Reinforcement Learning Final Project

pangbo

2024 年 2 月 19 日

1 Value-based Reinforcement Learning

1.1 方法

基于价值的强化学习是一种强化学习方法，其核心思想是通过对价值函数进行建模和估计，以此为依据制定策略。在训练阶段，其学习目标是学到一个函数，知道当前状态和动作之后，这个函数可以输出状态下这个动作所能带来的长期价值期望，记为 Q 值。

在决策阶段，我们可以根据训练好的函数，在一个新的状态下，尝试可选动作集合中的每一个动作，最终采取 Q 值最大的动作，这样就可以带来最大的长期收益。

DQN 是基于价值的强化学习的一个典型代表，其核心思想是通过神经网络来估计 Q 值函数。在训练阶段，我们通过神经网络来估计 Q 值函数，然后通过 Q 值函数来制定策略。在决策阶段，我们可以根据训练好的神经网络，在一个给定状态下，采取 Q 值最大的动作。

原始 DQN 在实际运用中面临的一大挑战是难以收敛，因此，研究人员提出了许多改进方法以提高 DQN 的性能。

1.1.1 Double Q-learning

原始 DQN 的更新公式如下：

$$Q(s_t, a_t) \leftarrow r_{t+1} + \max_a Q(s_{t+1}, a) \tag{1}$$

可以看出，原始 DQN 需要使用 Q 函数的值来更新 Q 函数，而 Q 函数是在更新过程中不断变化的，这导致神经网络的训练目标不断变化，不利于神经网络的收敛。另一个问题在于，Q 函数可能高估某些动作的 Q 值，而使用单个 Q 函数进行训练的方式难以修复这种问题。

为解决这些问题，研究人员提出了 Double Q-learning 的方法。Double Q-learning 的核心思想是使用两个 Q 函数来分别估计 Q 值，然后使用其中一个 Q 函数来选择动作，使用另一个 Q 函数来估计 Q 值。两个 Q 函数轮流进行更新。具体来说，Double Q-learning 的更新公式如下：

$$Q(s_t, a_t) \leftarrow r_{t+1} + Q'\left(s_t, \arg\max_a Q(s_{t+1}, a)\right) \tag{2}$$

1.1.2 Dueling DQN

Dueling DQN 的核心思想是将 Q 函数分解为状态值函数和优势函数，即：

$$Q(s, a) = V(s) + A(s, a) \tag{3}$$

其中，$V(s)$ 表示状态值函数，$A(s,a)$ 表示优势函数。状态值函数表示状态 s 自身的价值，而优势函数表示在状态 s 下，采取动作 a 相对于采取其他动作所能带来的长期收益的期望。

Dueling DQN 可以更好地对状态价值进行评估，拥有更快的学习速度和更好的收敛性。在具有多个冗余或者近似的动作时，Dueling DQN 可以比 DQN 更快的识别出策正确动作。

1.1.3 Multi-step Learning

Q-learning 一般使用时序差分（Temporal Difference）的方法来更新 Q 值，即只考虑下一步的 Q 值。Multi-step Learning 的核心思想是在更新 Q 值时，不仅考虑下一步的 Q 值，而是考虑未来多步的 Q 值。Multi-step Learning 是一种介于时序差分与蒙特卡洛方法之间的方法，可以更好地平衡学习速度和收敛性。其更新公式如下：

$$Q(s_t,a_t) \leftarrow \sum_{i=1}^{m} \gamma^{i-1} r_{t+i} + \max_a Q(s_{t+m},a) \tag{4}$$

1.1.4 Noisy Net

在 DQN 的训练过程中，我们需要不断地探索新的状态和动作，以便更好地估计 Q 值函数。ϵ-greedy 方法是一种常用的探索方法，其核心思想是以 ϵ 的概率随机选择动作，以 $1-\epsilon$ 的概率选择 Q 值最大的动作。在刚开始训练时，我们需要大量地探索新的状态和动作，因此 ϵ 的值需要设置的较大，随着训练的进行，我们需要逐渐减小 ϵ 的值，以便更多地选择 Q 值最大的动作。

Noisy Net 的核心思想是直接在神经网络中引入噪声，以便更好地探索新的状态和动作。相较于 ϵ-greedy 方法，Noisy Net 拥有更大的不确定性、更好的探索性和更快的收敛性。

1.1.5 Prioritized Experience Replay

在 DQN 的训练过程中，我们需要不断地从历史回放中均匀随机采样，以便训练神经网络。Prioritized Experience Replay 的核心思想是在采样时，不再是均匀随机采样，而是根据经验的优先级进行采样。样本的优先级可以根据样本在训练过程中的损失计算，即每一轮从 Replay Buffer 中选择样本输入模型后，使用模型预测结果的损失反过来更新该样本的优先级，损失越大，优先级越高。总而言之，Prioritized Experience Replay 可以更好地利用经验，提高训练效率。

1.1.6 Rainbow

Rainbow 将 DQN 的多种改进方法进行组合，以获得更好的性能。Rainbow 的改进方法包括 Double Q-learning、Dueling DQN、Multi-step Learning、Noisy Net、Distributional RL 和 Prioritized Experience Replay。将这些改进方法组合起来，可以获得比只采用部分改进方法更好的性能。

1.2 实验

参照 Rainbow 方法，我在 DQN 的基础上实现了 Double Q-learning、Dueling DQN、Multi-step Learning、Noisy Net 和 Prioritized Experience Replay。我使用 VideoPinball-ramNoFrameskip-v4 环境进行了测试，模型在此环境下的表现如图 1所示。

图 1中横轴代表进行的游戏局数，纵轴代表游戏的得分，图中背景的蓝色曲线代表每一局游戏的得分原始数据曲线，由于每一局游戏的得分波动极大，为方便看出整体趋势，我对原始数据进行

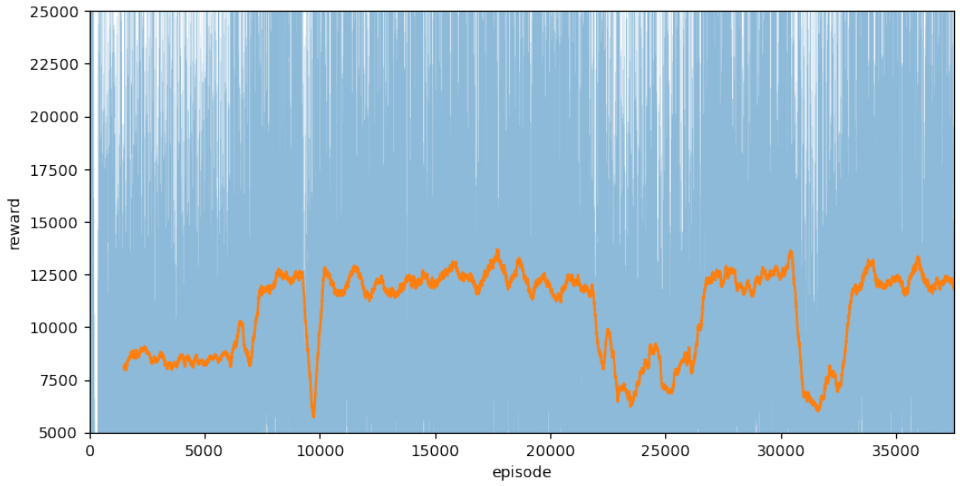

图 1: Rainbow 在 VideoPinball-ramNoFrameskip-v4 环境下的表现

了滑动平均处理，结果为图中的橙色曲线。可以看出，虽然在训练过程中仍然存在较大的波动，我们的模型整体上呈现出上升趋势，模型在与环境的交互过程中逐渐学到了更好的策略。

然后，我还设计了消融实验，测试 Rainbow 中的每一种改进方法对模型性能的影响。我将 Rainbow 中的每一种改进方法分别去除，然后在 VideoPinball-ramNoFrameskip-v4 环境下进行测试，每个模型训练 10000 局游戏，由于训练轮数较少，我增大了训练前期随机探索的概率，这导致曲线可能出现更大的波动。消融实验的结果如图 2所示。

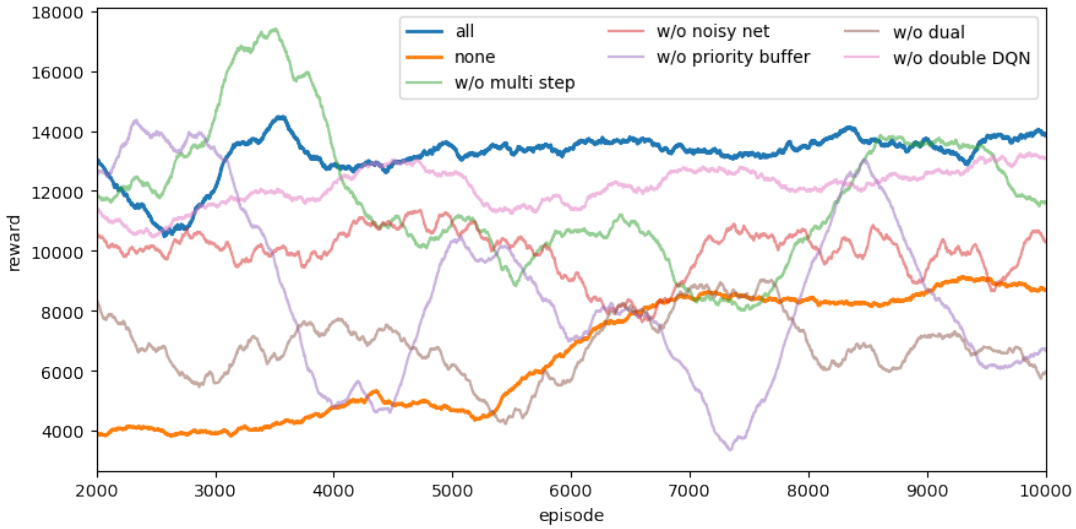

图 2: Rainbow 中改进方法对模型性能的影响

图 2中，all 曲线表示使用了我实现的所有改进方法的模型表现，none 曲线为原始 DQN 方法模型的表现，其余曲线代表去除单个改进方法后模型的表现。可以看到，在 5000 局游戏之后，采用了全部改进方法的模型获得了最好的表现。原始 DQN 在一开始时表现最差，但是随着训练轮数增加，其表现有一个较为稳定的提升，最终超过了部分其他模型。

图 2中曲线波动较大，为了更直观展示出模型之间的性能差异，我根据最后 500/1000 局游戏的平均得分绘制了条形统计图，如图 3所示，图中可以较为直观的看出不同模型之间的差异。

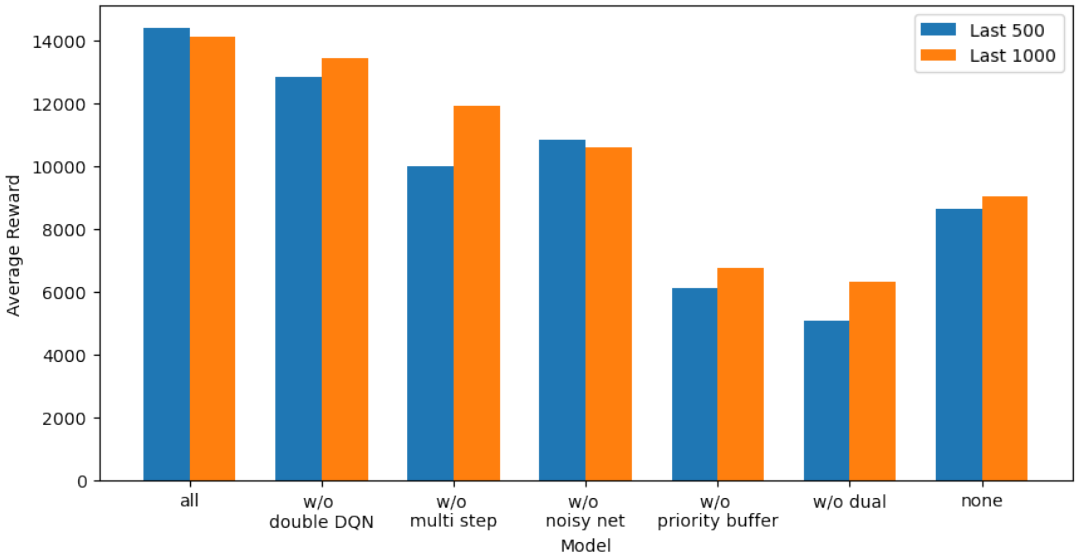

图 3: Rainbow 中改进方法对模型性能的影响

我还在其他环境中测试了我实现的算法，图 4是在 BreakoutNoFrameskip-v4 环境下的测试结果，由于图像版本的环境需要消耗大量计算量在模型的卷积部分，计算量过大导致训练缓慢，而这部分本身和强化学习没有太大关系，难以测试出模型随着与环境交互次数增加的性能变化。

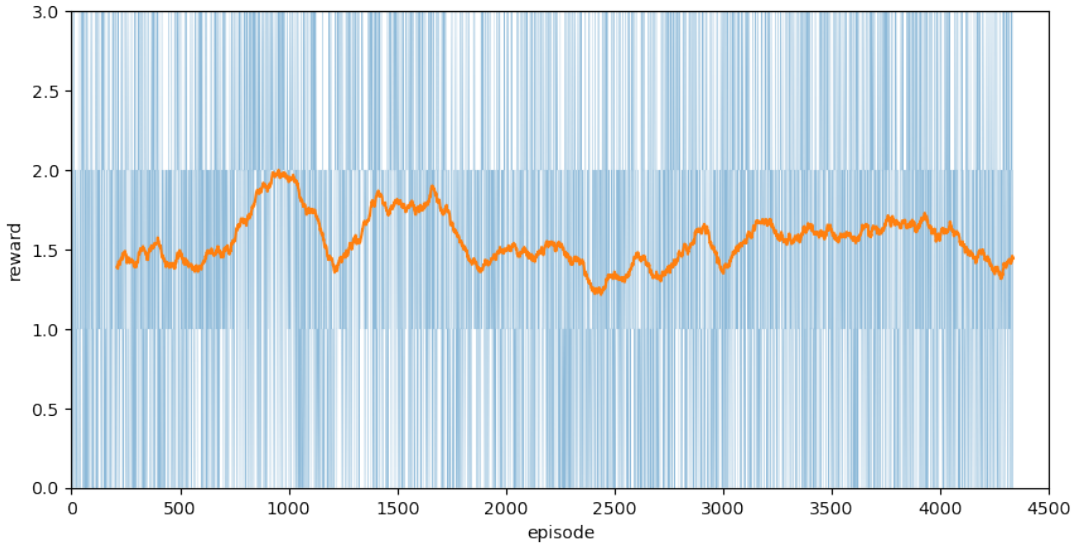

图 4: BreakoutNoFrameskip-v4 环境测试结果

出于计算量考虑，我改用了 ram 版本的环境，图 5和图 6分别展示了 Breakout-ramNoFrameskip-v4 和 Pong-ramNoFrameskip-v4 环境下的测试结果。模型与每个环境交互 50000 局游戏。

从图 5看到，我们的模型在 Breakout-ramNoFrameskip-v4 环境下的表现并不理想，训练开始便出现了一次性能下降，后续表现虽然有缓慢的提升的迹象，但整体提升不明显。相比之下，模型在 Pong-ramNoFrameskip-v4 环境下的表现要好不少，从图 6中可以明显看到模型的性能有一个较为稳定的提升趋势；但是我们也必须指出，注意到纵坐标的得分，虽然有稳定的提升趋势，这个模型的表现依然是非常差的。

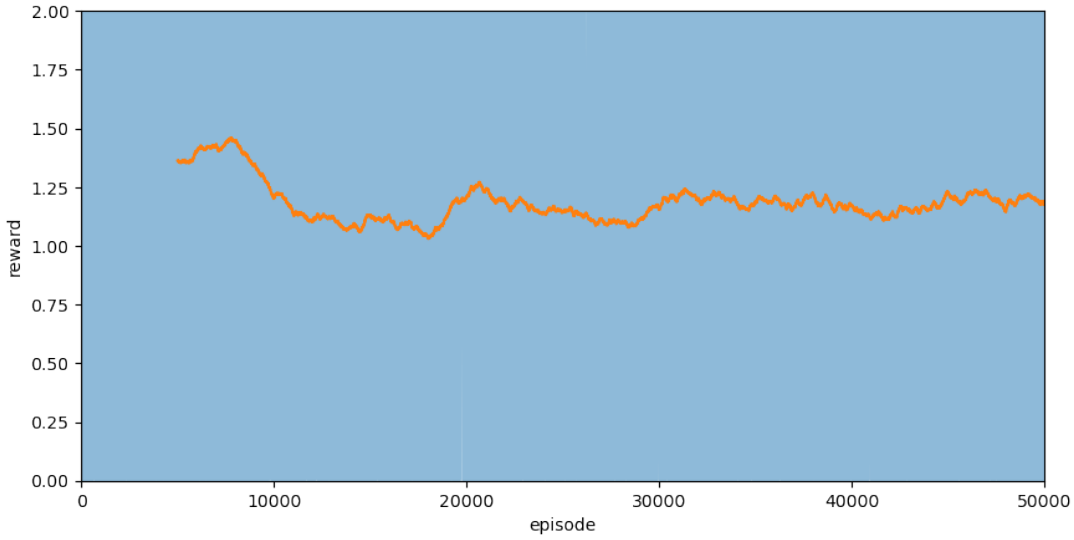

图 5: Breakout-ramNoFrameskip-v4 环境测试结果

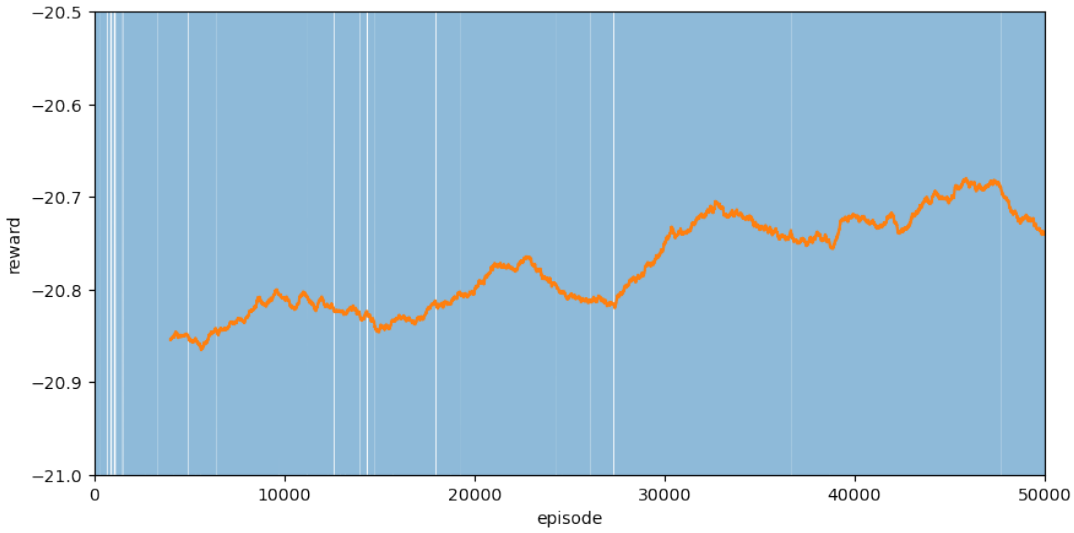

图 6: Pong-ramNoFrameskip-v4 环境测试结果

我的强化学习模型在 Breakout 与 Pong 游戏中的表现明显差于 VideoPinball 游戏，我认为可能得原因包括：

- VideoPinball 状态相较于其余游戏更为简单，模型只需要关注弹球的位置，而 Breakout 与 Pong 游戏不仅有大量状态变化，还可能存在对手。
- VideoPinball 游戏拥有更高容错率，少量错误决策对游戏的影响较小甚至完全没有影响，而 Breakout 与 Pong 游戏对错误决策非常敏感。
- VideoPinball 游戏决策之间的时序关联性较弱，模型只需要关注当前帧做出决策，不需提前为后续状态行动。Breakout 与 Pong 游戏都需要移动“球拍”，这需要模型提前做出正确决策。
- VideoPinball 游戏中的模型有更多机会进行探索，因为对于 VideoPinball 游戏而言，完全随机动作也可以获得较高的得分，模型有更多机会接触到新的状态和动作。而 Breakout 与 Pong 游戏中，完全随机动作几乎不可能获得高分，模型的探索性较差。

模型提升缓慢是上述所有环境的共同特点，这也是强化学习面临的一大挑战。由于本项目要求必须使用 model-free 方法，我们的模型只能等待环境给出的奖励，但奖励可能是稀疏且滞后的，这导致模型非常难评估单个动作的好坏，因此模型的学习速度非常慢。如果允许使用 model-based 方法，我们可以根据对环境的理解，人为加入奖励，定向诱导模型向正确的方向学习，这样可以大大提高模型的学习速度。

2 Policy-based Reinforcement Learning

2.1 方法

基于策略的强化学习是一种强化学习方法，其核心思想是通过对策略进行建模和估计，以此为依据制定策略。在训练阶段，其学习目标是学到一个函数，知道当前状态之后，这个函数可以输出状态下每个动作的概率，记为策略。在决策阶段，我们可以根据训练好的函数，在一个新的状态下，根据策略来选择动作。

相较于基于价值的强化学习，基于策略的强化学习不需要估计 Q 值函数便可以直接估计出策略，因此可以用于连续决策空间问题的策略预测。

2.1.1 Policy Gradient

对于给定的决策模型参数 θ，在任意状态下，决策模型会给出选择每个动作的概率，记为 $\pi_\theta(a|s)$，同时采取每个动作之后可以获得 $R(s,a)$ 的收益。对于一个状态 s，当前决策模型的期望收益可以表示为：

$$R_\theta(s) = \sum_a R(s,a) p_\theta(a|s) \tag{5}$$

我们的优化目标位最大化期望收益，上式中动作的收益 $R(s,a)$ 与决策模型无关，因此我们需要优化 $p_\theta(a|s)$。对 R_θ 求梯度有

$$\nabla R_\theta = \sum_a R(s,a)\nabla p_\theta(a|s) \tag{6}$$

$$= \sum_a R(s,a)p_\theta(a|s)\frac{\nabla p_\theta(a|s)}{p_\theta(a|s)} \tag{7}$$

$$= \sum_a R(s,a)p_\theta(a|s)\nabla \log p_\theta(a|s) \tag{8}$$

使用梯度下降更新参数，其中 η 为学习率：

$$\theta \leftarrow \theta + \eta\nabla R_\theta \tag{9}$$

2.1.2 Actor-Critic

在等式 8中，我们需要正确计算出动作的收益 $R(s,a)$ 才能正确更新模型。然而，我们不仅需要关注单步动作的奖励，还需考虑后续状态的价值，如果使用蒙特卡洛方法进行估计，则有 $R(s,a) = \sum_{t'=t}^{T}\gamma^{t'-t}r_{t'}$。在实际问题中，这要求我们必须等到整个序列结束后才能进行更新，也要求任务必须在有限步内结束。

Actor-Critic 方法借鉴基于价值的强化学习，尝试使用模型对 $R(s,a)$ 的值进行估计。Actor-Critic 方法将决策模型分为两部分：Actor 用于估计策略，Critic 用于估计价值。

在 Actor-Critic 方法方法下，$R(a,s)$ 可以写为：

$$R(a,s) = r + \gamma V(s') - V(s) \tag{10}$$

其中 r 为状态 s 下行动 a 带来的即时奖励，V 为 Critic 模型，$V(s)$ 为状态 s 的价值，$V(s')$ 为状态 s' 的价值。

记 Critic 模型的参数为 ω，其损失函数为

$$L(\omega) = \frac{1}{2}(r + \gamma V_\omega(s_{t+1}) - V_\omega(s_t))^2 \tag{11}$$

其梯度为：

$$\nabla_\omega L(\omega) = -(r + \gamma V_\omega(s_{t+1}) - V_\omega(s_t))\nabla_\omega V_\omega(s_t) \tag{12}$$

2.1.3 Deep Deterministic Policy Gradient

DDPG 是一种基于策略的强化学习方法。与 Policy Gradient、Actor-Critic 不同，DDPG 是一种确定性的策略，即在给定状态下，DDPG 会输出一个确定的动作，而不是一个动作的概率分布，这使得 DDPG 可以用于连续动作空间的问题。此外，DDPG 还是一种 off-policy 的方法，即与环境交互的策略和更新的策略可以不同，这提高了 DDPG 的训练效率。

DDPG 和 Actor-Critic 一样使用了两个模型分别估计策略和价值。为了避免价值估计模型高估价值的问题，DDPG 使用了与 Double Q-learning 类似的方法，即使用两套模型轮流进行更新。

2.2 实验

我在多个环境下测试了我实现的 DDPG 算法。HalfCheetah-v4 环境下的测试结果见图 7。Ant-v4 环境下的测试结果见图 8。Hopper-v4 环境下的测试结果见图 9。Humanoid-v4 环境下的测试结果见图 10。

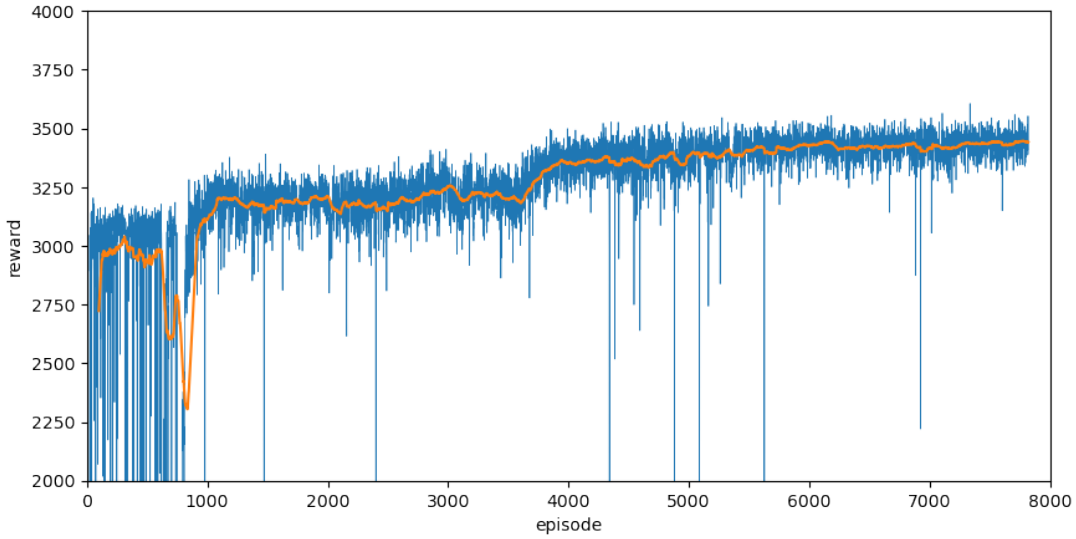

图 7: HalfCheetah-v4 环境测试结果

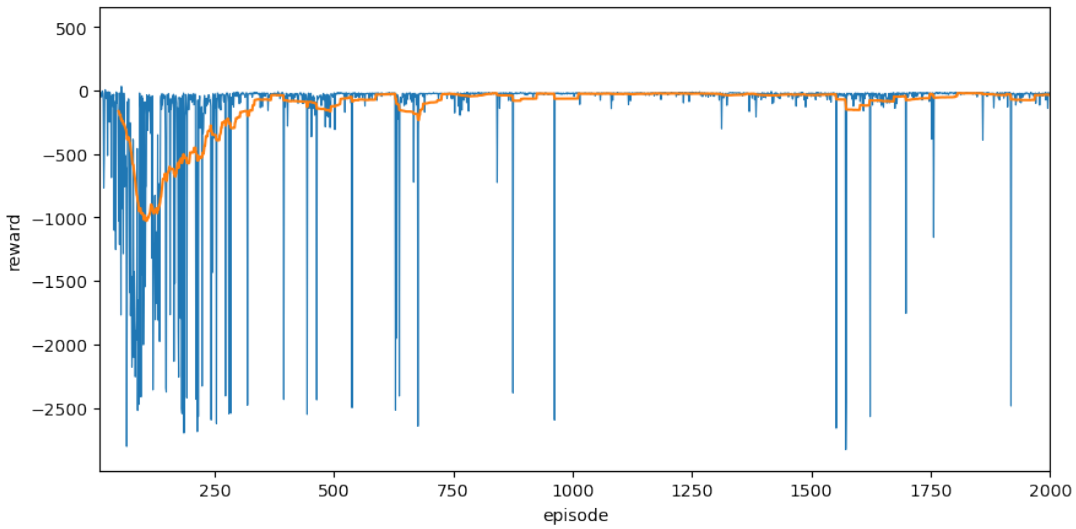

图 8: Ant-v4 环境测试结果

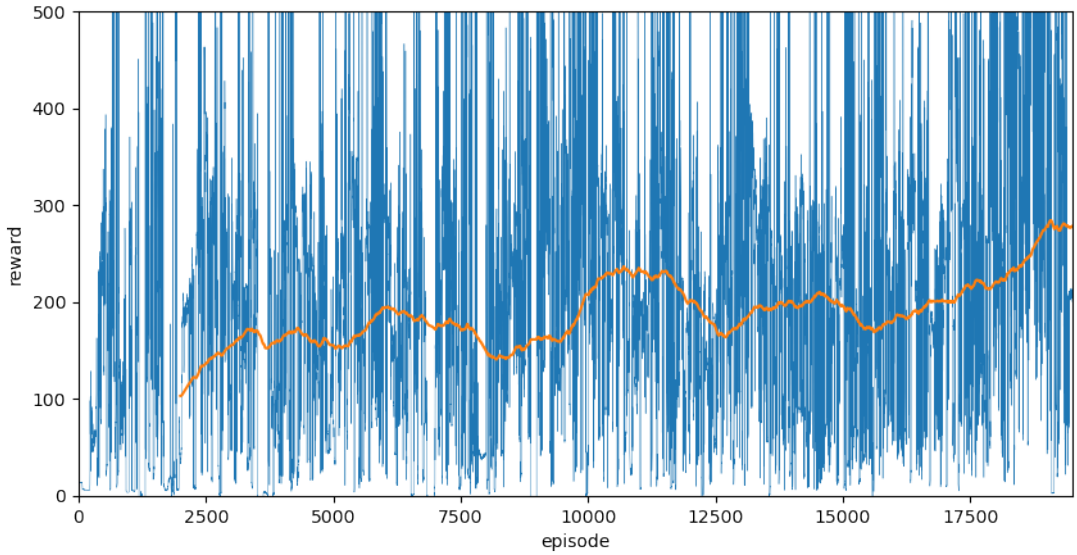

图 9: Hopper-v4 环境测试结果

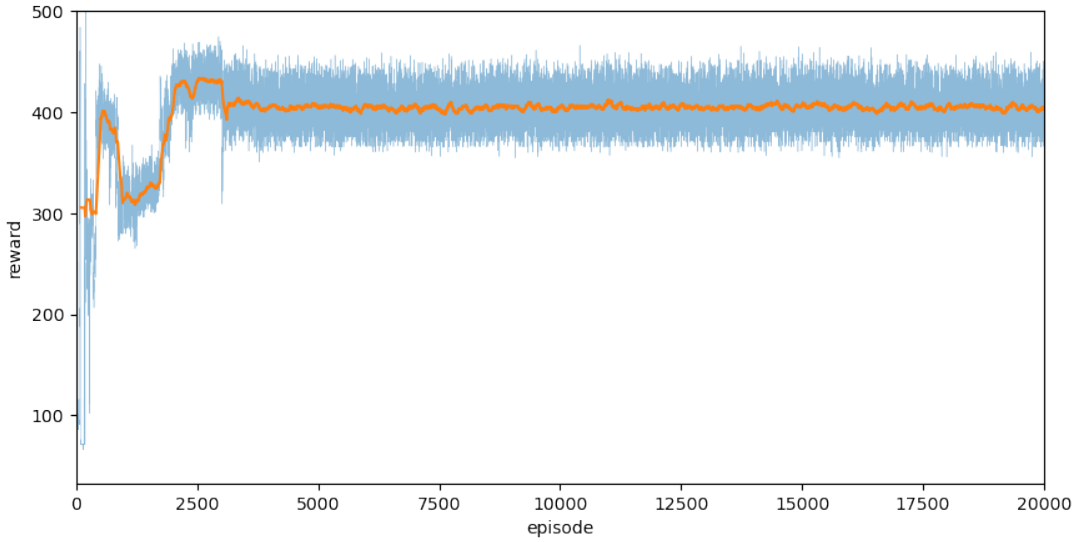

图 10: Humanoid-v4 环境测试结果

整体上看，DDPG 在所有测试环境下都取得了不错的表现，随着与环境交互轮数增加，模型表现呈现出提升趋势。而不同环境之间也存在一些差异，例如图 9中，每一局游戏的得分波动较大，只能从平滑曲线中看出整体平均得分在上升。图 8与图 10中，在游戏轮数达到一定值后，模型提升趋势出现停滞现象，这可能是因为随着轮数增加，模型随机性降低，模型探索范围减小，导致模型陷入局部最优解，一种可能的改进方案是当检测到模型提升停滞时，强制增大模型决策的随机性，让模型有更多机会探索新的状态和动作。