

语言评判模型作为 MuJoCo 中 PPO 的奖励来源： Kev-4B 与同等规模的自回归评判模型

DC-Jev-RL 项目，第一部分

2026 年 10 月

摘要

一个读取机器人行为文字摘要的 40 亿参数语言评判模型，能否替代 MuJoCo 任务中为 PPO 设计的工程化奖励？我们比较 Kev-4B——一种在一次前向传播中回答关于 JSON 状态的问题的决策模型——与读取同样文本、同家族同规模的自回归（AR）评判模型（Qwen3.5-4B）。我们在五个任务上用七种奖励来源训练 PPO，每种 5 个种子（共 165 个 run，每个 10^6 步）。加上健康门控后，Kev-4B 奖励的归一化分数四分位均值为 1.00 [0.81, 1.09]，工程化奖励为 1.05。用 Kev-4B 在十分之一窗口上的标签蒸馏出的奖励网络达到 0.98，AR 评判的奖励为 0.35。汇总掩盖了一个失败：在 Reacher 上，每种评判奖励的最终结果都低于随机策略。AR 评判的奖励让 Hopper 和 Walker2d 学会了站着不动，尽管它的分数给窗口排序的能力几乎和 Kev-4B 一样好：在 Hopper 上，它给一个直立而缓慢的窗口的报酬是快速窗口的 0.87，Kev-4B 为 0.48，工程化奖励为 0.31。没有任何评判奖励让猎豹翻成背朝下，而工程化奖励在五个 run 中有 1 个翻倒。由于等待一个被最多 24 个 run 共享的评判模型，每个 run 慢了约 17 倍。代码、日志和本文的每一个数字都在仓库中。

1 引言

强化学习需要奖励，而大多数任务的奖励都得有人来写。MuJoCo 运动控制基准自带工程化的奖励：前进速度加上存活奖励，再减去控制代价，各项权重靠人工调出。如果一个语言模型能根据任务描述来评判行为，这一步就可以省掉。这个想法已有多种形式：语言模型编写奖励代码 [7, 16]；在文本环境中直接充当奖励函数 [5]；视觉-语言模型为机器人姿态的图像打分 [10]；把语言模型对事件描述的偏好蒸馏成内在奖励 [4]。

本文研究这个问题的一个窄版本，并配齐回答它所需的对照。设想一个 40 亿参数的评判模型读一段文字摘要，内容是机器人在 32 步窗口内的表现。用它的分数训练 PPO，能否替代标准基准的工程化奖励？代价又是多少？我们比较同一家族、同样大小的两个评判模型。一个是 Kev-4B [8]，一种决策模型：对 JSON 状态提出的问题，它一次前向传播就给出答案，带指针头和校准温度。另一个是自回归（AR）评判模型 Qwen3.5-4B，以同样的方式读出它在各答案选项上的下一词元分布。两个评判模型看到的文本完全相同。另设一个对照条件 `env_window`：用真实奖励，但时间分辨率与评判模型相同。它把“按窗口给反馈”的影响与“评判模型本身”的影响分开。

贡献。

- 在五个 MuJoCo 任务上，对 PPO 的七种奖励来源做受控比较：工程化奖励；它的窗口平均版本；Kev-4B 的分数，带或不带“健康”门控；由 Kev-4B 在 10% 窗口上标注、在线蒸馏出的奖励模型；以及 AR 评判模型的期望读出和采样读出。每个任务训练 100 万步、5 个种子（采样读出的 AR 评判为 3 个），统计方法依照 Agarwal et al. [1] (§7)。
- 一项离线一致性研究：在同一批窗口上，把每种评判信号与环境奖励联系起来 (§6)。
- 对最终策略实际行为的分析，它解释了主要差异 (§7.1)：AR 评判模型的奖励让 Hopper 和 Walker2d 学会了站着不动，因为它给一个直立而缓慢的窗口的分数几乎和快速窗口一样高；没有任何评判奖励让猎豹翻成背朝下，而工程化奖励会；在 Reacher 上，每种评判奖励都让机械臂乱甩。
- 实测代价：每个 run 被评判的窗口数、评判耗时和总耗时 (§8)。

2 相关工作

语言模型作为奖励设计者和奖励函数。 Eureka [7] 和 Language to Rewards [16] 用语言模型编写或参数化奖励代码。这样策略的奖励仍然便宜，但设计工作转移到了模型的编程能力和外层搜索循环上。像 Kwon et al. [5] 在谈判等文本游戏中那样直接把模型当作奖励，不需要代码，但训练期间就要调用模型。VLM-RM [10] 用 CLIP 为人形机器人渲染画面中的姿态打分。Motif [4] 把语言模型对 NetHack 事件描述的偏好蒸馏成奖励网络，我们的 `kev_distill` 条件与之类似。我们的评判模型读的是遥测数据的文字摘要而非像素。这样把“评判”与“感知”分开，代价是摘要需要人来设计。

LLM 评判。 用语言模型给输出打分 [6, 18] 已是评测中的常规做法。作为奖励模型，它们已知可被利用 [2, 12]。生成式验证器 [17] 读取评判模型回答“是/否”的概率，我们的 AR 评判也是如此。Kev-4B 的不同之处在于：它的答案是专用输出头给出的分布，经过温度校准，并且多个问题在一次前向传播中算完。

深度强化学习的评估。 每个任务只有少数种子时，平均回报的点估计并不可靠。我们依照 Agarwal et al. [1]：对全部 run 计算四分位间均值 (IQM)，给出分层 bootstrap 置信区间，以及改进概率。各任务的回报也一并报告。

3 评判模型

Kev-4B。 Kev-4B [8] 是一种“系统一”决策模型：在 Qwen3.5-4B-Base [9]（一种混合架构模型，各层交替使用 Gated DeltaNet [15] 和全注意力）上加一个 LoRA 适配器（秩 16）和一个指针头。一个请求包含一个状态（JSON 对象，由 `kev.api` 渲染成文本）和一个或多个问题。`score` 型问题列出有序的评判等级；`noul` 型问题要求回答是或否。Kev-4B 不生成文本。每个问题作为一个分支

接在渲染后的状态之后，指针头读取决策位置和各答案选项位置上的隐藏状态，再经过温度（存于检查点中， $T = 2.406$ ）的 softmax 得到选项上的分布。同一请求的所有问题在一次前向传播中完成：共享的状态只算一次，各问题的短分支并行计算。我们原样使用 kev 代码中的模式定义、渲染、编码器和输出头（修订版 6cfce5c），我们的封装只改变行的批处理方式。在 300 个问题，它的 bf16 概率与 kev 参考推理路径的差异最大为 0.040（argmax 一致率 99.7%），fp32 下最大为 0.0009（experiments/validation/kev_parity.json）。

自回归（AR）评判模型。 基线评判模型是 Qwen3.5-4B [9]（修订版 851bf6e），即同一家族、同样大小的指令微调模型，关闭思考模式。它在对话提示中得到同样的渲染状态和同样的问题文本，后接编号的选项，并被要求只回答选项编号（或是/否）。我们读取一次下一词元分布，限制到合法答案对应的词元上并重新归一化。归一化之前合法答案上的概率质量会被记录下来 (§6)：对是/否问题它至少为 1.00，但对五级 progress 问题只有 0.75 到 0.96，所以在那里重新归一化并非可有可无。AR 评判与 Kev-4B 一样，每个问题只需一次前向传播；不同的是，它每个问题需要单独一行，不能共享状态，而且没有做任何校准。

问题。 每个任务的状态都是 {任务描述, 参考信息, 片段摘要}（表 1）；问题固定在 experiments/part1_control/prompts/judge_prompts.json 中，在网格实验开始前即已冻结。progress 是一个五级 score 问题（failing、poor、fair、good、excellent，每级附一行说明），读数为期望等级除以 4，取值于 [0,1]。healthy 是一个 noul 问题（“机器人是否在以受控、直立的步态真正向前推进……？”），读数为 $p(\text{yes})$ 。

4 方法

评判模型看到什么。 评判模型从不看到环境奖励。它看到的是一个回合中连续若干步构成的窗口（32 步；Reacher 为 25 步）的简短摘要，由从仿真器读出的物理量计算得到：躯干的前进速度（窗口内的均值、最小值和最大值），Hopper 和 Walker2d 的躯干高度与倾角，HalfCheetah 的躯干俯仰角，Reacher 的指尖到目标的距离（窗口开始和结束时的值及其均值），动作平方的均值（“控制力度”），以及对 Hopper 和 Walker2d，回合是否在该窗口内终止（fell_over）。参考信息字段给出每个任务的两条事实，例如“这个机器人的强策略以约 3m/s 的速度向前跳”。窗口在回合结束处和每次 PPO rollout 结束处截断，所以窗口可能短于 32 步。

这是一个有意对语言评判模型有利的设定：摘要由仿真器的特权状态构造，点明了重要的物理量，参考信息还给出了目标速度。实验检验的是，一个 4B 评判模型能否把任务描述和数值化的行为摘要变成 PPO 可以优化的奖励；它不检验从像素或原始观测中进行感知的能力。

奖励条件。 窗口中的每一步都得到该窗口的分数。各条件如下：

- env: 环境自身的逐步奖励，即工程化的参照。

task	task description	reference	healthy question
Hopper	A one-legged hopping robot (MuJoCo Hopper) must hop forward along the x-axis as fast as possible without falling over.	When the robot stands upright its torso is about 1.25 m above the ground. A strong policy for this robot hops forward at about 3 m/s.	Is the robot making genuine forward progress with a controlled, upright gait (not fallen, not falling, not standing still)?
Walker2d	A two-legged walking robot (MuJoCo Walker2d) must walk or run forward along the x-axis as fast as possible without falling over.	When the robot stands upright its torso is about 1.25 m above the ground. A strong policy for this robot runs forward at about 4 m/s.	Is the robot making genuine forward progress with a controlled, upright gait (not fallen, not falling, not standing still)?
HalfCheetah	A planar cheetah-like robot with two legs (MuJoCo HalfCheetah) must run forward along the x-axis as fast as possible.	The robot cannot fall, but it can flip onto its back, which is undesirable. A strong policy for this robot runs forward at about 10 m/s.	Is the robot running forward with an upright, coordinated gait (not flipped onto its back, not stuck in place)?
Swimmer	A three-segment swimming robot in a viscous fluid (MuJoCo Swimmer) must swim forward along the x-axis as fast as possible.	Motion in the fluid is slow: a strong policy for this robot swims forward at about 0.3 m/s.	Is the robot swimming forward with a coordinated undulating motion (not drifting backward, not stuck)?
Reacher	A two-joint robot arm (MuJoCo Reacher) must move its fingertip to a target point and keep it there, using as little control effort as possible.	The arm is about 0.21 m long. A fingertip within about 0.01 m of the target is on target; control effort ranges from 0 (none) to 1 (maximum).	Is the arm bringing its fingertip to the target or holding it there (not moving away, not flailing)?

表 1: 两个评判模型读到的各任务文本 (评判模型读的是英文原文)。每个请求还包含该窗口的摘要 (§4 所列各物理量)。**progress** 问题 (“How well is the robot accomplishing its task in this segment?”) 有五个随任务而定的等级, 从 *failing* 到 *excellent*; 全部内容见 `experiments/part1_control/prompts/judge_prompts.json`。

- **env_window**: 把环境奖励在每个窗口内平均, 再赋给窗口中的每一步。这个对照具有评判模型的时间分辨率和真实奖励的信息, 用来区分 “按窗口给反馈” 的影响和 “评判模型” 的影响。
- **kev_score**: Kev-4B 的 **progress** 分数。
- **kev_gated**: Kev-4B 的 **progress** 分数乘以它的 $p(\text{healthy})$ 。这个乘法门控本应压低那些看似很快、实则并非真正推进的窗口。
- **kev_distill**: 一个小型奖励网络 (两个 64 单元的隐藏层), 用置信度加权的平方损失在线训练, 标签是每次 rollout 中随机 10% 窗口 (第一次 rollout 为全部窗口) 的 **kev_gated** 分数。它根据摘要所报告的同一组物理量的数值特征, 预测其余窗口的奖励。每个标签的权重是 Kev-4B 的 **progress** 分布的置信度, 截断到 $[0.05, 1]$ 。这个条件调用评判模型的次数约少十倍。
- **ar_gated**: AR 评判的期望 **progress** 等级乘以它的 $p(\text{healthy})$, 即 **kev_gated** 的 AR 对应版本。

- **ar_sampled**: 与 **ar_gated** 相同, 但 **progress** 等级从 AR 评判的分布中采样, 而非取期望。这是 AR 评判以温度 1 用文字作答时给出的奖励 (3 个种子)。

所有条件都经过相同的奖励归一化 (逐步复现 gymnasium 的 `NormalizeReward`, 并截断到 $[-10, 10]$)。评判模型是冻结的, 没有任何条件让评判模型适应策略。

PPO。 我们使用 CleanRL 的 `ppo_continuous_action.py` [3, 11] 的单环境移植版, 采用其默认超参数: 10^6 个环境步, 每次 rollout 2048 步, 10 个 epoch、每个 epoch 32 个小批量, 学习率 $3 \cdot 10^{-4}$ 线性退火到零, $\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$, 裁剪系数 0.2, 价值损失系数 0.5, 无熵奖励, 梯度范数裁剪到 0.5, 并对观测做滑动归一化。所有条件的超参数相同, 没有为任何条件调参。

评估。 评估时策略采用确定性动作 (高斯分布的均值), 冻结观测归一化的统计量, 使用环境自身的无折扣回报: 每 10^5 步评估 10 个回合, 结束时评估 20 个回合, 评估种子固定 (第 k 个回合分别为 $10,000 + k$ 和 $20,000 + k$)。评判模型从不看到这些回报。我们把一个 run 的最终回报 R 按任务归一化为 $(R - R_{\text{rand}})/(R_{\text{env}} - R_{\text{rand}})$, 其中 R_{rand} 是均匀随机策略的平均回报, R_{env} 是 **env** 条件的平均最终回报。分数为 1 表示 “与在工程化奖励上训练的 PPO 一样好”。对 5 个任务 $\times$ 5 个种子的汇总依照 Agarwal et al. [1]: 四分位间均值 (IQM) 及其 95% 分层 bootstrap 置信区间 (10000 次重采样), 以及一个条件相对另一个条件的改进概率。

评判一致性。 每个 run 随机记录 2% 的窗口, 包括摘要和该窗口的环境奖励。**env** 条件的 run (全部种子、整个训练过程) 记录下的窗口覆盖了参照学习者失败、平庸和良好的各种行为。离线时, 两个评判模型对每个任务最多 4000 个这样的窗口打分, 所用服务器和提示与训练时相同。我们报告每种信号与窗口逐步环境奖励之间的 Spearman 相关系数, 以及它与任务所关心的物理量之间的相关系数: 平均前进速度, Reacher 则是窗口内指尖到目标距离的减少量。对 Hopper 和 Walker2d, 还报告 $p(\text{healthy})$ 识别 “未以摔倒结束的窗口” 的 AUROC。训练期间, 每个以评判为奖励的 run 还会在每次 rollout 记录: 它所训练的窗口分数与同一批窗口的环境奖励之间的相关系数。

5 实验设置

任务为 Hopper、Walker2d、HalfCheetah、Swimmer 和 Reacher, 均为 Gymnasium 1.3 下的 MuJoCo v5 版本 [13, 14]。每个任务和条件有五个种子 (**ar_sampled** 为三个)。种子 s 决定策略初始化、环境的种子以及该 run 自身的随机选择, 包括为记录和蒸馏抽取的窗口。调用评判模型的 run 仍不能做到逐比特可复现: 评判模型的浮点结果会因服务器如何把该请求与其他 run 的请求合批而略有不同, 而且 **ar_sampled** 抽取的等级来自所有 run 共享的服务器随机数生成器。所有 run 都在同一台 Colab 虚拟机上完成, 配有一块 NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB) 和 48 个 vCPU。PPO 在 CPU 上运行, 每个 run 一个进程。两个评判模型在同一个 GPU 服务器进程中运行, 其中同时装载 Kev-4B (bf16, LoRA 已合并) 和 AR 评判模型。服务器把所有 run 的并发请求合并成批 (每批最多 4096 个窗口, 聚合等待 15 ms)。同时训练的 run 最

task	windows	Kev-4B			AR judge			
		progress	healthy	gated	progress	sampled	healthy	gated
Hopper	3233	0.73	0.56	0.70	0.66	0.22	0.50	0.65
Walker2d	3270	0.73	0.58	0.71	0.69	0.22	0.58	0.68
HalfCheetah	3167	0.93	0.85	0.88	0.64	0.02	0.76	0.79
Swimmer	3167	0.73	0.83	0.86	0.67	0.26	0.78	0.76
Reacher	3997	0.37	0.11	0.25	0.09	0.04	-0.05	0.02

表 2: 在 `env` 条件的 `run` 记录的窗口上（全部种子、整个训练过程），各评判信号与窗口逐步环境奖励的 Spearman 相关系数。*progress*: 期望等级除以 4; *healthy*: $p(\text{yes})$; *gated*: 两者之积; *sampled*: 从 AR 评判的分布中抽取的一个等级，即以温度 1 用文字作答的 AR 评判会给出的结果。

多 24 个，因此一个 `run` 的总耗时取决于有多少其他 `run` 共用评判服务器，我们按实测报告。软件版本固定在仓库中 (`scripts/setup_colab.sh`, 附两个环境的完整包列表)。因运行时重置而中断的 `run` 从最近一次定期检查点恢复。仿真器状态不在检查点中，因此正在进行的回合会被丢弃并开始一个新回合。日志记录每一次恢复，恢复过的 `run` 数会随结果一同报告。

6 评判分数是否跟得上奖励？

两个评判模型对 `env` 条件的 `run` 记录下的窗口打分：Hopper 3233 个，Walker2d 3270 个，HalfCheetah 3167 个，Swimmer 3167 个，Reacher 3997 个，来自全部五个 `run` 和整个训练过程。表 2 给出每种信号与窗口逐步环境奖励的 Spearman 相关系数。区间为两阶段 bootstrap 的 95% 区间：先对 `run` 重采样，再对每个 `run` 的窗口重采样，因此同一个 `run` 的窗口不被当作相互独立。

Kev-4B 比 AR 评判更紧密地跟随奖励。 在每个任务上，Kev-4B 的 *progress* 与环境奖励的相关性都强于 AR 评判的期望 *progress*。差值在 Walker2d 上最小 (+0.04 [0.03, 0.06])，在 HalfCheetah (+0.29 [0.18, 0.41]) 和 Reacher (+0.28 [0.25, 0.31]) 上最大，没有一个区间包含零。门控信号在每个任务上也保持同样的顺序（从 Walker2d 的 +0.02 到 Reacher 的 +0.23）。

Reacher 是需要限定这一结论的例外。 Reacher 的逐步奖励主要是指尖到目标的距离，这是一个状态；而 *progress* 的各等级描述的是接近目标并保持在目标上。以窗口内距离的减少量——问题主要关心的量——为准，两个评判模型相差不大，AR 评判略微领先 (Kev-4B 0.61, AR 0.64; 差值 -0.03 [-0.06, -0.00])。因此，Kev-4B 与 Reacher 奖励更高的相关性，并不是因为它更好地读懂了问题所问的“接近”。在四个运动任务上，奖励与前进速度对窗口的排序几乎相同，两种相关系数也几乎一致。

对 AR 评判的答案采样会丢掉大部分信号。 从 AR 评判的分布中抽取的等级与奖励的相关系数只有 0.02 (HalfCheetah) 到 0.26 (Swimmer)，而同一分布的期望等级为 0.09 到 0.69。在单个窗

task	env	env_window	kev_score	kev_gated	kev_distill	ar_gated	ar_sampled
Hopper	2820 $\pm$ 969	3419 $\pm$ 450	3061 $\pm$ 314	3169 $\pm$ 174	2833 $\pm$ 909	1165 $\pm$ 309	1127 $\pm$ 293
Walker2d	3150 $\pm$ 1276	4068 $\pm$ 1909	2593 $\pm$ 509	2536 $\pm$ 712	2943 $\pm$ 562	1321 $\pm$ 189	1330 $\pm$ 165
HalfCheetah	3441 $\pm$ 1152	2376 $\pm$ 1349	2845 $\pm$ 638	4241 $\pm$ 242	4117 $\pm$ 460	1069 $\pm$ 1046	1050 $\pm$ 824
Swimmer	81 $\pm$ 22	278 $\pm$ 104	99 $\pm$ 148	141 $\pm$ 124	91 $\pm$ 41	141 $\pm$ 120	118 $\pm$ 166
Reacher	-6.4 $\pm$ 1.6	-28.0 $\pm$ 9.4	-48.9 $\pm$ 21.2	-75.8 $\pm$ 10.9	-88.4 $\pm$ 21.9	-75.2 $\pm$ 21.3	-68.8 $\pm$ 21.3

表 3: 各任务的最终回报：训练 10^6 步后，确定性策略在 20 个评估回合上的平均回报，再对种子取均值 $\pm$ 标准差。均匀随机策略的回报为 13 (Hopper)、-0.40 (Walker2d)、-280 (HalfCheetah)、-0.13 (Swimmer) 和 -42 (Reacher)。

口的尺度上，信息在分布之中，从中抽取一次的结果大部分是噪声。

摔倒能被看出；门控的效果有好有坏。 $p(\text{healthy})$ 几乎完美地区分了未摔倒和摔倒的窗口：Kev-4B 在 Hopper 和 Walker2d 上的 AUROC 为 1.00 和 0.99，AR 评判为 1.00 和 1.00。摘要本身就写明了机器人是否摔倒，所以这检验的是阅读而非判断。把 Kev-4B 的 `progress` 乘以其 $p(\text{healthy})$ ，在四个任务上降低了与奖励的一致性，只在 Swimmer 上提高了它 (0.86 对 0.73)。对 AR 评判，门控在 HalfCheetah 上也有帮助 (0.79 对 0.64)。

AR 评判把概率质量放在答案选项之外。 重新归一化之前，AR 评判的下一词元分布放在五个 `progress` 选项上的质量只有 0.75 到 0.96 (随任务而异)，放在是/否上的至少为 1.00。Kev-4B 没有这种损失，因为它的输出头只对选项打分。

这说明了什么，没有说明什么。 这些窗口来自优化真实奖励的策略，它们表明每个评判模型如何给这类策略产生的行为排序；它们不能表明，当策略优化的是评判模型本身时，评判模型会如何给其行为排序。后者由训练实验来测量 (§7)。

7 用评判模型的奖励训练

全部 165 个 run 都已完成：五个任务上的七种条件，每种 5 个种子 (`ar_sampled` 为 3 个)。虚拟机的一次重置中断了 24 个 run (`kev_score`, `kev_gated`, `kev_distill`, `ar_gated`)，它们从第 301,056 步的检查点恢复，每个丢弃了当时进行中的回合，最多 703 步。表 3 给出各任务的最终回报，表 4 给出归一化后的汇总结果，图 1 给出学习曲线。

从汇总看，Kev-4B 的门控奖励和蒸馏奖励接近工程化奖励，AR 评判的奖励则差得远。归一化分数的 IQM: `kev_gated` 为 1.00 [0.81,1.09], `kev_distill` 为 0.98 [0.82,1.10], `env` 为 1.05 [0.91,1.16], 区间大幅重叠。在同一任务上，用这两种奖励训练的 run 胜过用工程化奖励训练的 run 的概率分别为 0.44 (`kev_gated`) 和 0.44 (`kev_distill`)。去掉门控后，Kev-4B 的分数表现更差 (0.78 [0.60,0.91], 概率 0.28)。AR 评判的奖励只达到工程化奖励归一化分数的三分之一左右：期望

condition	runs	IQM [95% CI]	mean	median	$P(> \text{env})$	$P(> \text{env_window})$
env	25	1.05 [0.91, 1.16]	1.00	1.03	0.50	0.46
env_window	25	1.10 [0.89, 1.26]	1.41	1.21	0.54	0.50
kev_score	25	0.78 [0.60, 0.91]	0.76	0.82	0.28	0.27
kev_gated	25	1.00 [0.81, 1.09]	0.79	1.05	0.44	0.31
kev_distill	25	0.98 [0.82, 1.10]	0.59	1.02	0.44	0.27
ar_gated	25	0.35 [0.28, 0.46]	0.40	0.36	0.18	0.10
ar_sampled	15	0.31 [0.19, 0.45]	0.37	0.36	0.12	0.11

表 4: 全部任务和种子上的归一化分数 $(R - R_{\text{rand}})/(R_{\text{env}} - R_{\text{rand}})$: 四分位均值 (IQM) 及其 95% 分层 bootstrap 区间、均值、中位数, 以及该条件的一个 run 在同一任务上胜过 env 或 env_window 的一个 run 的概率 (对任务取平均)。

读出为 0.35 [0.28, 0.46], 采样读出为 0.31 [0.19, 0.45]。在同一任务上, 一个 kev_gated run 胜过一个 ar_gated run 的概率为 0.78; 分任务看, Hopper 上为 1.00, HalfCheetah 上为 1.00, Walker2d 上为 0.88, 而 Swimmer (0.52) 和 Reacher (0.48) 上接近持平。

汇总掩盖了 Reacher。 IQM 去掉了最低和最高各四分之一的 run。均值更低: kev_gated 为 0.79, kev_distill 为 0.59。在 Reacher 上, 每种评判奖励的最终结果都低于随机策略, 因此这些 run 的归一化分数为负 (表 3)。所以我们逐个任务来看。

逐个任务。 在 Hopper 上, Kev-4B 的奖励最终接近工程化奖励 (kev_gated 为 3169, env 为 2820; 改进概率 0.44)。在 Walker2d 上它们更低 (kev_gated 为 2536, kev_distill 为 2943, env 为 3150; 概率分别为 0.32 和 0.36)。在 HalfCheetah 上, kev_gated (4241) 和 kev_distill (4117) 高于工程化奖励 (3441; 两者的概率都是 0.72), 原因见 §7.1。在 Swimmer 上, 用工程化奖励训练的 PPO 几乎没让游动者动起来 (81); 其他每种奖励的平均结果都更好, 其中窗口平均的真实奖励最好 (278), 评判奖励在不同种子间差异很大。在 Reacher 上, 只有工程化奖励学会了任务: 用它为 -6.4 , 用它的窗口平均为 -28.0 , 用评判奖励为 -48.9 到 -88.4 。学习曲线显示, 评判奖励训练的 Reacher 策略随着训练越来越差 (图 1)。

窗口级反馈在运动任务上代价很小, 在 Reacher 上代价很大。 在评判窗口上平均的真实奖励 (env_window) 在汇总上至少不比逐步奖励差: IQM 1.10 [0.89, 1.26], 改进概率 0.54。它在 Hopper (0.80)、Walker2d (0.80) 和 Swimmer (0.92) 上领先; 在 HalfCheetah 上落后 (0.20), 它的五个 run 中有 3 个最终翻成背朝下 (§7.1); 在 Reacher 上也落后 (0.00)。一个 Reacher 回合只有两个 25 步的窗口, 平均后的奖励不再告诉策略是哪些动作让指尖靠近了目标, 其机械臂最终离目标并不比随机策略更近 (表 5)。因此, 评判模型的时间分辨率能解释它们在 Reacher 上的部分失败, 但完全解释不了 AR 评判在 Hopper 和 Walker2d 上的失败——在那里窗口平均奖励表现很好。

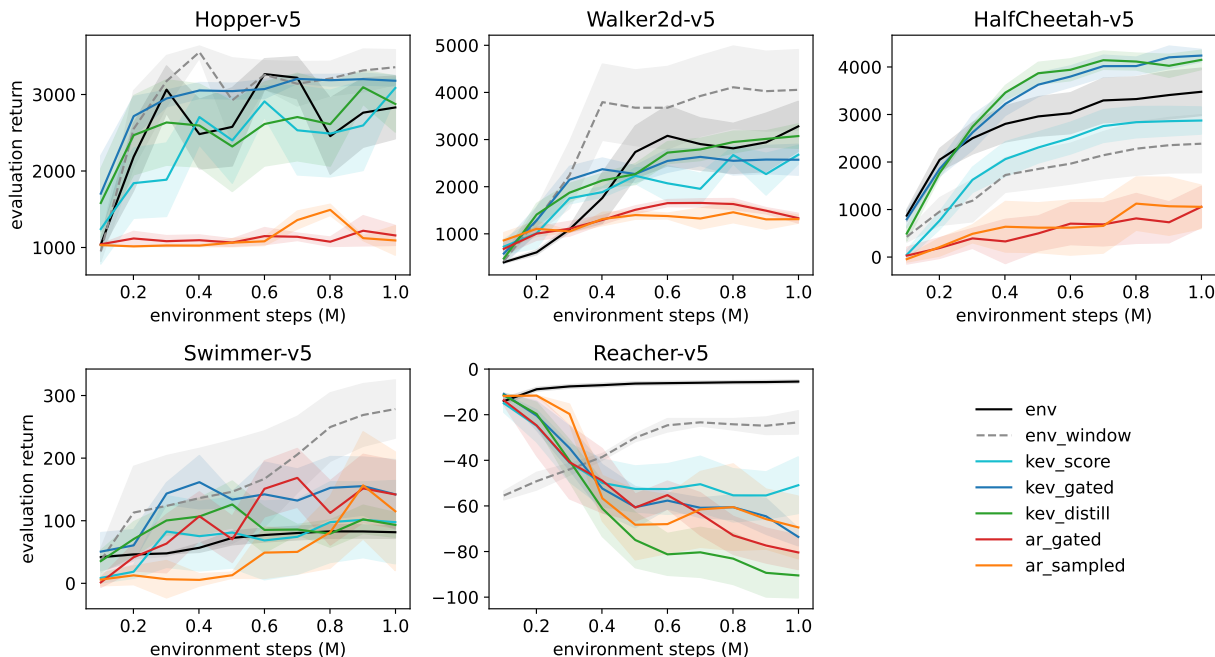


图 1: 训练过程中的评估回报: 每 10^5 步评估 10 个回合, 曲线为种子均值, 阴影为 $\pm$ 一个标准误。

蒸馏只用十分之一的评判调用就保住了 Kev-4B 的奖励。 `kev_distill` 每个 run 只向评判模型询问 3.5k 个窗口, 而不是 33.9k 个, 最终与 `kev_gated` 持平: IQM 0.98 对 1.00, 改进概率 0.45。它最弱的任务是 Reacher (-88.4)。

训练中, 两个评判模型的分数与奖励的相关程度相近。 每个 run 对每次 rollout 记录了它所训练的窗口分数与同一批窗口的环境奖励之间的相关系数。对 rollout 和 run 取平均, `kev_gated` 为 0.56, `ar_gated` 为 0.52, 而 `ar_sampled` 只有 0.17——它的单次抽样大部分是噪声 (§6)。两种门控奖励与环境奖励的相关程度差不多, 却训练出了截然不同的策略, 下一节会说明这一点。

7.1 策略学到了什么

回报并不说明策略做了什么。我们把每个最终策略在它的 20 个评估回合上重新运行一遍, 记录机器人的实际行为 (表 5)。重新运行的回报与日志中的回报完全一致 (最大差异 0)。

AR 评判的奖励让单腿跳跃机器人和双足行走机器人学会了站着。 最终的 `ar_gated` 策略在 Hopper 上有 49% 的步数站着不动, 在 Walker2d 上为 40%。`kev_gated` 的对应数字是 1% 和 10%, `env` 为 0% 和 1%。它们的平均前进速度为 0.17 和 0.37 m/s, 而 `kev_gated` 为 2.21 和 1.57 m/s, `env` 为 2.59 和 3.02 m/s。它们在 Hopper 上的回报 (1165) 接近 1000, 也就是在一个 1000 步的回合里一直站着、仅靠存活奖励就能拿到的分数。对等级采样并不能消除这一现象: `ar_sampled` 在两个任务上站着不动的步数占比为 26% 和 17%。

	Hopper	Walker2d	Swimmer	HalfCheetah	Reacher	
condition	still	still	still	runs flipped	effort	distance (m)
env	0%	1%	62%	1 of 5	0.01	0.101
env_window	0%	0%	14%	3 of 5	0.19	0.180
kev_score	0%	4%	11%	0 of 5	0.36	0.256
kev_gated	1%	10%	15%	0 of 5	0.63	0.263
kev_distill	1%	1%	11%	0 of 5	0.76	0.250
ar_gated	49%	40%	18%	0 of 5	0.63	0.253
ar_sampled	26%	17%	50%	0 of 3	0.56	0.252
random policy	31%	4%	26%	–	0.33	0.180

表 5: 最终策略的行为, 对回合和种子取平均 (每个 run 20 个评估回合)。still: $|v_x| < 0.1 \text{ m/s}$ 的步数占比。runs flipped: 猎豹有一半以上步数背朝下 ($\cos(\text{pitch}) < 0$) 的 run 数。effort: 动作平方的均值, 取值于 $[0, 1]$ 。distance: 一个回合中指尖到目标的平均距离。随机策略从均匀分布中采样动作。

task	windows (slow / fast)	env. reward	Kev-4B		AR judge	
			progress	gated	progress	gated
Hopper	42 / 430	0.31	0.59	0.48	0.89	0.87
Walker2d	386 / 1238	0.27	0.62	0.47	0.90	0.87
HalfCheetah	309 / 1368	−0.06	0.51	0.15	0.85	0.65

表 6: 在 §6 的窗口 (env 条件的 run 记录的窗口) 上, 每种信号给“站着”相对于“跑”的报酬。表中数值为: 未摔倒且慢于 0.5 m/s 的窗口上的均值, 除以快于 3 m/s 的窗口上的均值。env. reward: 逐步环境奖励。

表 6 说明了原因。它比较每种信号给一个慢于 0.5 m/s 的直立窗口和一个快于 3 m/s 的窗口各付多少。AR 评判的门控分数在 Hopper 上给慢窗口的报酬是快窗口的 0.87 (0.85 对 0.97), 在 Walker2d 上是 0.87。工程化奖励只付 0.31 和 0.27, Kev-4B 的门控分数付 0.48 和 0.47。回合中的每一步都得到其所在窗口的分数, 而摔倒会结束回合。因此在 AR 评判的奖励下, 直立站着每一步挣得几乎和跳跃一样多, 又没有摔倒的风险。PPO 很早就找到了这一点: Hopper 上 ar_gated 的曲线从第一次评估起就是平的 (图 1)。然而 AR 评判的分数对窗口的排序几乎和 Kev-4B 一样好: 它们与奖励的 Spearman 相关系数在 Hopper 上为 0.65 对 0.70, 在 Walker2d 上为 0.68 对 0.71 (表 2)。不同的是差距: AR 评判几乎把每个直立窗口都打在其分数范围的顶端。与奖励的排序一致性并没有预示出 PPO 能用上哪种奖励。

评判模型的奖励让猎豹保持直立。 在 HalfCheetah 上, env 的五个 run 中有 1 个、env_window 的五个 run 中有 3 个最终让猎豹背朝下。这是一个工程化奖励不加惩罚的局部最优。这些 run 的回报最多为 1599, 而同样两种条件下保持直立的 run 至少为 3049。用评判奖励训练的 23 个 run 中没有一个是翻倒。两个评判模型读到的任务描述都写明翻倒是不希望出现的, 最低的 progress 等

condition	judged windows	judge wait (h)	wall-clock (h)	corr. (score, env reward)
env	0	0.0	0.1	—
env_window	0	0.0	0.1	—
kev_score	33,914	2.0	2.1	0.57
kev_gated	33,942	1.9	2.1	0.56
kev_distill	3,477	1.9	2.0	0.54
ar_gated	33,861	1.9	2.0	0.52
ar_sampled	33,869	1.9	2.0	0.17

表 7: 每个 run 的代价，对任务和种子取平均：发送给评判模型的窗口数、等待其回答的时间、run 的总耗时，以及每次 rollout 中 run 所训练的窗口分数与同一批窗口的环境奖励之间相关系数的均值。一个 GPU 上的一个评判服务器同时服务最多 24 个 run，因此等待时间包含了它处理其他 run 请求的时间。

级也点了它的名。去掉翻倒的那个 run 后，env 在 HalfCheetah 上的平均回报为 3902，kev_gated 为 4241，所以 Kev-4B 在这里的优势很大一部分来自翻倒。在这个例子里，评判模型的奖励带有工程化奖励所缺少的偏好，而策略遵循了它。

在 Reacher 上，评判模型的奖励让机械臂乱甩。用评判奖励训练的机械臂，指尖离目标的平均距离为 0.250 到 0.263 m，比随机策略 (0.180 m) 和工程化奖励 (0.101 m) 都远。它们的控制力度也高于随机策略：0.36 到 0.76，随机策略为 0.33，env 为 0.01。Reacher 的奖励每一步都按动作的平方扣分，这就是这些策略最终低于随机策略的原因。两个评判模型的分数跟随的是窗口内距离的变化 (Kev-4B 与 AR 评判的 progress 的 Spearman 相关系数分别为 0.61 和 0.64，§6)，而对 Reacher 奖励每一步都在计算的距离本身跟随得少得多（与奖励的相关系数为 0.37 和 0.09）。每个回合只有两个窗口，这两种信号都没能把 PPO 引到目标上。

8 代价

表 7 给出评判模型在每个 run 上的代价。对每个窗口都询问评判模型的 run，在 10^6 步中大约发送 33.9k 个窗口，kev_distill 大约发送 3.5k 个。不用评判模型的 run 耗时 0.1 h；用评判模型的 run 耗时约 2.1 h，约为前者的 17 倍，其中几乎全部 (1.9 h) 花在等待评判上。这些时间反映的是我们的设置：一个 GPU 上的一个评判服务器同时回答最多 24 个 run 的请求，一个 run 的等待时间包含了服务器处理其他 run 请求的时间。这也是为什么 kev_distill 只有十分之一的窗口，等待时间却差不多 (1.9 h)：它的请求排在其他 run 的请求后面。我们没有测量评判模型只服务一个 run 的情形，而蒸馏奖励的节省会在那种情形下体现出来。

9 讨论

实验回答了什么。 问题是：一个读取机器人行为文字摘要的 4B 评判模型，能否给 PPO 提供一个可以学习的奖励。用 Kev-4B 的门控奖励或蒸馏奖励，PPO 在全部四个运动任务上都学会了向前运动：在 Hopper 上最终回报接近工程化奖励，在 HalfCheetah 上更高；在 Swimmer 上平均更高但分散很大，在 Walker2d 上更低。在 Reacher 上，每种评判奖励都失败了。同家族、同规模的自回归评判模型读的是同样的文本、回答的是同样的问题，但它的奖励让单腿跳跃机器人和双足行走机器人学会了站着。

相关性并不够。 在 Hopper 和 Walker2d 上，两种门控评判按奖励给窗口排序的能力几乎一样好，无论是离线 (§6) 还是训练中 (表 7)。然而一种奖励训练出了跳跃和行走，另一种训练出了站立。决定结果的是评判模型给任务想要的行为比给安全但无用的行为多付多少 (表 6)。把评判模型当作奖励进行离线检验时，除了相关系数，也应报告这个差距。在这里，摔倒会结束回合，而每一步都得到正的奖励，所以差距小时，存活比前进更划算；而分数挤在量表顶端的评判模型，正会让这个差距变小。

评判模型的奖励遵循的是任务描述。 它不是工程化奖励的带噪副本。HalfCheetah 的任务描述说翻倒是不希望出现的，没有任何评判奖励让猎豹翻倒，而工程化奖励在五个 run 中有 1 个翻倒。Reacher 的任务描述要求控制力度尽量小，摘要里也报告了力度，可评判模型训练出的机械臂仍然乱甩。两个评判模型更多地跟随窗口内距离的变化，而不是距离的大小；每个回合只有两个窗口，PPO 没能找到目标。评判模型可以加入工程化奖励所缺少的偏好，也可能漏掉工程化奖励所收取的代价。

蒸馏。 用 Kev-4B 在十分之一窗口上的标签训练出的奖励网络，与对每个窗口都询问 Kev-4B 的效果一样好。在我们的设置中，这一节省没有缩短 run 的时间，因为评判服务器是共享的。我们预计在评判模型只服务一个 run 时它会起作用，但没有测量。

局限。 评判模型读的是我们根据仿真器特权状态设计的摘要，摘要点明了关键的物理量并给出了目标速度。这对评判模型有利，本实验对于从像素或原始观测进行评判没有任何结论。所有条件使用同一套 PPO 配置 (CleanRL 的默认值)、 10^6 步、5 个种子 (`ar_sampled` 为 3 个)。置信区间很宽，各任务上介于约 0.3 和 0.7 之间的改进概率不应被解读为差异。AR 评判只用了一种提示格式和一种读出方式 (在选项词元上取期望并重新归一化)，没有校准，也不先推理再作答。换一种提示、校准或重新缩放读出、或先推理再作答，都可能改变它的差距；我们没有尝试，也没有为任何一个评判模型调提示。本文比较的是同一家族、同一规模的两个评判模型，并不能说明决策模型普遍比生成式评判模型更适合作为奖励。代价是在一个评判服务器被最多 24 个 run 共享的情况下测得的。最后，有 24 个 run 在机器重置后恢复，每个丢失了一个回合。

可复现性

所有 run 的代码、提示、配置、原始日志（每个训练回合、每次更新、每次评估，以及 2% 被评判窗口的样本）和最终策略都在仓库中（`experiments/part1_control/`）。表格和正文中的每个数字都由 `dcjev/analysis/part1.py`、`dcjev/analysis/judge_agreement.py`、`dcjev/analysis/behaviour.py` 和 `dcjev/analysis/paper_numbers.py` 从中生成。

参考文献

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021.
- [2] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.
- [3] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.
- [4] Martin Klissarov, Pierluca D’Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. In *International Conference on Learning Representations*, 2024.
- [5] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *International Conference on Learning Representations*, 2023.
- [6] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing*, 2023.
- [7] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.
- [8] Jared Palmer. Kev-4B: a system one decision model. <https://huggingface.co/jaredpalmer/kev-4b>, revision 6cfce5c2fa4b4bd64026336ab649c5ca78857d52, 2026.
- [9] Qwen Team. Qwen3.5-4B and Qwen3.5-4B-Base. <https://huggingface.co/Qwen/Qwen3.5-4B-Base>, 2026.

- [10] Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, 2024.
- [11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [12] Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems*, 2022.
- [13] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.
- [14] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.
- [15] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.
- [16] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning*, 2023.
- [17] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.
- [18] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.