{12}{10} = 66$	65	1	71.43 %	99.49 %
8	4	$\binom{12}{8} = 495$	423	72	66.67 %	95.15 %
6	4	$\binom{12}{6} = 924$	549	375	60 %	86.26 %
4	4	$\binom{12}{4} = 495$	123	372	50 %	74.01 %
2	4	$\binom{12}{2} = 66$	66	0	33.33 %	54.04 %

表2： 使用 *PERCENTILE* 筛选标准的组合山地车实验结果汇总。

组成		总数 # 实验	# 项实验		选择的概率 良好的 模型	
good	bad	子群 组合 的良好模型	**all** 存活 模型良好	**some** 存活 模型 劣质	朴素	我们的方法
12	4	$\binom{12}{12} = 1$	1	0	75 %	100%
10	4	$\binom{12}{10} = 66$	66	0	71.43 %	100 %
8	4	$\binom{12}{8} = 495$	486	9	66.67 %	99.34 %
6	4	$\binom{12}{6} = 924$	844	80	60 %	96.33 %
4	4	$\binom{12}{4} = 495$	375	120	50 %	87.32 %
2	4	$\binom{12}{2} = 66$	31	35	33.33 %	63.03 %

表 3： 使用 *MAX* 筛选标准的组合山地车实验结果汇总。

组成		总数 # 实验	# 次实验，使用		选择的概率 良好的 模型	
good	bad	子群 组合 的 良好模型	**all** 存活 模型 良好	**some** 存活 模型 劣质	朴素	我们的方法
12	4	$\binom{12}{12} = 1$	1	0	75 %	100%
10	4	$\binom{12}{10} = 66$	66	0	71.43 %	100 %
8	4	$\binom{12}{8} = 495$	481	12	66.67 %	99.26 %
6	4	$\binom{12}{6} = 924$	842	82	60 %	96.74 %
4	4	$\binom{12}{4} = 495$	372	123	50 %	88.09 %
2	4	$\binom{12}{2} = 66$	31	35	33.33 %	63.36 %

表4： 组合山地车实验的结果汇总，使用 *COMBINED* 筛选标准。

F Aurora：补充结果

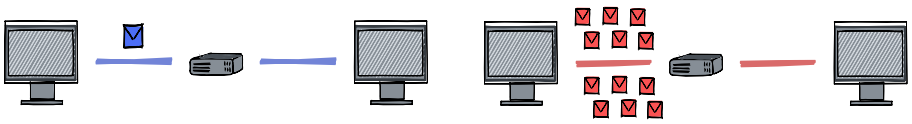
F.1 附加信息

1. 对 Aurora 输入统计信息的更详细说明：(i) 延迟梯度：最近监测间隔（MI，即“monitor interval”）内延迟（数据包延迟）的导数；(ii) 延迟比：当前监测间隔内的平均延迟与此前观测到的最小时延之比；以及 (iii) 发送比：最近监测间隔内发送的数据包数量与已确认的数据包数量之比。如前所述，这些指标反映了链路的拥塞程度。

2. 对于我们在此基准测试上的所有实验，我们将达到平均奖励高于阈值 **99** 的模型定义为“优质”模型；“劣质”模型则是指奖励低于该阈值的模型。

3. 分布内的平均奖励不一定与分布外的平均奖励相关。例如，在训练期间采用短回合的实验1中（见图20）：

   (a) 在分布内，模型 $\{4\}$ 的奖励低于模型 $\{2\}$ 和$\{5\}$，但在分布外的奖励更高。(b) 在分布内，模型 $\{16\}$ 的奖励低于模型 $\{15\}$，但在分布外的奖励更高。

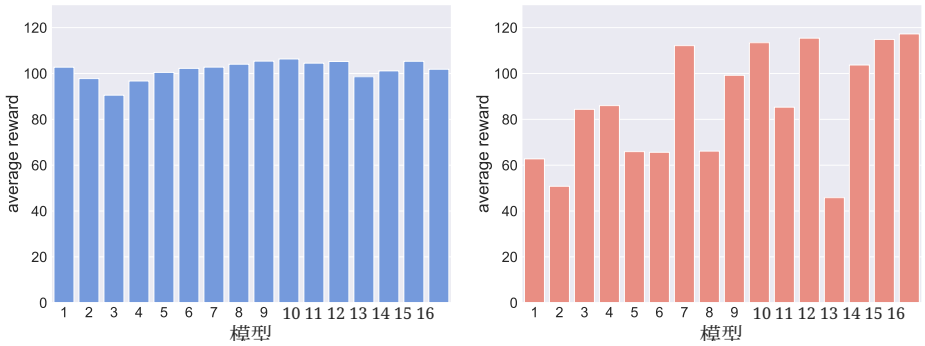


(a) *In-distribution setting, in which the agent was trained, with low packet congestion*

(b) *OOD settings, of a much higher congestion rate, with a significantly greater packet loss*

图 19： *Aurora*：分布内与分布外设置示意图。

实验(1)：Aurora：短训练回合



(a) *In-distribution* (b) *OOD*

图 20：*Aurora* 实验 *1*：模型在不同分布上进行模拟时的平均奖励。

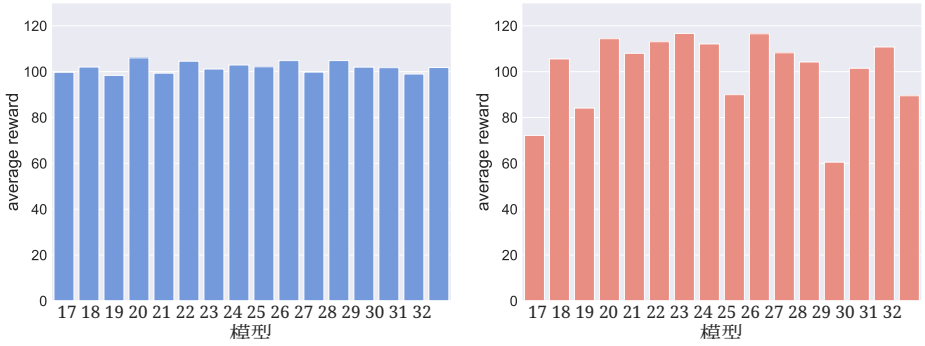
实验（3）：Aurora：长训练回合

imile 16 个智能体。在该实验中，我们将每个训练回合增加到 400 步（而“短”训练中为 50 步）。其余参数与实验 1 中之前的设置完全相同。这一次，有 5 个模型在分布外环境中表现不佳（即未达到我们的 99 奖励阈值），而剩余的 11 个模型在分布内和分布外环境中均表现良好。

使用 MAX 准则运行我们的方法时，算法返回了 4 个模型，它们都是成功泛化的 11 个模型中的子集，并且筛选过程完全剔除了所有未成功的模型。使用 PERCENTILE 或 COMBINED 准则运行算法，也得到了这一组模型的子集，这表明筛选过程再次取得了成功（且对各种算法超参数具有鲁棒性）。

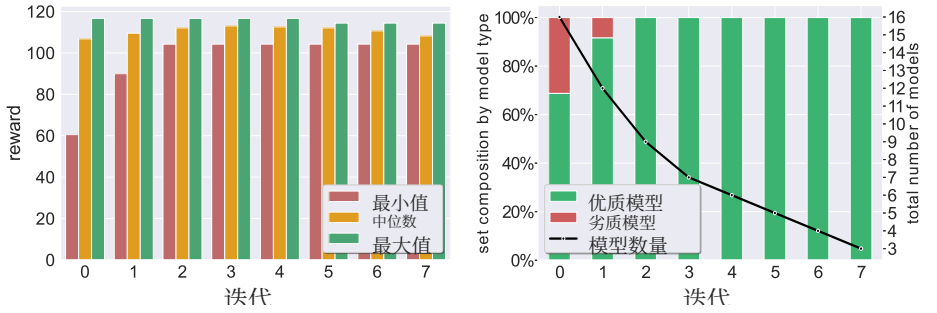
F.2 其他概率密度函数

以下是第 4.3 小节中讨论的结果。为了进一步展示我们的方法对不同类型分布外输入的鲁棒性，我们不仅将其应用于不同的值（例如，高发送速率值），还将其应用于相关分布外输入域中值的各种概率密度函数（概率密度函数）。更具体地说，我们使用不同的概率密度函数重复了分布外实验（实验 1 和实验 3）。在原始设置中，环境的所有参数（链路带宽、延迟等）均从范围 $[low, high]$ 中均匀抽取。然而，在本实验中，我们额外生成了两个概率密度函数：在范围 $[low, high]$ 内截断的截断正态（记为 $\mathcal{TN}_{[low,high]}(\mu, \sigma^2)$)分布。第一个概率密度函数



(a) *In-distribution* (b) *OOD*

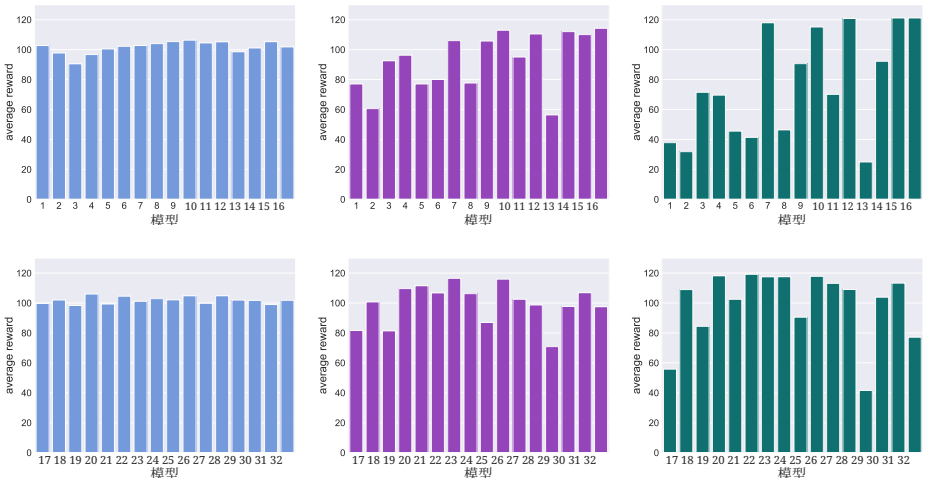
图 21：*Aurora* 实验 *3*：模型在不同分布上进行模拟时的平均奖励。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

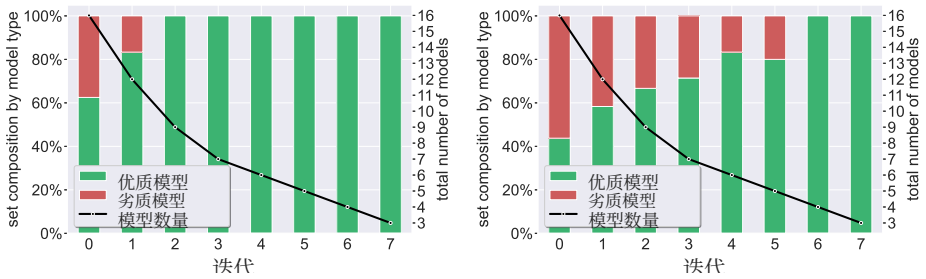
图 22：*Aurora* 实验 *3*：模型选择结果。

用于 $\mu_{low} = 0.3 * high + (1 - 0.3) * low$，另一个用于具有 a $\mu_{high} = 0.8 * high + (1 - 0.8) * low$。对于这两个概率密度函数，方差 σ^2 被任意设为 $\frac{high-low}{4}$。这些新分布如图 23 所示，并用于测试两批 Aurora 实验（实验 1 和实验 3）中的模型。



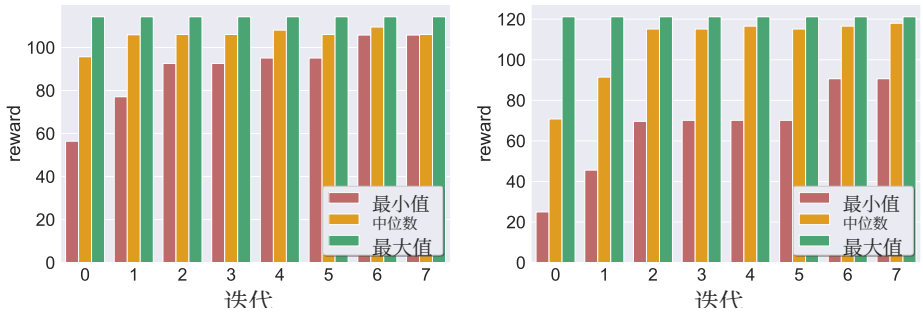
(a) *In-distribution* ($\sim \mathcal{U}(low, high)$)　(b) *OOD* ($\sim \mathcal{TN}(\mu_{low},\ \sigma^2)$)　(c) *OOD* ($\sim \mathcal{TN}(\mu_{high},\ \sigma^2)$)

图 23: *Aurora*：不同概率密度函数下模型的平均奖励。上排：实验 *1* 中使用的模型结果。下排：实验 *3* 中使用的模型结果。



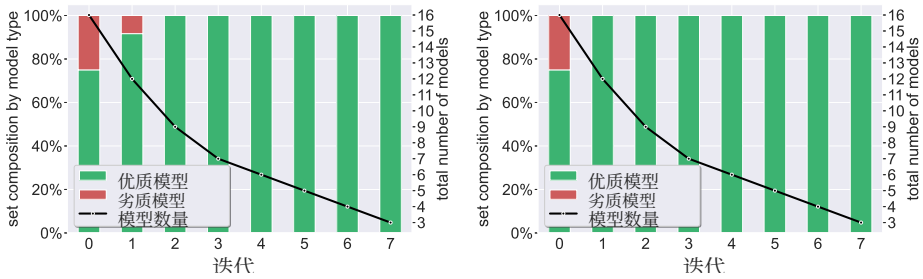
(a) $OOD \sim \mathcal{TN}(\mu_{low},\ \sigma^2)$　(b) $OOD \sim \mathcal{TN}(\mu_{high},\ \sigma^2)$

图 24: *Aurora*：其他 *PDF*：分布外值的模型选择结果；模型与实验 *1* 中的相同。



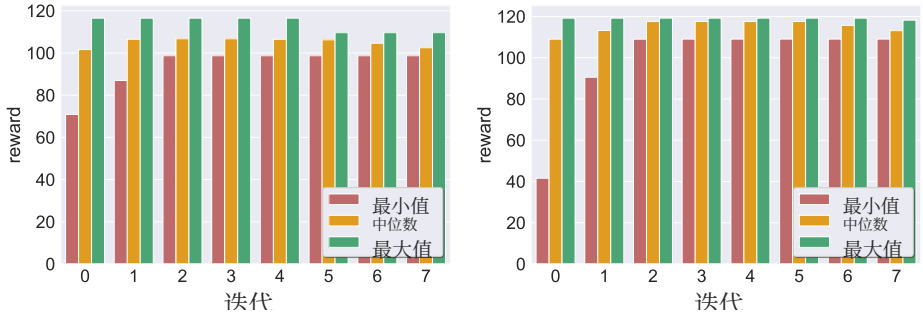
(a) $OOD \sim \mathcal{TN}(\mu_{low}, \sigma^2)$ (b) $OOD \sim \mathcal{TN}(\mu_{high}, \sigma^2)$

图 25: *Aurora*: 其他 *PDF*: 模型选择结果: 每次迭代的奖励统计; 这些模型与实验 *1* 中相同。



(a) $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$ (b) $\sim \mathcal{TN}(\mu_{high}, \sigma^2)$

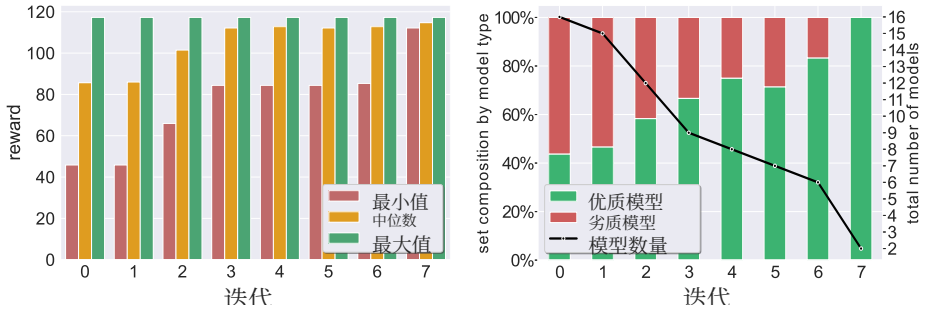
图 26: *Aurora*: 其他 *PDF*: 分布外值的模型选择结果; 这些模型与实验 *3* 中相同。



(a) $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$ (b) $\sim \mathcal{TN}(\mu_{high}, \sigma^2)$

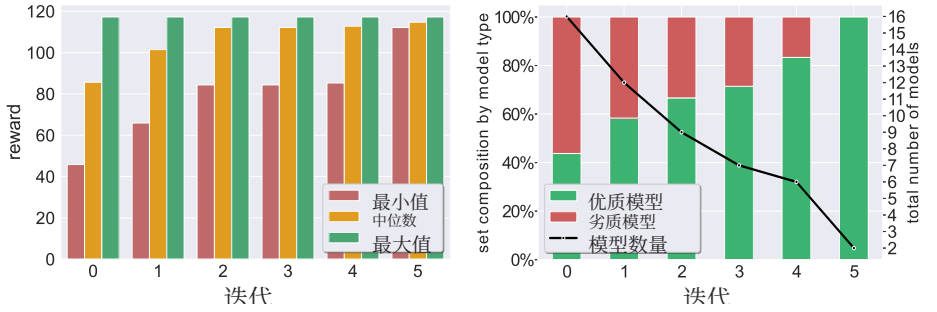
图 27: *Aurora*: 其他 *PDF*: 模型选择结果: 每次迭代的奖励统计; 模型与实验 *3* 中的相同。

F.3 附加筛选标准：实验 1



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

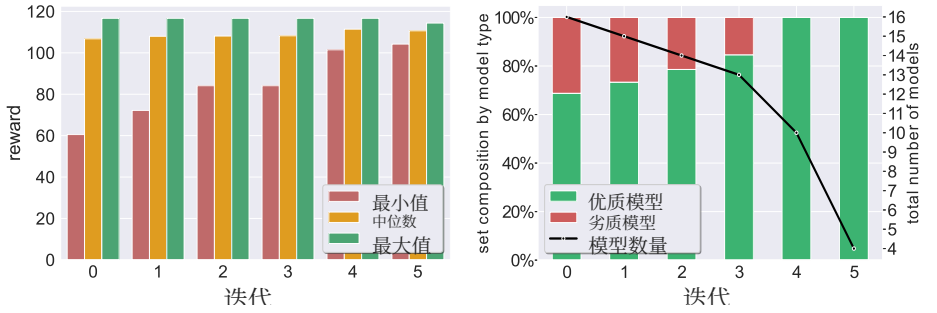
图 28：*Aurora* 实验 *1*：使用 `MAX` 筛选标准的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

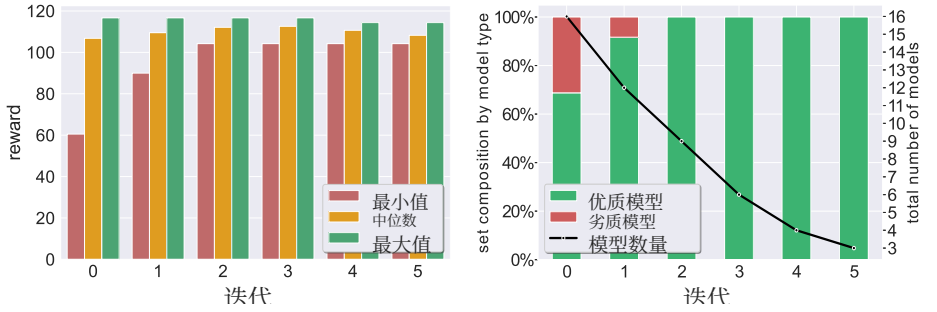
图 29：*Aurora* 实验 *1*：使用 `COMBINED` 筛选标准的结果。

F.4 附加筛选标准：实验 3



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

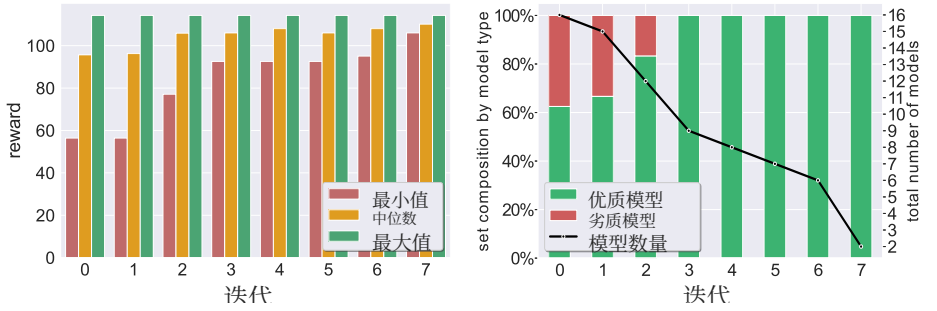
图 30： *Aurora* 实验 *3*：使用 `MAX` 筛选标准得出的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

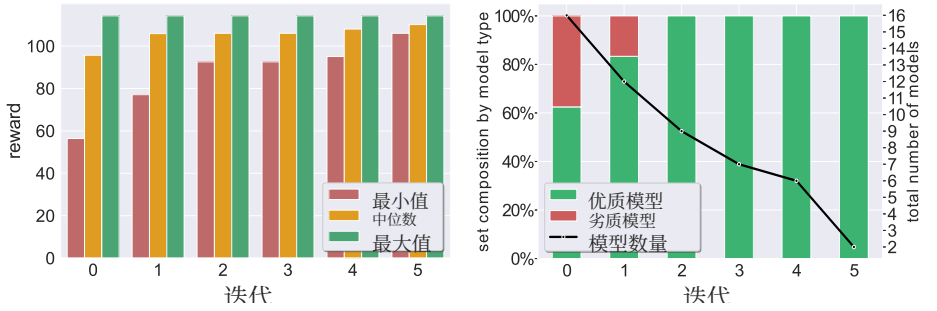
图 31： *Aurora* 实验 *3*：使用 `COMBINED` 筛选标准得出的结果。

F.5 附加筛选标准：附加概率密度函数



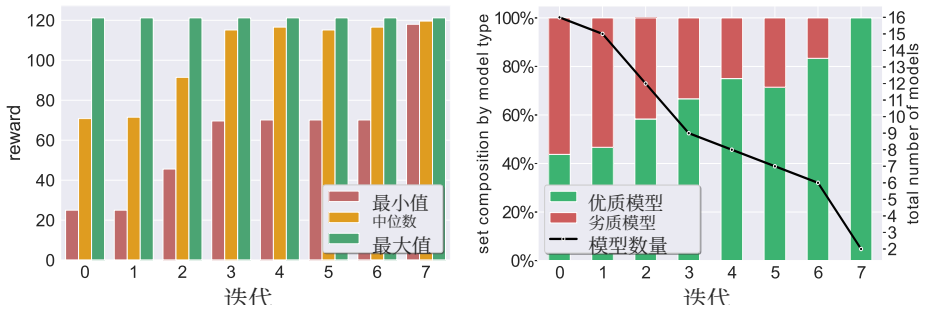
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 32: *Aurora* 实验 *1*：概率密度函数 $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$：采用 `MAX`筛选标准的结果。



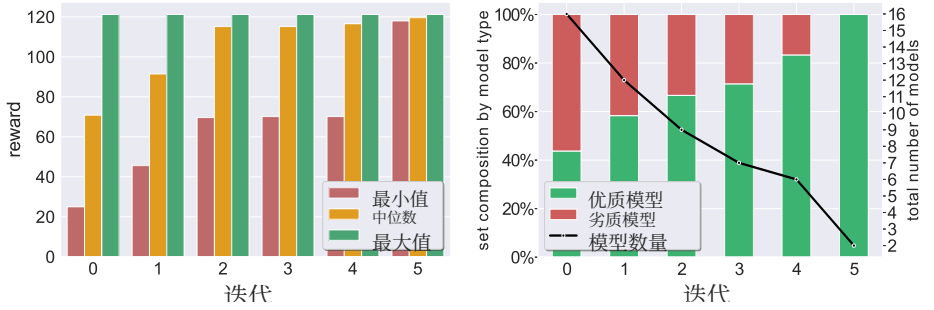
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 33: *Aurora* 实验 *1*：概率密度函数 $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$：采用`COMBINED`筛选标准的结果。



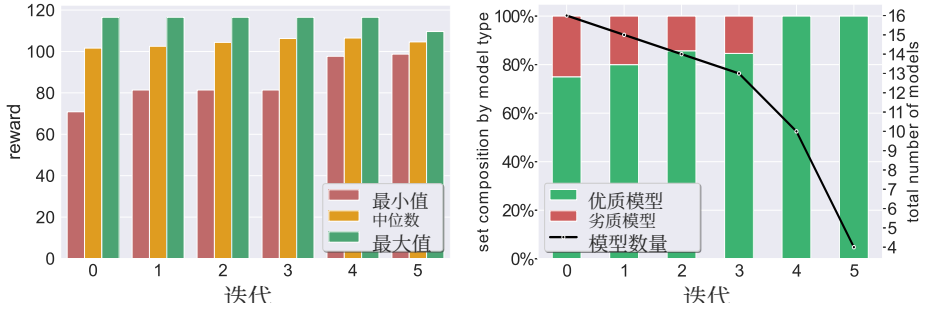
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 34: *Aurora* 实验 *1*: 概率密度函数 $\sim \mathcal{TN}(\mu_{high}, \sigma^2)$: 使用 `MAX`筛选标准的结果。



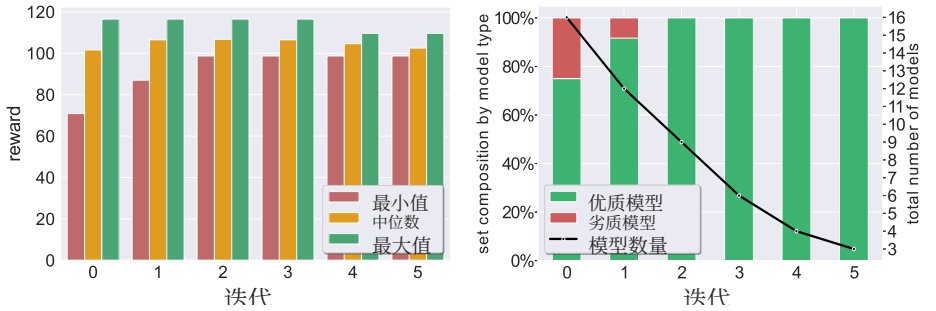
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 35: *Aurora* 实验 *1*: 概率密度函数 $\sim \mathcal{TN}(\mu_{high}, \sigma^2)$: 使用`COMBINED`筛选标准的结果。



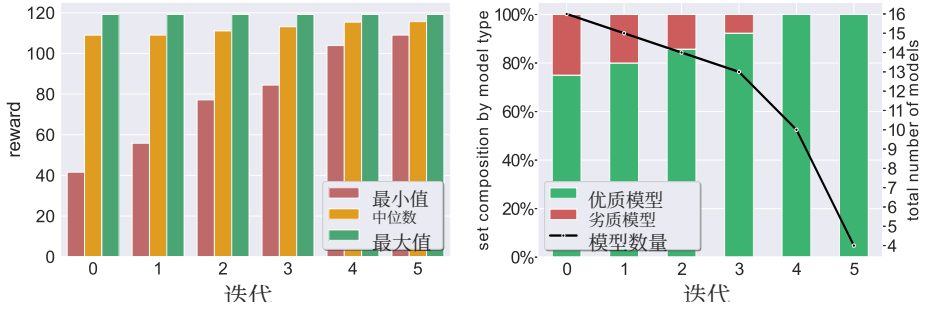
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 36: *Aurora* 实验 *3*: 概率密度函数 $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$: 使用 `MAX`筛选标准得到的结果。



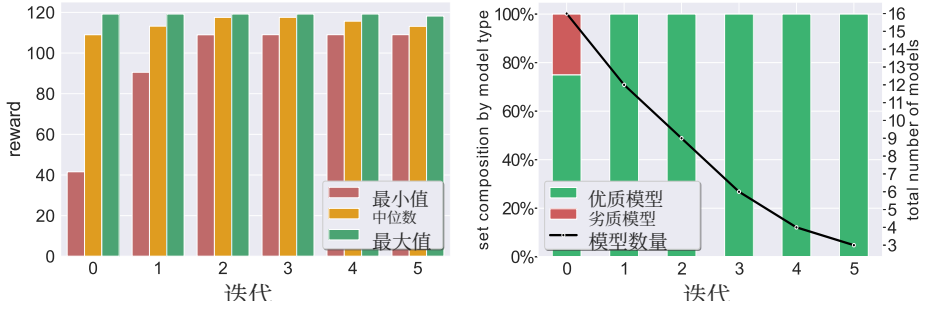
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 37: *Aurora* 实验 *3*：概率密度函数 $\sim \mathcal{TN}(\mu_{low}, \sigma^2)$*:* 使用`COMBINED`筛选标准的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 38: *Aurora* 实验 *3*：概率密度函数 $\sim \mathcal{TN}(\mu_{high}, \sigma^2)$*:* 使用 `MAX`筛选标准的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 39: *Aurora* 实验 *3*：概率密度函数 $\sim \mathcal{TN}\ (\mu_{high}, \sigma^2)$ ： 使用 `COMBINED` 筛选标准的结果。

G 与基于梯度的方法的比较

G.1 引言

本文介绍的方法有效地利用深度神经网络验证技术（算法1中的第4行）来解决一个优化问题：给定一对深度神经网络、预定义的输入域和一个距离函数，它们的输出之间的最大距离是多少？换言之，我们使用验证来寻找一个输入，使两个神经网络的输出差异最大，同时满足特定约束。尽管验证非常复杂 [53],，我们证明了它在我们的设置中至关重要。具体而言，我们展示了将验证替换为其他技术时我们的方法所取得的结果。优化非凸函数的一种技术是采用基于梯度的算法（“攻击”），例如梯度下降 [90], 、投影梯度下降（PGD） [69],等。然而，这些方法并不适合直接优化非平凡约束，因此需要进行修改，才能在我们的场景中奏效（参见附录B中的“距离函数”）。我们生成了三种梯度攻击：

1. 梯度攻击 *# 1*：一种非迭代式**快速梯度符号法 (FGSM)** [46]
攻击，用于优化线性约束（即 L_1 范数；见附录B），例如 Aurora 中的情况。2. 梯度攻击 *# 2*：一种**迭代式PGD** [69] 攻击，也用于优化线性约束（即 L_1 范数；见附录B），例如 Aurora 中的情况。我们指出，当先前的攻击在线性约束上失败时，我们会使用这种攻击。3. 梯度攻击 *# 3*：一种**约束迭代投影梯度下降** [69] 攻击，用于编码非线性约束（即*c*-距离函数；见附录B）的情况，例如 Cartpole 和 Mountain Car。

接下来，我们将这些受约束优化问题形式化。

G.2 形式化

在我们的研究中，我们始终关注一个输出空间 $\mathcal{O} = \mathbb{R}$。对于一个共享输入域 $\mathcal{D}$ 以及两个神经网络 $N_1 : \mathcal{D} \to \mathbb{R}$ 和 $N_2 : \mathcal{D} \to \mathbb{R}$，我们希望找到一个输入 $\boldsymbol{x} \in \mathcal{D}$，使这两个神经网络的输出差异最大化。

形式上，对于 L_1 范数，我们希望求解以下优化问题：

$$\begin{aligned} \max_{x} \quad & |N_1(\boldsymbol{x}) - N_2(\boldsymbol{x})| \\ \textbf{s.t.} \quad & \boldsymbol{x} \in \mathcal{D} \end{aligned}$$

梯度攻击 1 和 2。当仅存在输入约束时，可以通过传统的梯度方法求得局部最大值，即最大化以下目标函数：

$$L(\boldsymbol{x}) = |N_1(\boldsymbol{x}) - N_2(\boldsymbol{x})|$$

沿着其梯度方向迈出步，并将其投影到定义域 $\mathcal{D}$ 中，即：

$$\begin{aligned} \boldsymbol{x_0} &\in \mathcal{D} \\ \boldsymbol{x}_{t+1} &= [\boldsymbol{x}_t + \epsilon \cdot \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_t)]_{\mathcal{D}} \end{aligned}$$

其中，$[\cdot]_{\mathcal{D}}$：$\mathbb{R}^n \to \mathcal{D}$ 将结果投影到 $\mathcal{D}$，而 ϵ 是步长。我们注意到，$[\cdot]_{\mathcal{D}}$ 的实现可能并不简单；然而，对于 $\mathcal{D} \equiv \{\boldsymbol{x} \mid x \in \mathbb{R}^n \ \forall i \in [n] : \ l_i \leq x_i \leq u_i\}$，可以通过将每个坐标裁剪到其适当范围内来实现，而 $\boldsymbol{x_0}$ 则可以通过取 $\boldsymbol{x_0} = \frac{\boldsymbol{l}+\boldsymbol{u}}{2}$ 获得。

对于深度神经网络上的对抗攻击，即相对于其输入，最大化一对深度神经网络的损失函数，快速梯度符号法算法（梯度攻击 *# 1*）沿梯度方向单步移动。事实证明，这种简单的攻击在导致错误分类方面非常高效。该（投影）快速梯度符号法形式化如下：

算法 3 快速梯度符号法

输入： 目标函数 L，变量 $\boldsymbol{x}$，输入域 $\mathcal{D}$：（初始化，投影），步长 ϵ
输出： 对抗输入 $\boldsymbol{x}$

$\boldsymbol{x_0} \leftarrow \textsc{Init}(\mathcal{D})$
$\boldsymbol{x_{adv}} \leftarrow \textsc{Project}(\boldsymbol{x}_0 + \epsilon \cdot$**符号**$(\nabla_{\boldsymbol{x}} L(\boldsymbol{x}_0)))$
返回 $\boldsymbol{x_{adv}}$

在我们的算法中，我们通过两个函数定义 $\mathcal{D}$：INIT 从 $\mathcal{D}$ 返回一个初始值，而 PROJECT 则实现 $[\cdot]_{\mathcal{D}}$。

这种攻击的一种更强大的扩展是 PGD 算法，我们将其称为梯度攻击 *# 2*。这种攻击会迭代地沿着梯度方向移动，与单步攻击相比，通常能取得更好的结果。该攻击的形式化定义如下：

算法 4 投影梯度下降（最大化）

输入： 目标函数 L，变量 $\boldsymbol{x}$，输入域 $\mathcal{D}$：(INIT，PROJECT)，迭代次数 T，步长 ϵ
输出： 对抗输入 $\boldsymbol{x}$

$\boldsymbol{x_0} \leftarrow$INIT($\mathcal{D}$)
对于 $t = 0 \ldots T-1$ **执行**
    $\boldsymbol{x}_{t+1} \leftarrow$ 投影$(\boldsymbol{x}_t + \epsilon \cdot \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_t))$ ▷ **符号**$(\nabla_{\boldsymbol{x}} L(\boldsymbol{x}_t))$ 也可以使用
结束循环
返回 $\boldsymbol{x}_T$

我们指出，使用 PGD 来最小化 目标函数的情况具有对称性。

梯度攻击 3。 然而，在某些情况下，梯度攻击还需要优化一个损失函数，以表示对深度神经网络对输出 的约束。Cartpole 和 MountainCar 基准测试就是这种情况，在这些测试中我们使用了 c-距离函数。具体而言，在这种情形下，我们可能需要对以下形式的约束进行编码：

$$
\begin{aligned}
N_1(\boldsymbol{x}) &\leq 0 \\
N_2(\boldsymbol{x}) &\leq 0
\end{aligned}
$$

从而得到以下约束 优化问题：

$$
\begin{aligned}
\max_{x} \quad & |N_1(\boldsymbol{x}) - N_2(\boldsymbol{x})| \\
\textbf{s.t.} \quad & \boldsymbol{x} \in \mathcal{D} \\
& N_1(\boldsymbol{x}) \leq 0 \\
& N_2(\boldsymbol{x}) \leq 0
\end{aligned}
$$

传统的梯度攻击通常不用于求解此类优化问题。因此，我们定制了一种额外的梯度攻击（梯度攻击 # *3*），将迭代式PGD攻击与拉格朗日乘子 [87] $\boldsymbol{\lambda} \equiv (\lambda^{(1)}, \lambda^{(2)})$相结合，从而能够高效地求解此类问题，并对不满足约束的解施加惩罚。为此，我们定义了一个新颖的目标函数：

$$
L_{-}(\boldsymbol{x}, \boldsymbol{\lambda}) = |N_1(\boldsymbol{x}) - N_2(\boldsymbol{x})| - \lambda^{(1)} \cdot \mathrm{ReLU}(N_1(\boldsymbol{x})) - \lambda^{(2)} \cdot \mathrm{ReLU}(N_2(\boldsymbol{x}))
$$

从而得到以下优化问题：

$$
\begin{aligned}
\max_{x} \min_{\lambda} \quad & L_{-}(\boldsymbol{x}, \boldsymbol{\lambda}) \\
\textbf{s.t.} \quad & \boldsymbol{x} \in \mathcal{D} \\
& \lambda^{(1)} \geq 0 \\
& \lambda^{(2)} \geq 0
\end{aligned}
$$

并实现了一种约束迭代投影梯度下降算法，用于近似求解：

算法 5 约束迭代投影梯度下降

输入： 目标函数 L，输入域 $\mathcal{D}$，约束：$C_i(\boldsymbol{x})$，迭代次数：T，T_x，T_λ，步长：ϵ_x ,ϵ_λ

输出： 对抗输入 $\boldsymbol{x}$

$\boldsymbol{x_0} \leftarrow \textsc{Init}(\mathcal{D})$

$L_C(\boldsymbol{x}, \boldsymbol{\lambda}) \equiv L(\boldsymbol{x}) - \sum_{i=0}^{k} \lambda^{(i)} \cdot \text{ReLU}(C_i(\boldsymbol{x}))$ ▷ 新的目标函数

对于 $t = 0 \ldots T-1$ **执行**

$\boldsymbol{\lambda}_{t+1} \leftarrow \text{PGD}_{\textsc{min}}(L_C, \boldsymbol{\lambda}, (\boldsymbol{\lambda} \leftarrow 0, \boldsymbol{\lambda} \geq 0) T_\lambda, \epsilon_\lambda)$ ▷最小化 $L_C(\boldsymbol{x}_t, \boldsymbol{\lambda})$，其中 $\boldsymbol{x}_t$ 作为常数

$\boldsymbol{x}_{t+1} \leftarrow \text{PGD}_{\textsc{max}}(L_C, \boldsymbol{x}, \mathcal{D}, T_x, \epsilon_x)$ ▷最大化 $L_C(\boldsymbol{x}, \boldsymbol{\lambda}_t)$，其中 $\boldsymbol{\lambda}_t$ 作为常数

结束循环

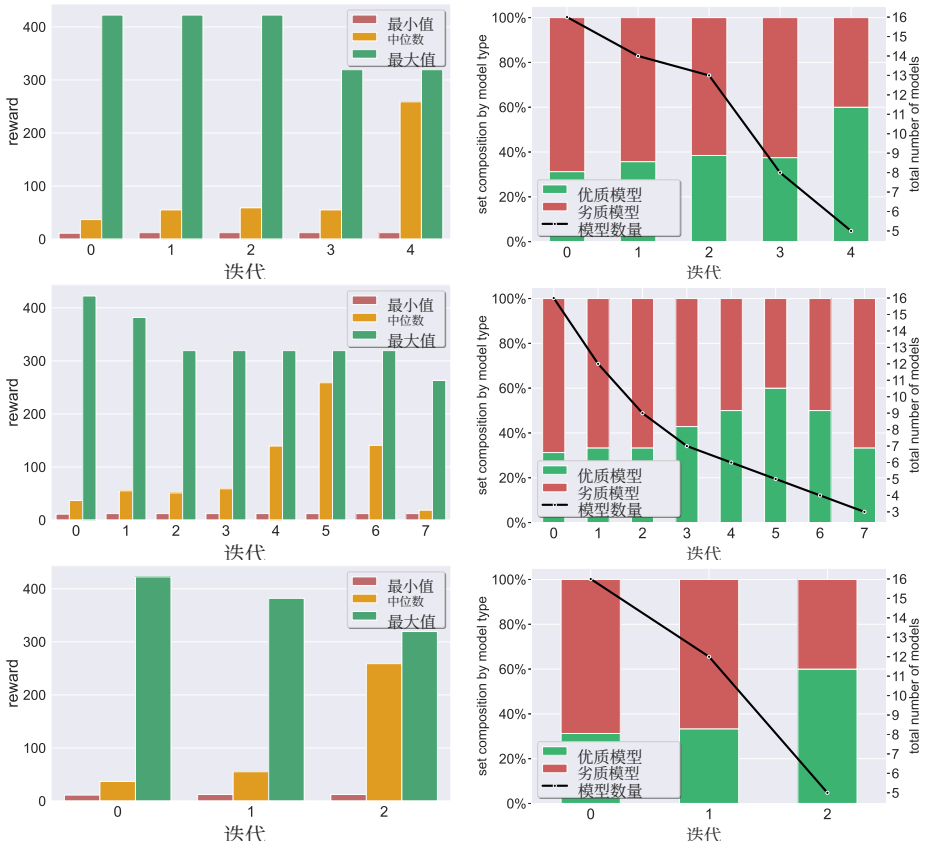
返回 $\boldsymbol{x}_T$

G.3 结果

如第 4.4 小节所述，我们在原始模型上运行了我们的算法，唯一的区别是用上述攻击替换了后端验证引擎（算法 1 第 4 行）。前两种攻击（即快速梯度符号法和迭代式PGD）用于 Aurora 的两批训练（“短”和“长”），第三种攻击（约束迭代投影梯度下降）则用于 Cartpole 和 Mountain Car。需要指出的是，对于 Aurora，我们仅在较弱的攻击失败时才运行迭代投影梯度下降攻击（因此，仅针对实验 1 中的模型）。结果（汇总于表 5）验证了，在大多数情况下，攻击要么失败，要么在预定义输入区域内错误地将劣质模型识别为优质模型。有关可行梯度攻击失败情形的更多详情，请参见图 40、图 41 和图 42。这些结果表明，在我们的方法中使用验证至关重要。

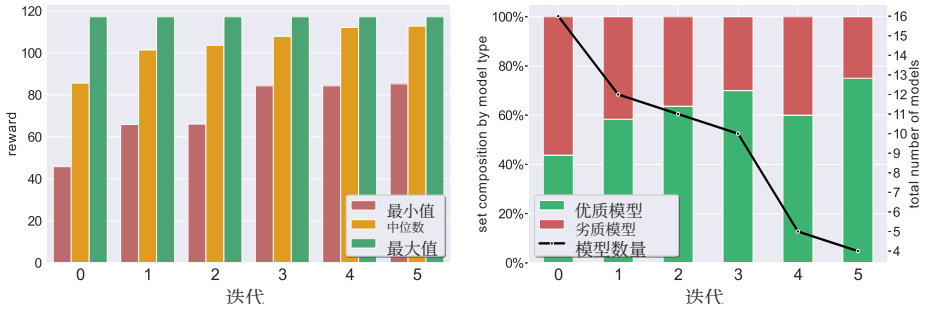
攻击	基准测试	可行 # 配对 # 对齐 # 未收紧 # 未通过的标准 成功						
1	Aurora（简称）	yes	120	70	50	0	MAX	no
							组合	yes
							百分位数	yes
1	Aurora（长）	yes	120	111	9	0	MAX	yes
							组合	yes
							百分位数	yes
2	Aurora（短）	yes	120	104	16	0	MAX	no
							组合	yes
							百分位数	yes
3	Mountain Car	no	120	38	35	47	MAX	no
							组合	no
							百分位数	no
3	Cartpole	部分地	120	56	61	3	MAX	no
							百分位数	no
							百分位数	no

表 5： 梯度攻击比较概要。前两列分别描述所选攻击及其评估所用的基准测试；第三列说明不完整攻击是否允许对所有 *PDT* 分数进行基于梯度的近似；接下来的四列依次表示 *DNN* 对总数、攻击返回的 *PDT* 分数与验证引擎收到的原始分数相同的配对数、攻击返回的分数精度低于验证引擎返回分数的配对数，以及攻击失败的案例数。倒数第二列显示筛选标准，最后一列则指出使用基于攻击的分数是否只得到了优质模型（使用验证时也是如此）。



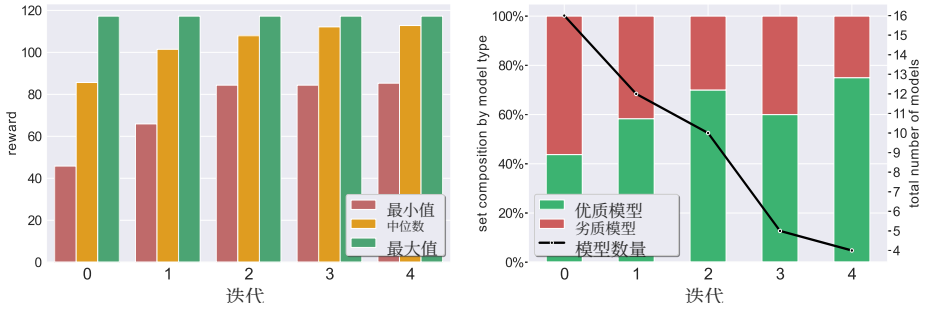
(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 40: *Cartpole*：攻击 *# 3*（约束迭代投影梯度下降）：使用PDT 分数筛选模型的结果，这些分数由梯度攻击 近似得出（而非使用验证引擎），对象为 *Cartpole* 模型。每行从上到下依次展示了使用不同筛选标准的结果（若分歧分数不大于2，则提前终止）： *PERCENTILE* （与图 *3* 和图 *10* 对比）， *MAX* （与图 *11* 对比），以及 *COMBINED* （与图 *12* 对比）。在所有情况下，该算法都会返回至少一个劣质模型（通常不止一个），导致这些模型的平均奖励低于我们基于验证的结果。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 41: *Aurora*: 攻击 *# 1*（单步快速梯度符号法）：在短时训练的 *Aurora* 模型上，使用 *MAX* 准则筛选模型，并以梯度攻击 近似PDT 分数（而非使用验证引擎）所得的结果（若分歧分数不大于2，则提前终止）。与我们的验证驱动方法相比，最终结果中包含一个劣质模型。请与图 *28* 对比。



(a) *Rewards statistics* (b) *Remaining good and bad models ratio*

图 42: *Aurora*: 攻击 *# 2*（迭代式*PGD*）：在短时训练的 *Aurora* 模型上，使用 *MAX* 准则筛选模型，并以梯度攻击 近似PDT 分数（而非使用验证引擎）所得的结果（若分歧分数不大于2，则提前终止）。与我们的验证驱动方法相比，最终结果中包含一个劣质模型。请与图 *28* 对比。