PPO×Family

第七讲：挖掘黑科技

主办 中国计算机学会 CCF

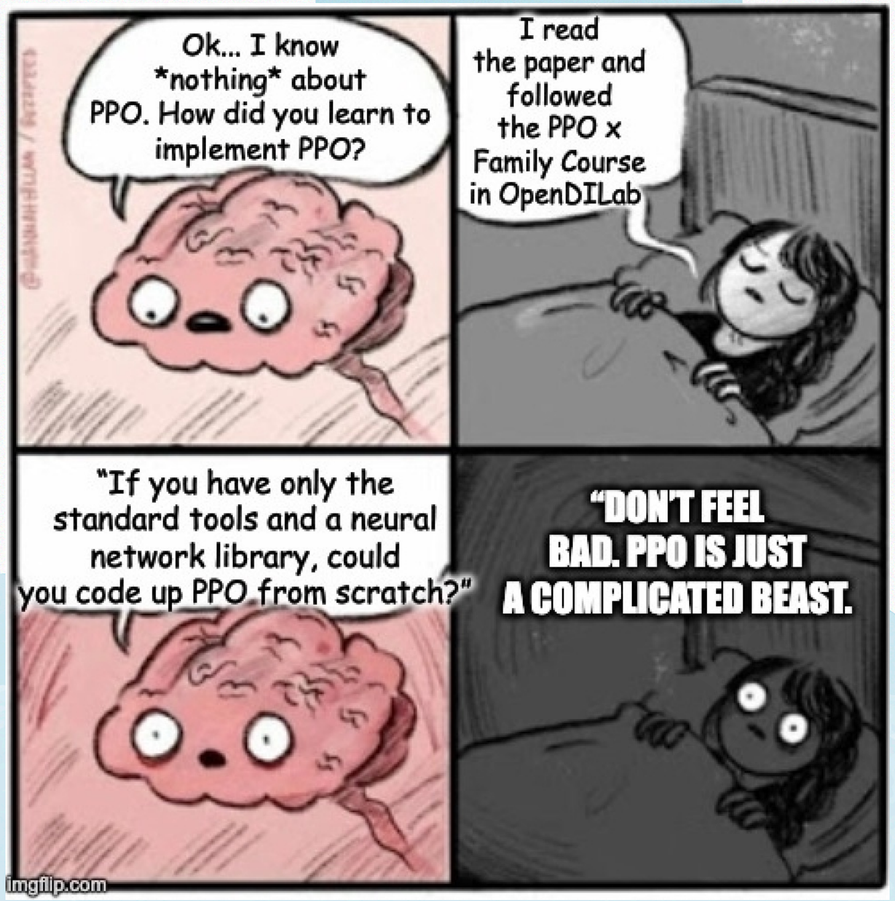
承办 上海人工智能实验室 Shanghai Artificial Intelligence Laboratory

协办 GAIAA 北京大学人工智能研究院 INSTITUTE FOR ARTIFICIAL INTELLIGENCE, PEKING UNIVERSITY 浙江大学上海高等研究院 SHANGHAI INSTITUTE FOR ADVANCED STUDY ZHEJIANG UNIVERSITY 商汤 sensetime

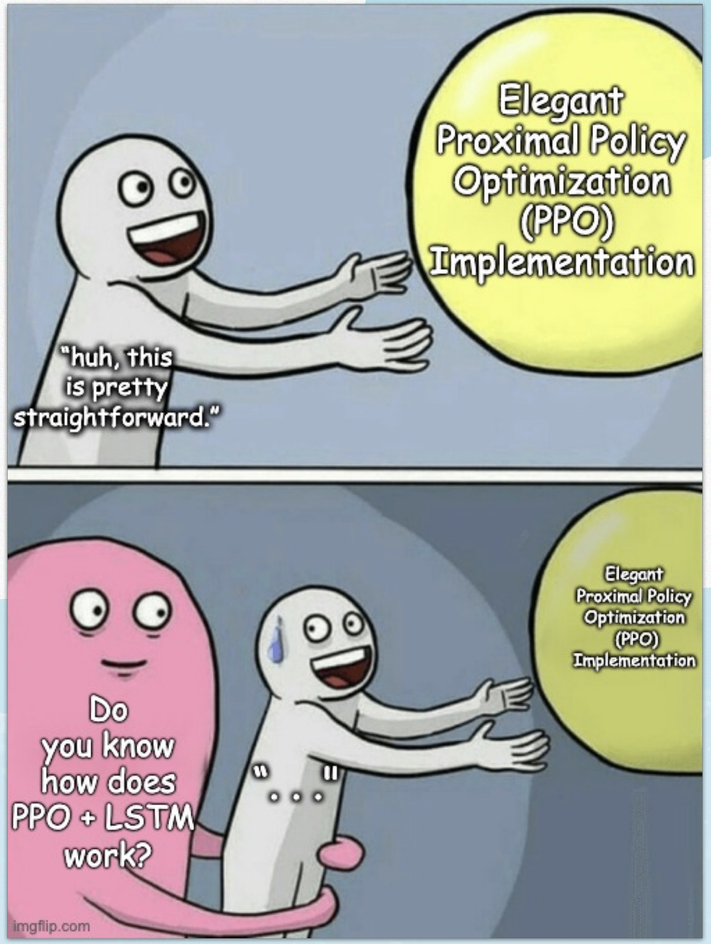
支持 知乎 机器之心 智海 新一代人工智能讲教平台 paperweekly

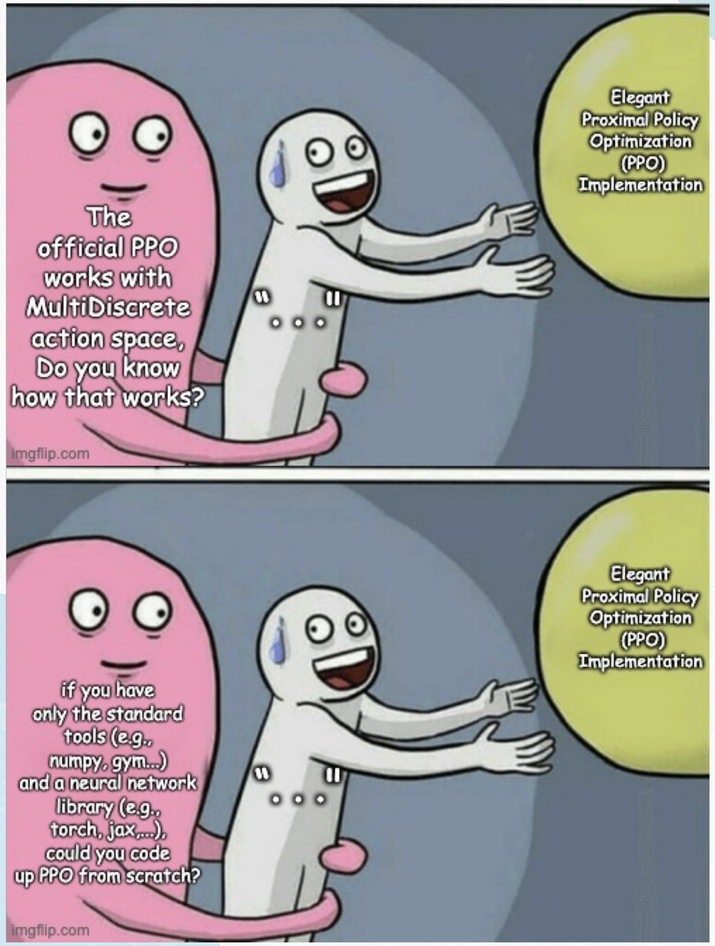
Bags of Tricks

问题：强化学习究竟是不是玄学？



问题：别人家的PPO为什么这么强？



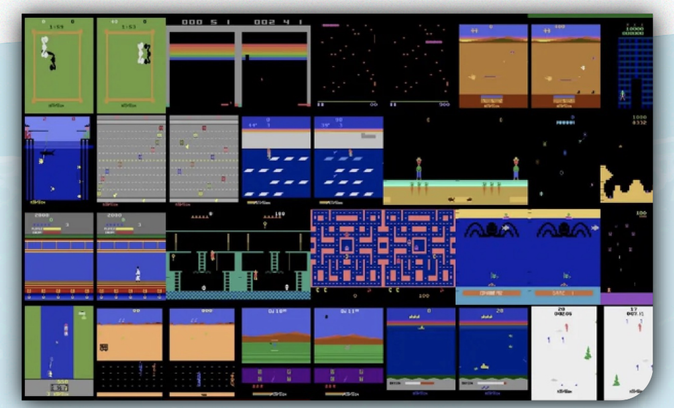


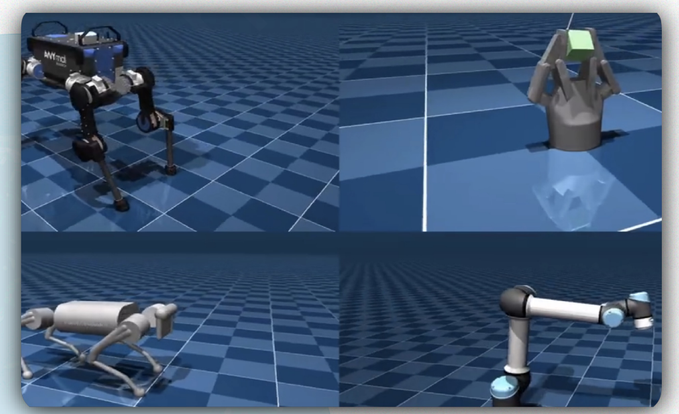
- 集成万千技巧（Tricks）
- 组合诸多领域（Areas）
- 应对大量环境（Envs）

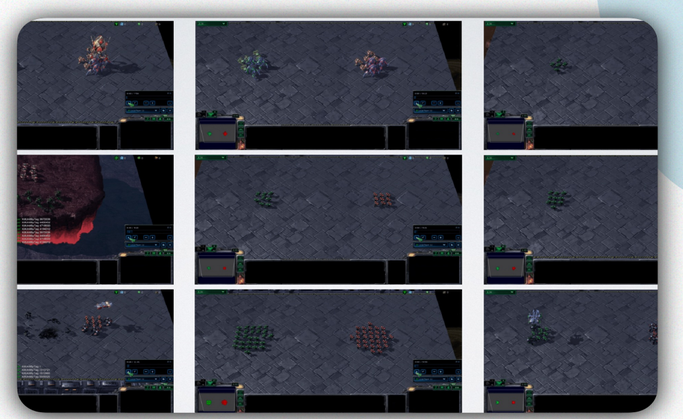
The Seven Realms of PPO

调优PPO的七重境界

- 合理运用随机：Dynamic Seed
- 亡羊补牢：Extra Clip
- 笨鸟先飞：Network Initialization
- 免死金牌：Grad Clip
- 权衡偏差与方差：GAE
- 稳扎稳打：Recompute Advantage
- 巧夺天工：Entropy Balance







一重境界：权衡偏差与方差的GAE

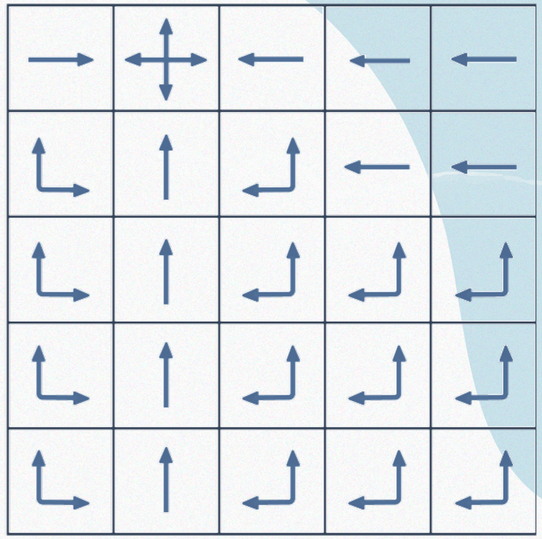
价值函数

$$V^{\pi}(s) = \mathbb{E}_{\tau \sim \pi}[G(\tau)|s_t = s]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau \sim \pi}[G(\tau)|s_t = s, a_t = a]$$

示例：格子世界（Grid World）

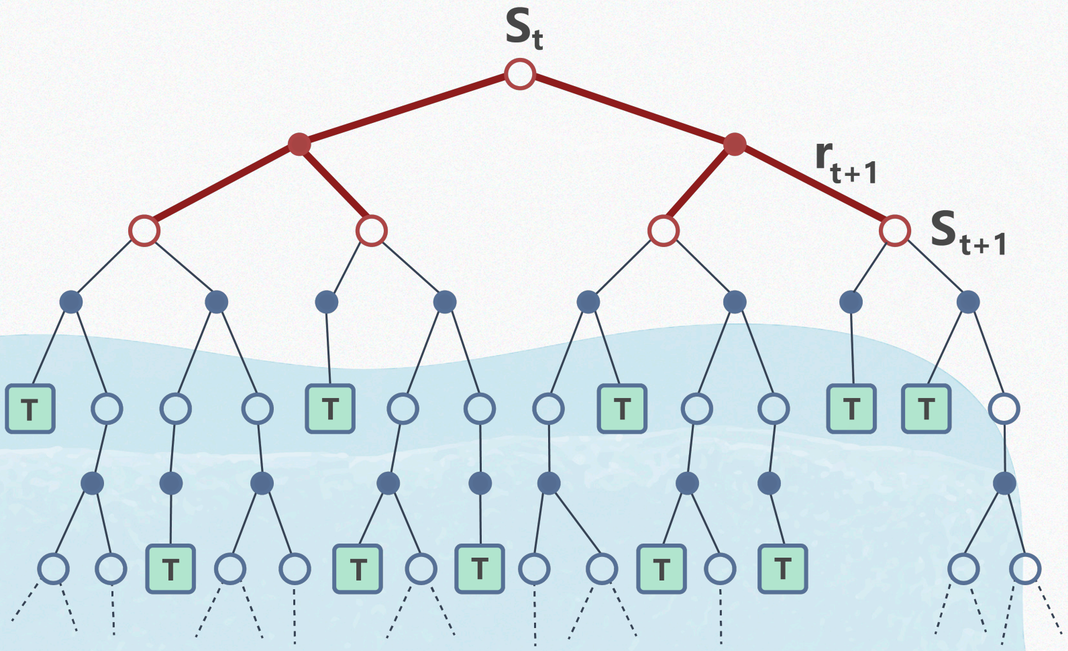
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7



一重境界：权衡偏差与方差的GAE

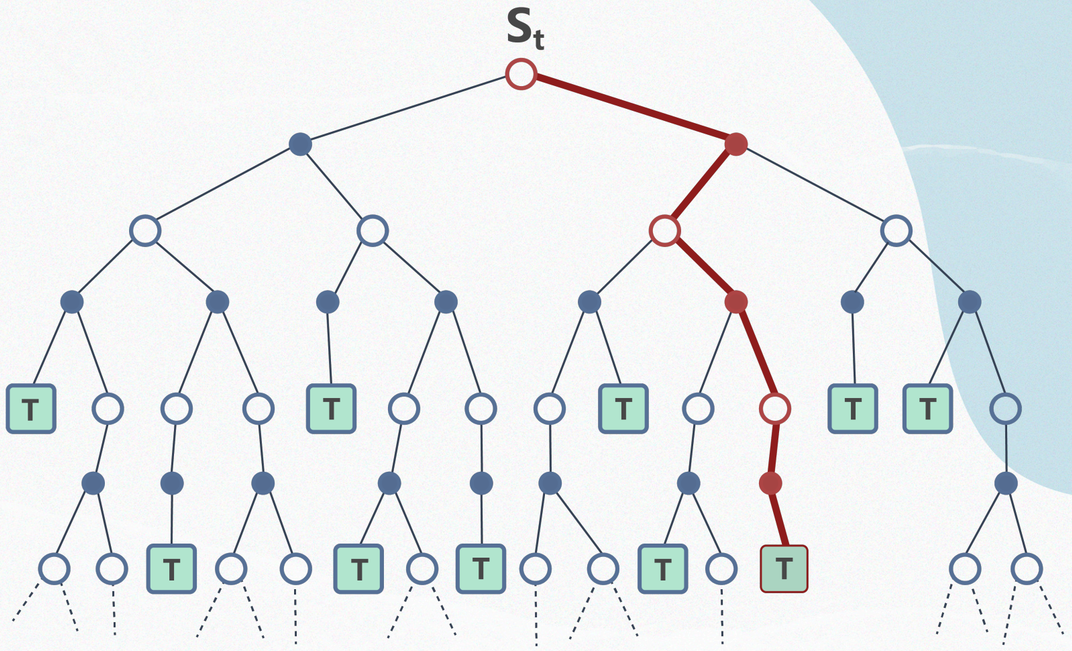
动态规划（DP）

$$V(s_t) \leftarrow \mathbb{E}_{\tau\sim\pi}[r_t + \gamma V(s_{t+1})]$$

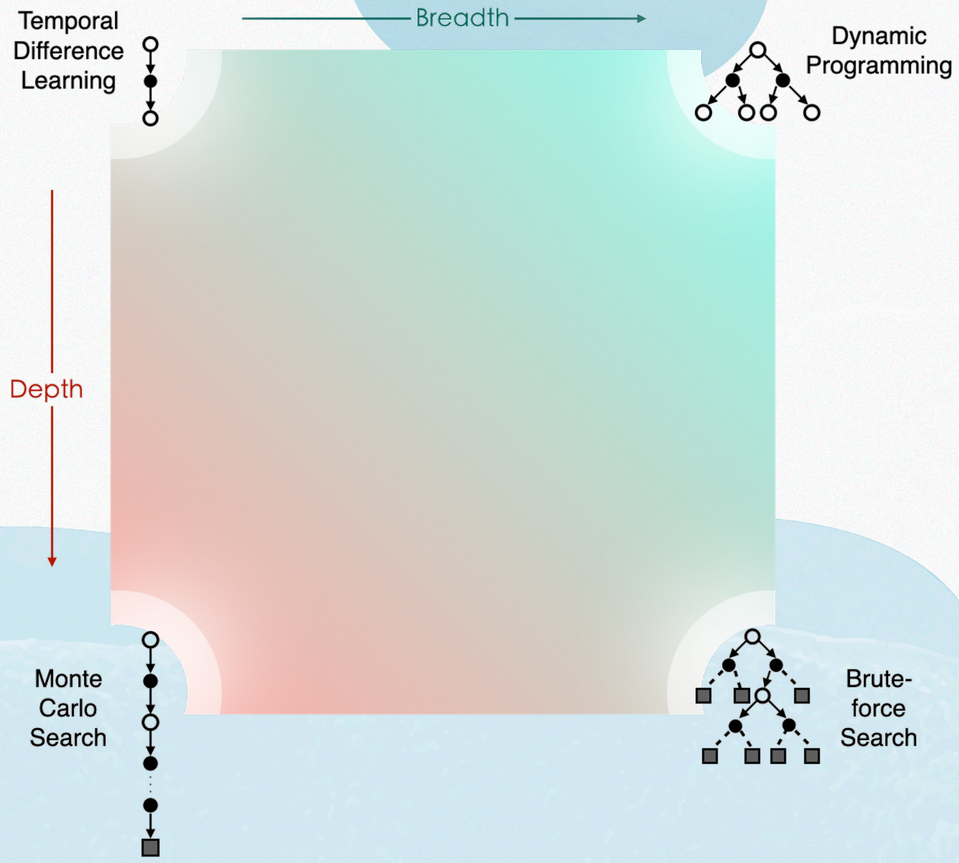


蒙特卡洛（MC）

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t(\tau) - V(s_t))$$



一重境界：权衡偏差与方差的GAE



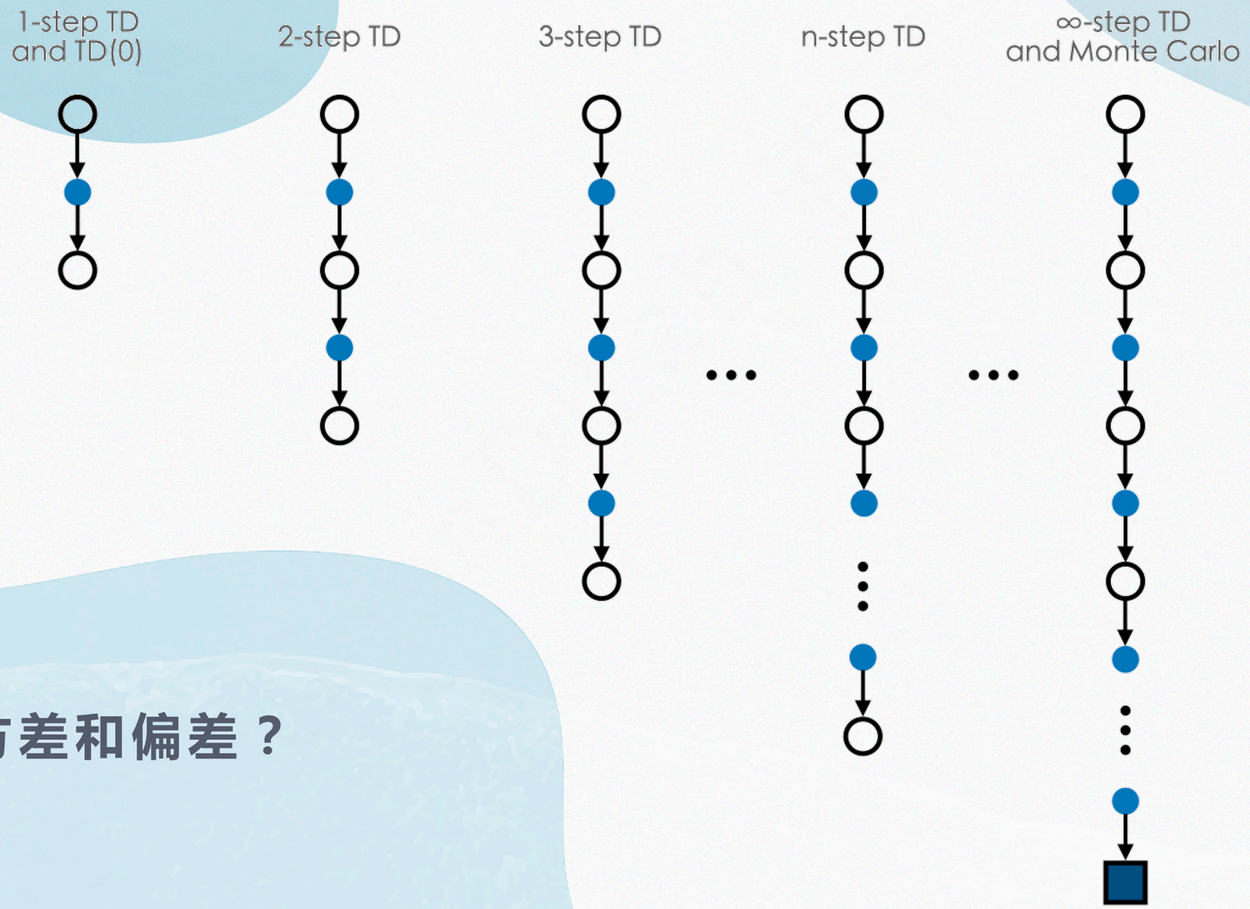
$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

- $V(S_t)$: New value of state t
- $V(S_t)$: Former estimation of value of state t
- α: Learning Rate
- R_{t+1}: Reward
- $\gamma V(S_{t+1})$: Discounted value of next state

Method Name	Bootstrapping	Sampling
DP	✓	✗
MC	✗	✓
TD	✓	✓

一重境界：权衡偏差与方差的GAE

偏差（Bias）



方差

（Variance）

- 如何更好地平衡方差和偏差？

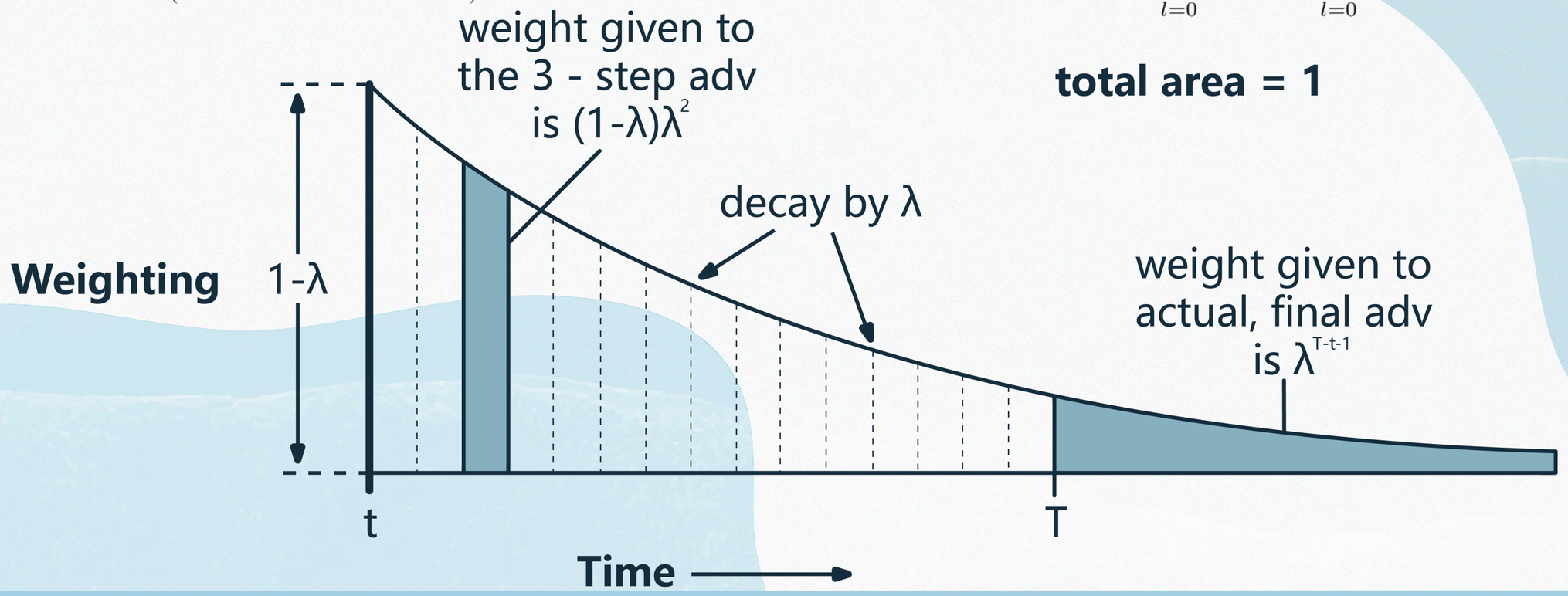
一重境界：权衡偏差与方差的GAE

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})$$

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \cdots\right)$$

$$\mathrm{GAE}(\gamma, 0): \quad \hat{A}_t := \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\mathrm{GAE}(\gamma, 1): \quad \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t)$$



High-Dimensional Continuous Control Using Generalized Advantage Estimation: https://arxiv.org/pdf/1506.02438.pdf

一重境界：权衡偏差与方差的GAE

Algorithm 1 Proximal Policy Optimization (PPO)

Input: initialize policy (actor) parameters θ and value (critic) parameters ϕ. Training epochs per collect E, trajectory length T, batch size B.
for $k = 0, 1, 2, ...$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$by running policy π_{θ_k} interaction with environment.
  Compute trajectory target return estimates $\hat{R}_t$. (e.g. n-step TD or other methods)
  Compute advantage estimates $\hat{A}^{\theta_k}$ with value V_{ϕ_k}. (e.g. GAE or other methods)
  for $e = 0, 1, ..., E-1$ **do**
    for minibatch:$b \in \mathcal{D}_k$ **do**
      Update the policy by maximizing the PPO-Clip objective with SGD:

$$L_\theta = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} \min(r(\theta)\hat{A}^{\theta_k}(s_t, a_t), \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\theta_k}(s_t, a_t))$$

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$$

      Fit the value by regression on mean-squared error with SGD:

$$L_\phi = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} (V_\phi(s_t) - \hat{R}_t)^2$$

    end for
  end for
end for

- 收集数据需要保留时间顺序
- 在收集完毕，训练之前计算
- 多个 trajectory 拼接后计算

```
next_value *= (1 - done)
delta = reward + gamma * next_value - value
factor = gamma * lambda_ * (1 - traj_flag)
adv = torch.zeros_like(value)
gae_item = torch.zeros_like(value[0])

for t in reversed(range(reward.shape[0])):
    gae_item = delta[t] + factor[t] * gae_item
    adv[t] = gae_item
```

GAE 代码示例：https://opendilab.github.io/PPOxFamily/gae_zh.html
Advantage 计算补充材料：https://github.com/opendilab/PPOxFamily/blob/main/chapter7_tricks/chapter7_supp_adv.pdf

二重境界：稳扎稳打之Recompute

$$L_\theta = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} \min(r(\theta)\hat{A}^{\theta_k}(s_t, a_t), \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\theta_k}(s_t, a_t))$$

- 新老策略尽可能地接近，IS 和 adv 可能存在的偏差就更小

- PPO 常用实现中包含两重循环
  - 第一重：训练多个epoch
  - 第二重，一个 epoch 中划分 N 个 minibatch
- 收集一次数据，训练 epoch x N 个迭代
- 只要更新一个迭代，新老策略就已经产生了不一致，这与 PPO 的设计初衷相冲突

PPO off-policy 版本计算补充材料：https://github.com/opendilab/PPOxFamily/blob/main/chapter7_tricks/chapter7_supp_ppo_offpolicy.pdf

二重境界：稳扎稳打之Recompute

Algorithm 1 Proximal Policy Optimization (PPO)

Input: initialize policy (actor) parameters θ and value (critic) parameters ϕ. Training epochs per collect E, trajectory length T, batch size B.
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$by running policy π_{θ_k} interaction with environment.
  Compute trajectory target return estimates $\hat{R}_t$. (e.g. n-step TD or other methods)
  for $e = 0, 1, \ldots, E-1$ **do**
    Recompute advantage estimates $\hat{A}^{\theta_k}$ with the latest value V_ϕ (e.g. GAE or other methods)
    for minibatch:$b \in \mathcal{D}_k$ **do**
      Update the policy by maximizing the PPO-Clip objective with SGD:

$$L_\theta = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} \min(r(\theta)\hat{A}^{\theta_k}(s_t, a_t), \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\theta_k}(s_t, a_t))$$

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$$

      Fit the value by regression on mean-squared error with SGD:

$$L_\phi = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} (V_\phi(s_t) - \hat{R}_t)^2$$

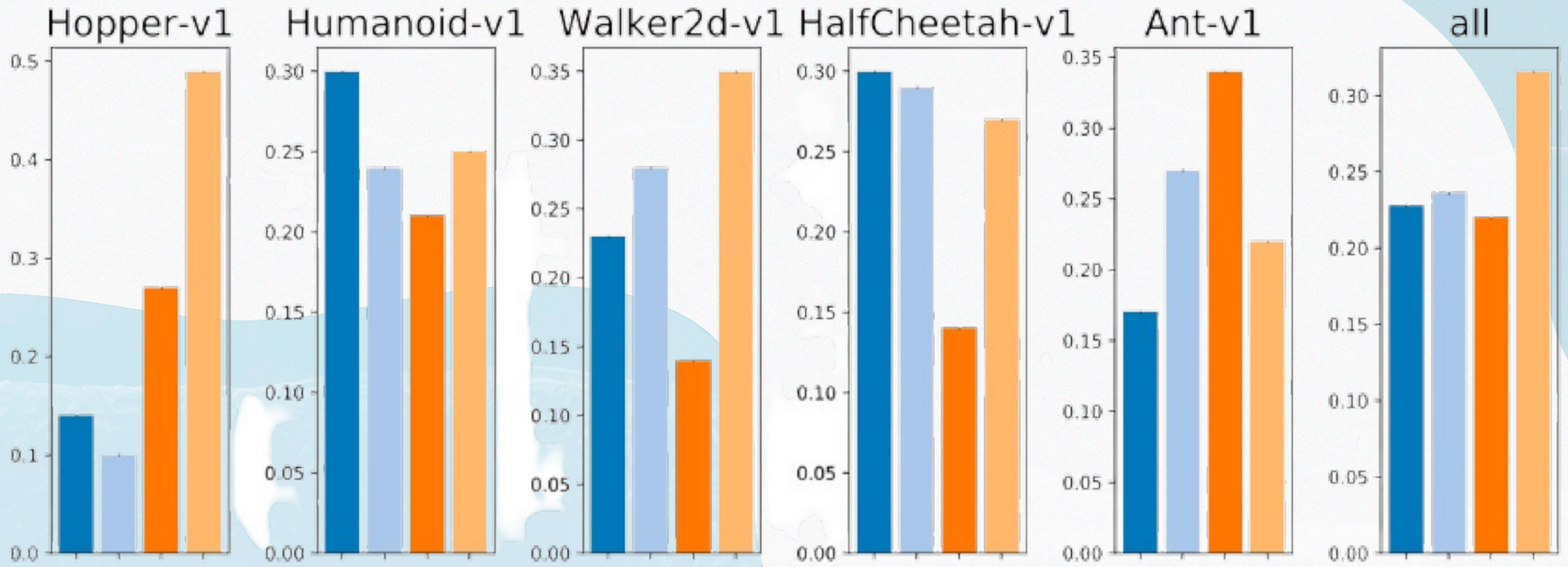
    end for
  end for
end for

- Recompute 步骤无需梯度计算，节省时间和显存
- 在第一重循环内部整体计算。不需要额外维护时间顺序关系，每次计算完成后先 shuffle 数据再切分minibatch 即可
- 在 epoch 较大时尤其有用

Recompute Advantage 代码示例：https://opendilab.github.io/PPOxFamily/recompute_adv_zh.html

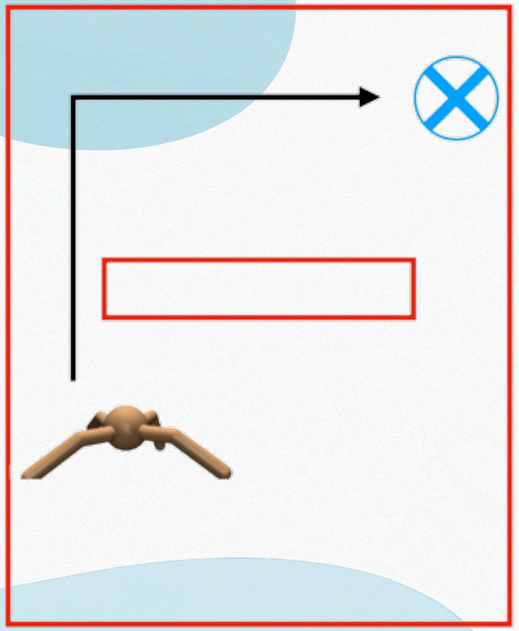
二重境界：稳扎稳打之Recompute

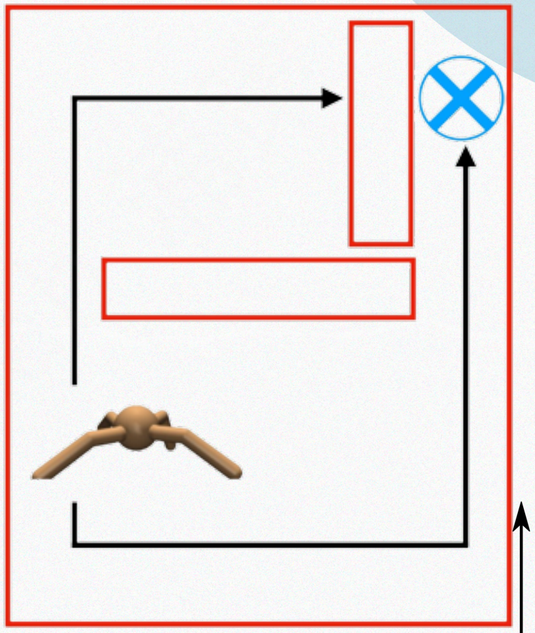
从左到右依次是：Fixed、Shuffle Trajectories、Shuffle Transitions、Shuffle Transitions + Recompute（整体效果最好）

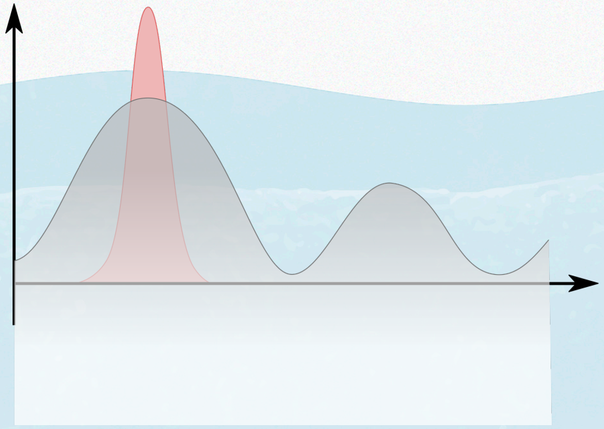


What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study: https://arxiv.org/pdf/2006.05990.pdf

三重境界：巧夺天工之Entropy



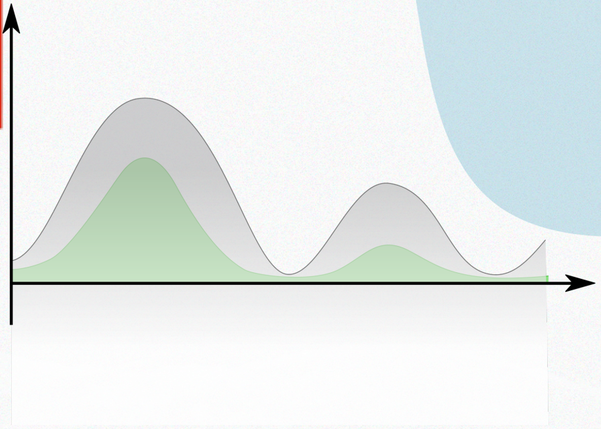




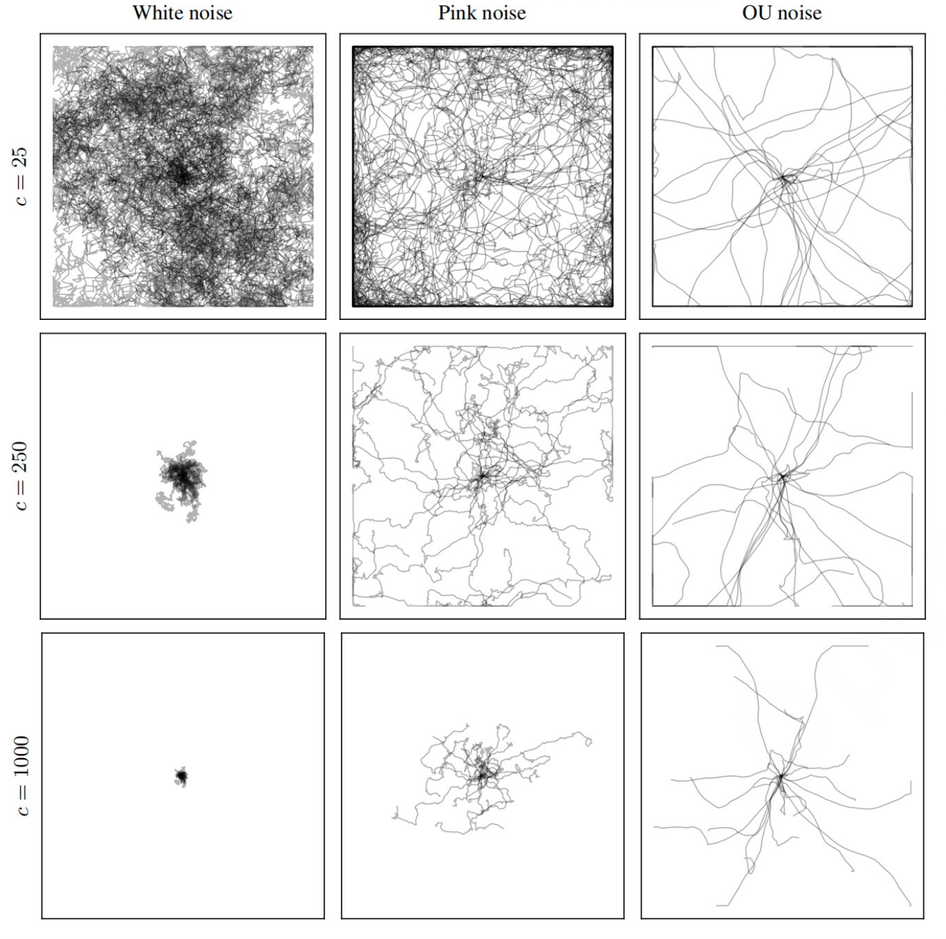
策略早熟：

策略以某种“投机取巧”的方式

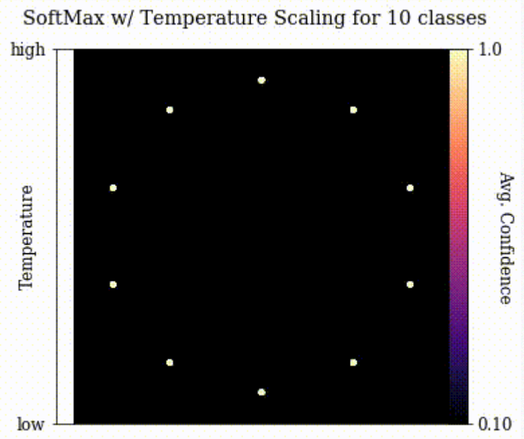
找到了局部最优解



三重境界：巧夺天工之Entropy



- 不同的环境类型，不同的环境设置，所需的策略探索能力不同
- PPO中控制策略探索能力的两个关键点：
  - 选择动作时的 temperature



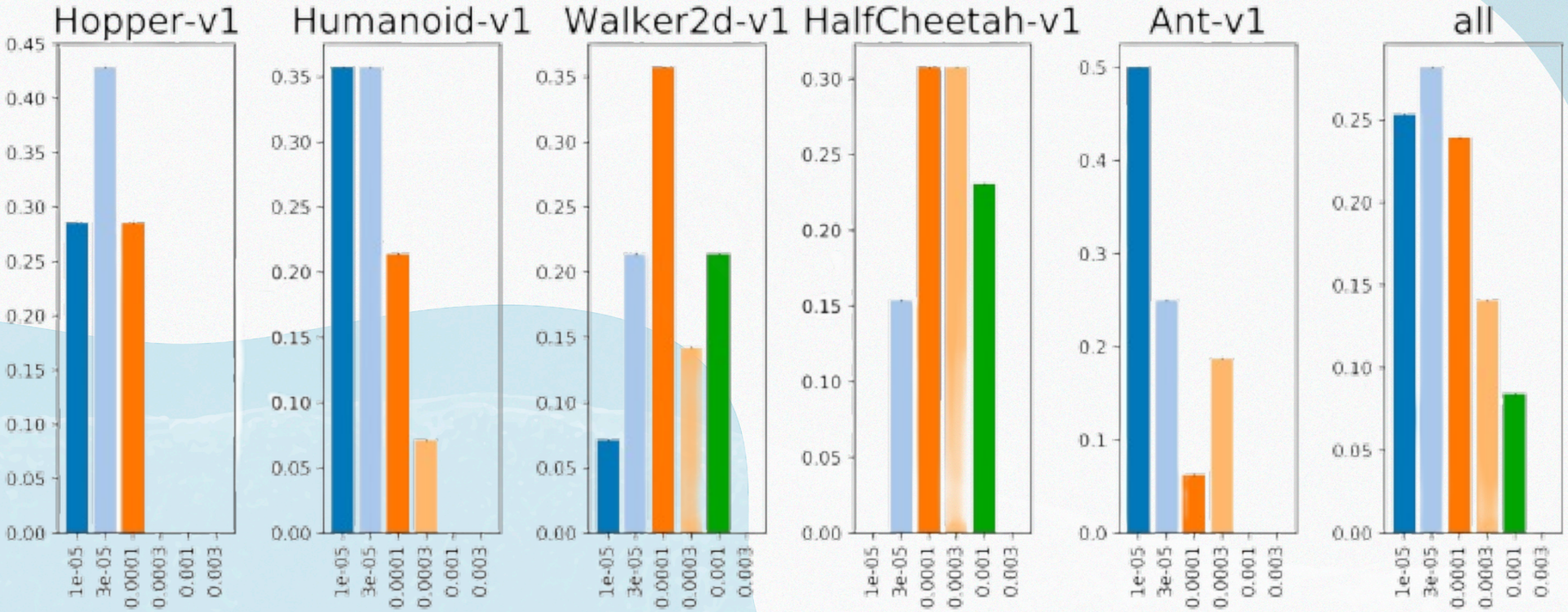
  - 计算损失函数时的 Entropy Bonus

Pink Noise Is All You Need: Colored Noise Exploration in Deep Reinforcement Learning: https://openreview.net/pdf?id=hQ9V5QN27eS
GIF Ref: https://charlielehman.github.io/post/visualizing-tempscaling/

三重境界：巧夺天工之Entropy

合理的 Entropy Bonus 才能有效防止策略“早熟”并高效稳定收敛

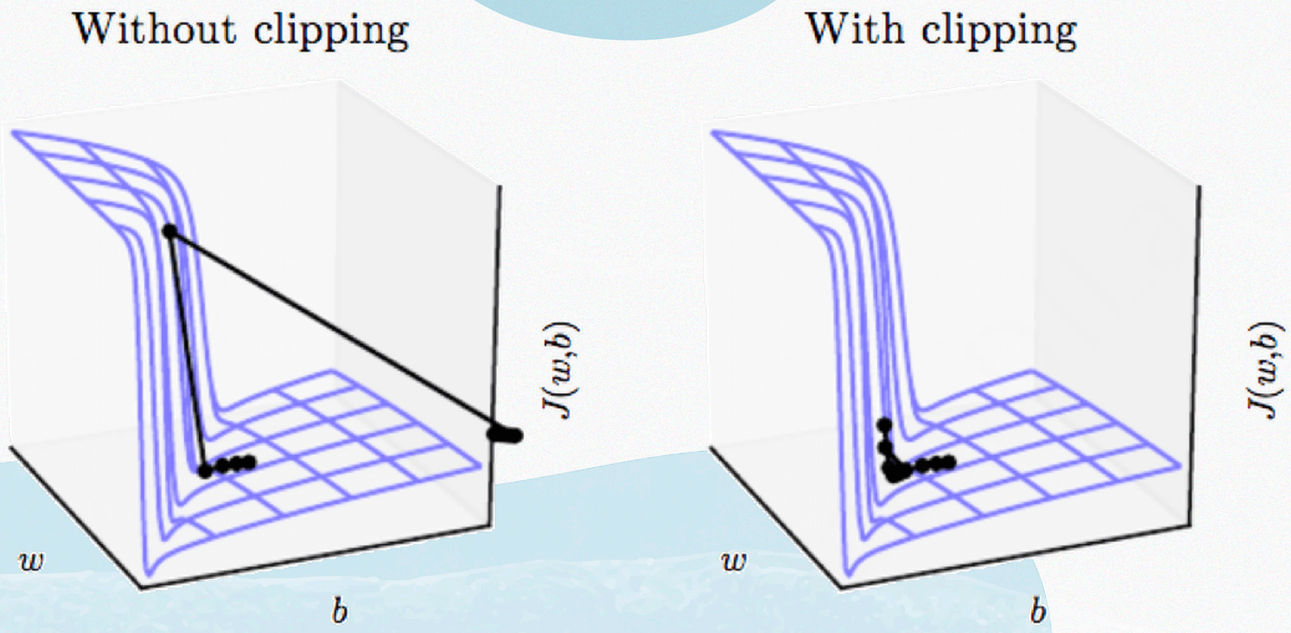
更进一步，可以在训练周期内设置相应的 bonus 衰减（指数衰减 or 类比 SAC）



What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study: https://arxiv.org/pdf/2006.05990.pdf
Entropy 补充材料：https://github.com/opendilab/PPOxFamily/blob/main/chapter7_tricks/chapter7_supp_entropy.pdf

四重境界：免死金牌之Grad Clip

四重境界：免死金牌之Grad Clip



Standard gradient $g(\theta) = \frac{1}{N}\sum_{n=1}^{N}\nabla l_\theta(x_n, y_n)$

Clipped gradient $\bar{g}_\tau(\theta) = \text{clip}_\tau(g(\theta))$

↓

Clip norm to τ

四重境界：免死金牌之Grad Clip

梯度的最大范数（max norm）：

$$\text{total\_norm}^{\infty} = \max_{\theta_i \in \Theta} |\text{grad}(\theta_i)|$$

梯度的 p-范数（p-norm）：

$$\begin{aligned}\text{total\_norm} &= (\sum_{\theta \in \Theta}((\sum_{\theta_i}\text{grad}(\theta_i)^p)^{\frac{1}{p}})^p)^{\frac{1}{p}} \\ &= (\sum_{\theta \in \Theta}(\sum_{\theta_i}\text{grad}(\theta_i)^p))^{\frac{1}{p}}\end{aligned}$$

裁减系数（1e-6用于避免分母为零）：

$$\text{clip\_coef} = \frac{\text{max\_norm}}{\text{total\_norm}}$$

将系数的最大值固定为1

如果 total_norm < max_norm，torch.clamp 操作将 clip_coef 的最大值固定为1，所以 clip_coef_clamped = 1，这样 total_norm 将保持不变。
如果 total_norm > max_norm，将对原来的梯度进行裁减，使得裁减后的梯度对应的 total_norm 的大小为 max_norm：

$$\begin{aligned}\text{total\_norm}' &= (\sum_{\theta \in \Theta}(\sum_{\theta}(\text{grad}(\theta_i) \cdot \frac{\text{max\_norm}}{\text{total\_norm}})^p))^{\frac{1}{p}} \\ &= \frac{(\sum_{\theta \in \Theta}(\sum_{\theta}\text{grad}(\theta_i)^p))^{\frac{1}{p}}}{\text{total\_norm}} \cdot \text{max\_norm} \\ &= \text{max\_norm}\end{aligned}$$

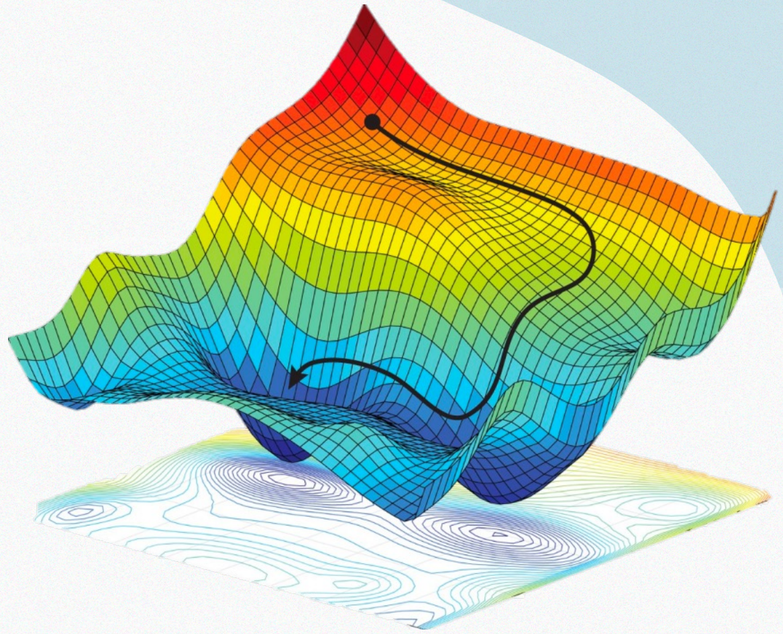
```python
    if norm_type == inf:
        norms = [g.detach().abs().max().to(device) for g in grads]
        total_norm = norms[0] if len(norms) == 1 else torch.max(torch.stack(norms))
    else:
        total_norm = torch.norm(torch.stack([torch.norm(g.detach(), norm_type).to(device) for g in grads]), norm_type)
    clip_coef = max_norm / (total_norm + 1e-6)
    clip_coef_clamped = torch.clamp(clip_coef, max=1.0)
    for g in grads:
        g.detach().mul_(clip_coef_clamped.to(g.device))
    return total_norm
```

Grad Clip 代码示例：https://opendilab.github.io/PPOxFamily/grad_clip_zh.html

五重境界：笨鸟先飞之Init

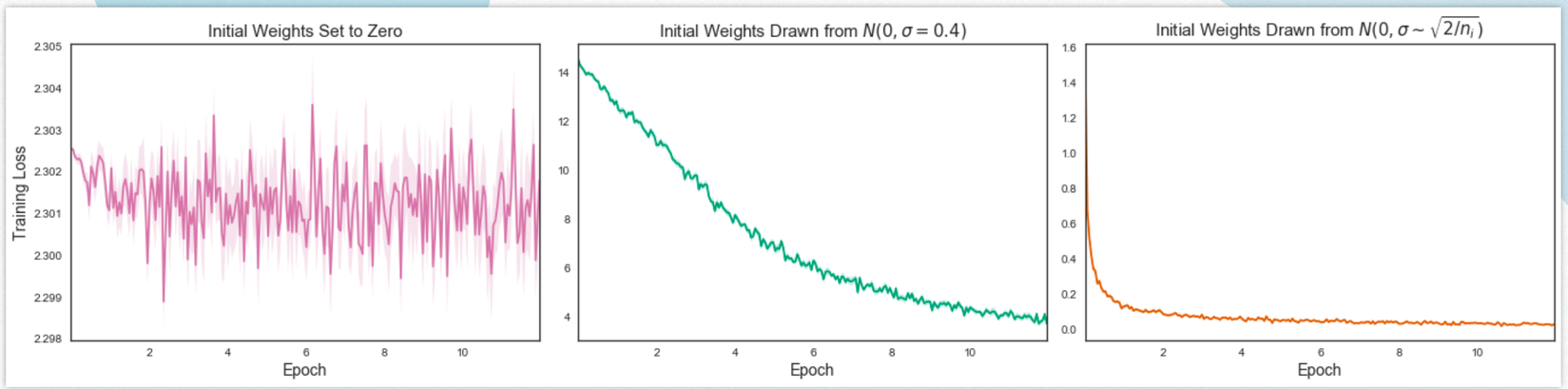
神经网络初始化设计的五大因素

- 输入层数量
- 输出层数量
- 非线性函数（激活函数）的类型
- 网络结构的类型（Linear/Conv/MHA）
- 网络的设计使用目的

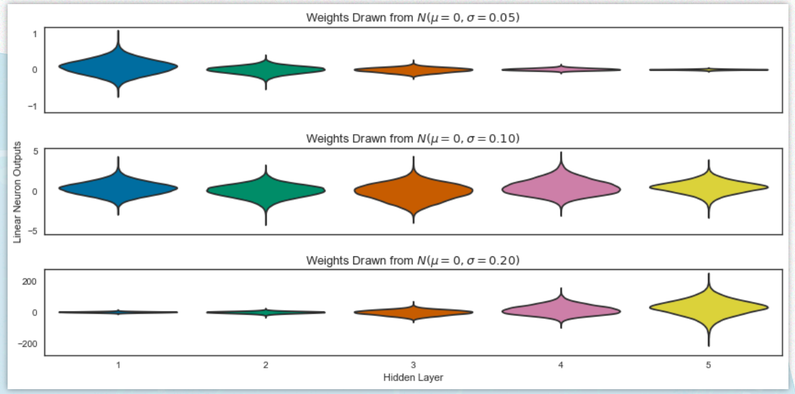


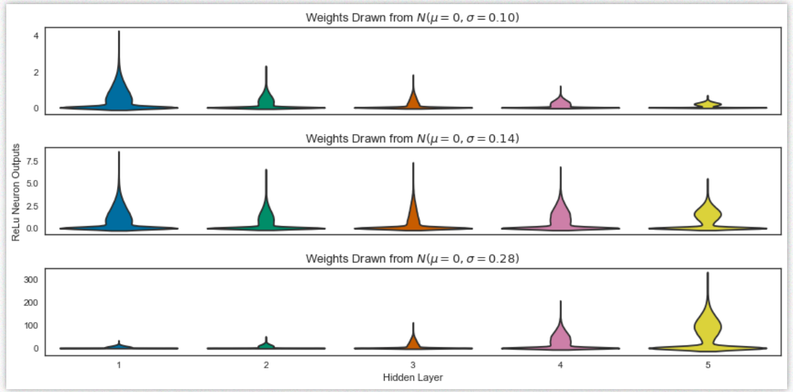
五重境界：笨鸟先飞之Init

- 为什么不使用全零或固定方差的初始化



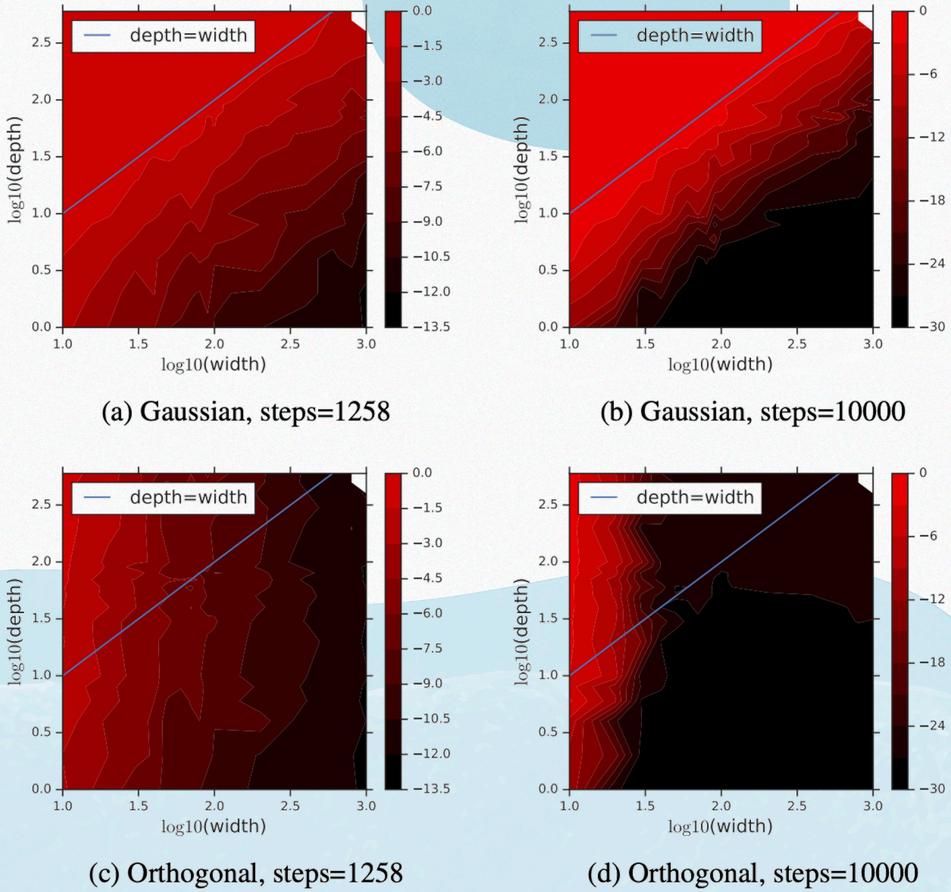
- 为什么要初始化考虑非线性函数的类型





Ref: https://intoli.com/blog/neural-network-initialization/

五重境界：笨鸟先飞之Init

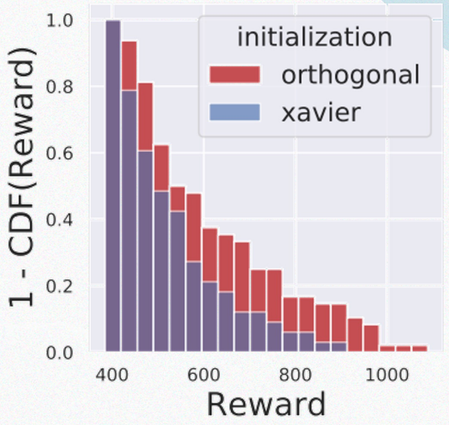


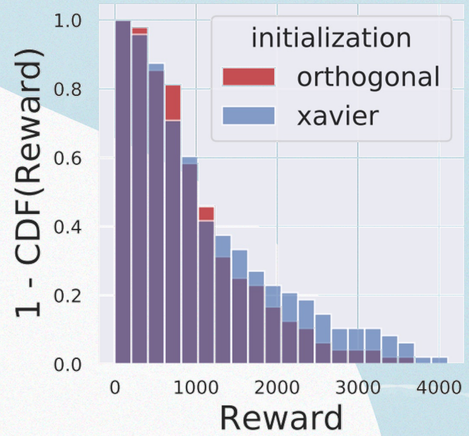
(a) Gaussian, steps=1258

(b) Gaussian, steps=10000

(c) Orthogonal, steps=1258

(d) Orthogonal, steps=10000



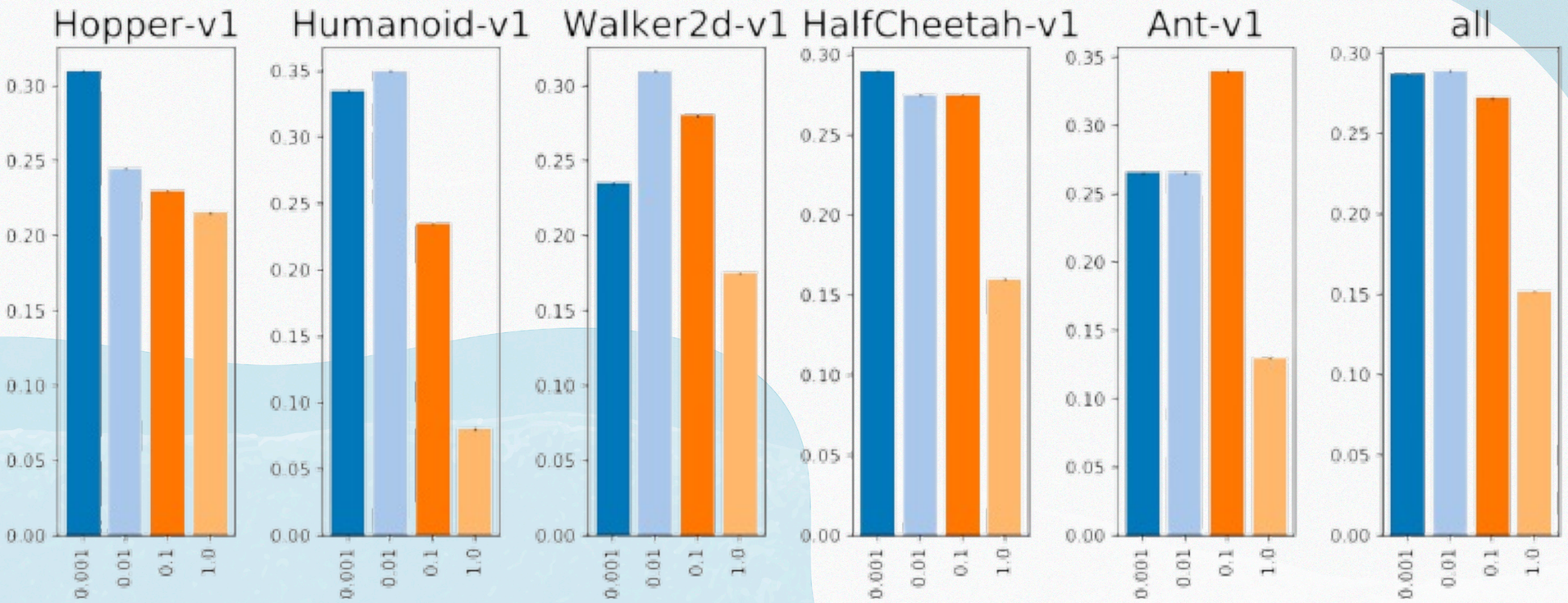



```
# Compute the qr factorization
q, r = torch.linalg.qr(flattened)
# Make Q uniform according to
# https://arxiv.org/pdf/math-ph/0609050.pdf
d = torch.diag(r, 0)
ph = d.sign()
q *= ph
# Initialize tensor
with torch.no_grad():
    tensor.view_as(q).copy_(q)
    tensor.mul_(gain)
```

PROVABLE BENEFIT OF ORTHOGONAL INITIALIZATION IN OPTIMIZING DEEP LINEAR NETWORKS: https://arxiv.org/pdf/2001.05992.pdf
IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO: https://arxiv.org/pdf/2005.12729.pdf
正交初始化代码示例：https://opendilab.github.io/PPOxFamily/orthogonal_init_zh.html

五重境界：笨鸟先飞之Init

策略输出层的特殊初始化，保证训练初期的均匀探索

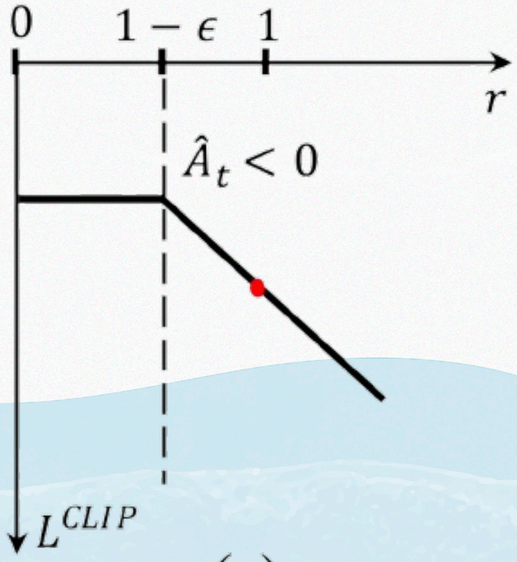


What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study: https://arxiv.org/pdf/2006.05990.pdf

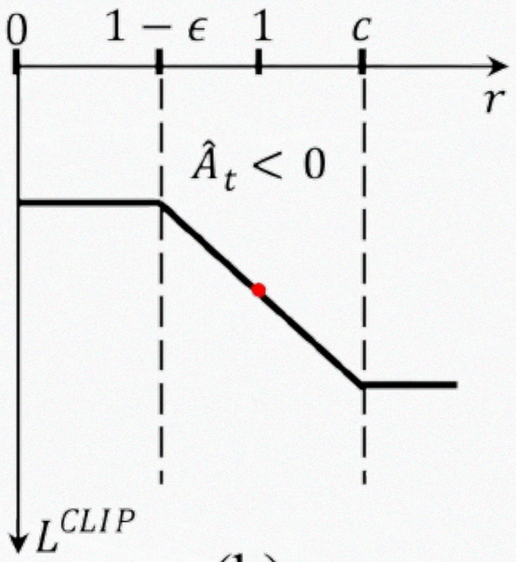
六重境界：亡羊补牢之Extra Clip

问题：当 A<0 时，如果新旧策略差异非常大（例如分布式训练情形），IS会异常偏大

解法：对 A<0 的情况额外增加一个 clip 操作，即 dual clip 形式



(a)



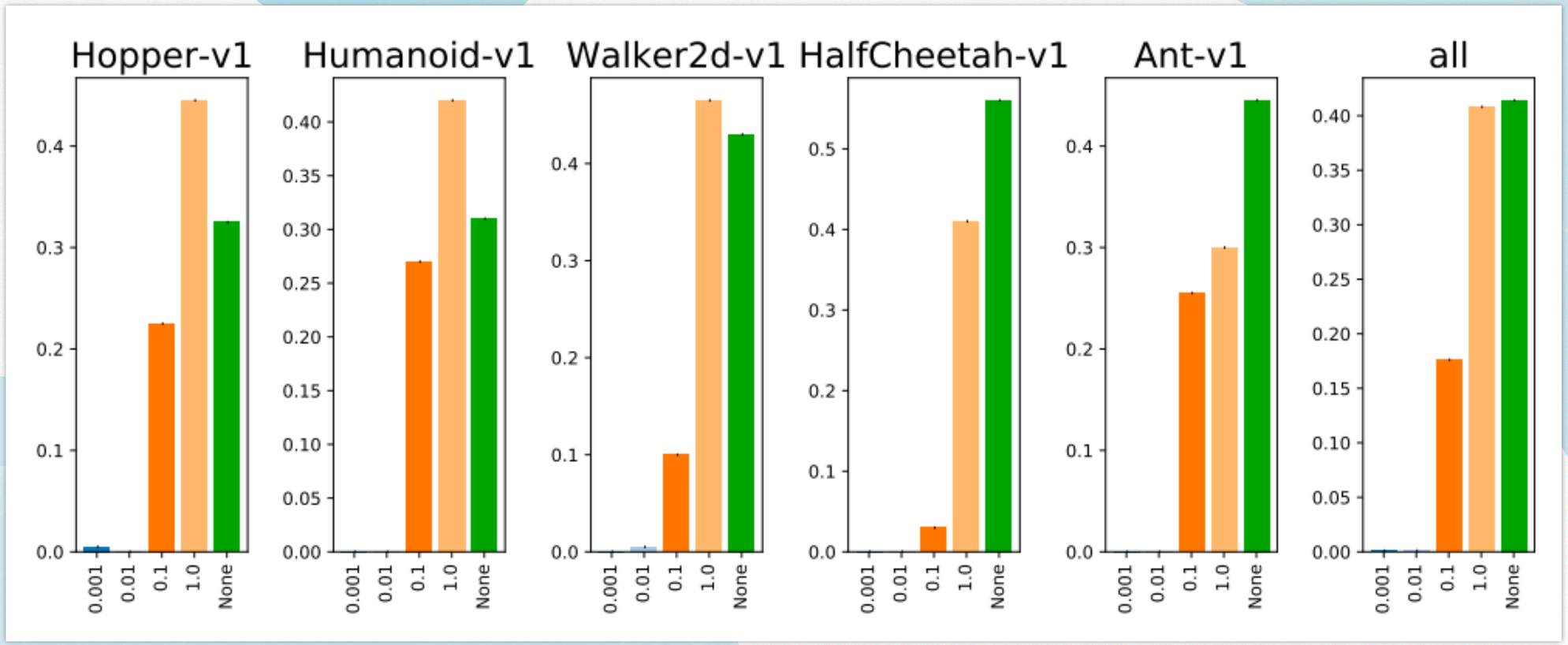
(b)

$$L_\theta = \frac{1}{B \cdot T} \sum_{\tau \in \mathrm{b}} \sum_{t=0}^{T-1} \max \left(\min(r(\theta)\hat{A}^{\theta_k}, \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\theta_k}), c\hat{A}^{\theta_k} \right)$$

Mastering Complex Control in MOBA Games with Deep Reinforcement Learning: https://arxiv.org/pdf/1912.09729.pdf
Dual Clip 代码示例: https://opendilab.github.io/PPOxFamily/dual_clip_zh.html

六重境界：亡羊补牢之Extra Clip

做法：对 Value Network 的优化也添加 clip 操作



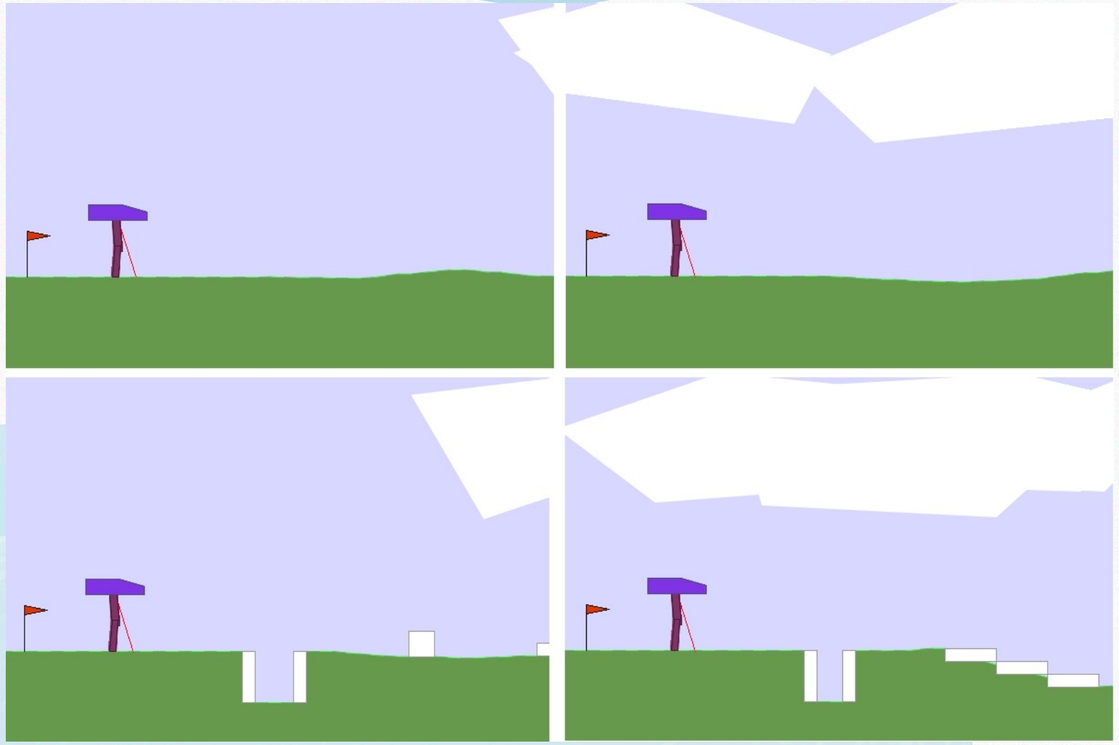
What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study: https://arxiv.org/pdf/2006.05990.pdf
Value Clip 代码示例：https://opendilab.github.io/PPOxFamily/value_clip_zh.html

Realm 7

七重境界：合理利用随机

环境的随机种子：影响环境的状态空间（布局，内容）和状态转移（物理仿真参数）

Bipedalwalker



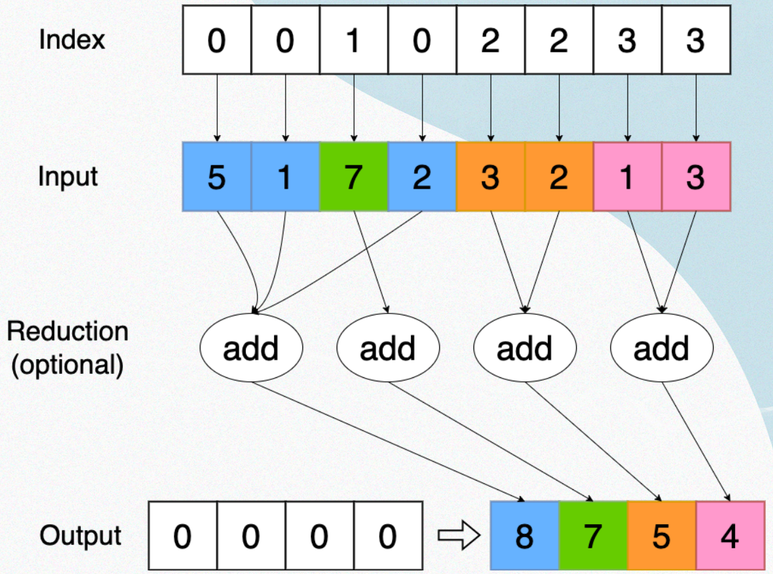
Procgen: procedurally-generated

DI-engine + Bipedalwalker: https://di-engine-docs.readthedocs.io/zh_CN/latest/13_envs/bipedalwalker_zh.html
DI-engine + Procgen: https://di-engine-docs.readthedocs.io/zh_CN/latest/13_envs/procgen_zh.html

七重境界：合理利用随机

工具库导致的随机

- 收集数据时随机选择动作
- 训练前随机从 buffer 中采样数据
- 神经网络参数的初始化
- 数据处理/变换带来的随机
- 不同计算设备引发的随机
- 异步/并行计算带来的随机




```
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
```

Ref: https://pytorch.org/docs/stable/notes/randomness.html?highlight=reproduce
Ref: https://pytorch.org/docs/stable/generated/torch.use_deterministic_algorithms.html#torch.use_deterministic_algorithms

Realm 7

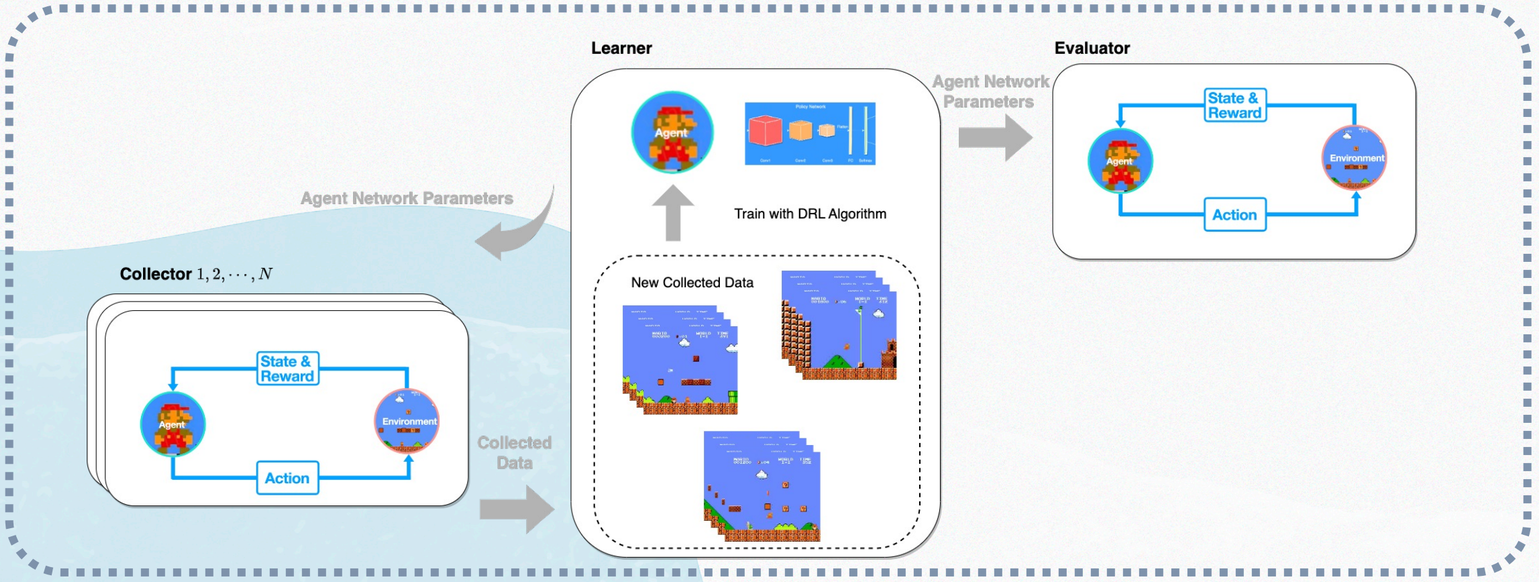
七重境界：合理利用随机

收集数据（collect）：

- 目的是收集训练数据。欢迎各类合理的随机性，增加数据的多样性，从而提升策略通用性

评估测试（eval）：

- 目的是评估智能体的性能。单次测试要足够覆盖环境的随机性，但多次测试之间要可比



Realm 7

七重境界：合理利用随机

		分散程度（Dispersion）	风险度量（Risk）
训练	多次训练时刻 （在一次训练之内）	滑动窗口平均后，算法表现的IQR	短期指标：算法表现的一阶差分的CVaR 长期指标：算法表现最大回撤的CVaR
	多次训练	低通滤波后的算法表现的 IQR	算法表现的 CVaR
测试	固定策略的多次试验	对于某个固定策略的多次试验中算法表现的 IQR	对于某个固定策略的多次试验中算法表现的 CVaR

IQR:

inter-quartile range

即75和25分位点之差

CVaR:

conditional value at risk

即最坏情况下的期望回撤

MEASURING THE RELIABILITY OF REINFORCEMENT LEARNING ALGORITHMS: https://openreview.net/pdf?id=SJlpYJBKvH

下节预告

（八）突破智能体终极界限

- PPO + 分布式训练
- PPO + 大语言模型（LLM）
- 神秘终极挑战任务