评分：_____

上海大学
Shanghai University

人工智能导论

课程实践报告

自博弈的强化学习五子棋智能体

学　　号　25725250

姓　　名　闵泳昶

一、项目介绍

1. AI 技术

Github 仓库网址：https://github.com/minsyc00/Five-Chess

2. 项目背景

(1) 强化学习载体

五子棋规则简单，落子位置有限却蕴含复杂博弈策略，是典型的完全信息、零和对抗环境。与围棋相比，棋盘规模较小，适合在有限算力下探索深度强化学习算法。

(2) 传统方法的局限

传统 AI 主流是基于规则库、估值函数或 Alpha-Beta 剪枝的搜索方式，难以在高维状态空间中挖掘长远策略；人工设计的启发式需大量专业知识，泛化能力有限。

(3) 强化学习的机遇

自 AlphaGo 以来，“深度学习+强化学习+自博弈”已验证在棋类游戏中的有效性。通过自学习不断迭代策略，无需人工标注即可取得强大的基准表现。

(4) 课程实践目标

本实践从零构建 PPO 自博弈系统，加深对策略梯度、价值函数、蒙特卡洛树搜索的理解，同时训练工程思维，掌握模型保存、GPU 加速、人机交互等实际技能，为后续的强化学习及其工程实践打下扎实

基础。

3. 核心目标

本次课程实践，通过神经网络结构与强化学习，构造集自博弈训练、人机交互、策略搜索于一体的五子棋智能体系统。项目以策略梯度家族的 Proximal Policy Optimization（PPO）为核心，结合卷积神经网络提取棋盘特征、广义优势估计（GAE）改善训练稳定性，并在对弈阶段引入蒙特卡洛树搜索（MCTS）作为决策增强，显著提升 AI 的棋力与实时响应能力。系统采用 PyTorch 作为深度学习框架，pygame 实现棋盘界面；训练阶段实现了自对弈数据自动采集、按优化步数保存模型、损坏 checkpoint 的鲁棒加载策略、人机模式实时加载最新权重等功能。经过多轮训练与人机对战评估，AI 能够学习到占据中心、做活三活四、堵截对手等策略，中后期对弈胜率明显提升。实践过程中重点解决了旧版模型不兼容导致的加载失败、训练数值震荡、推理时落子不稳定等问题，并总结了 PPO 超参数调优、MCTS 搜索宽度选取、模型持久化策略等工程经验。

4. AI 技术

(1) 强化学习

状态 s：棋盘上黑白棋子布局。

动作 a：在空位落子。

奖励 r：胜利+1，失败-1，平局 0。

策略 π(a|s)：给定状态下采取动作的概率。

价值函数 V(s)：从状态出发按当前策略获得的期望回报。

(2) PPO

定义旧策略 π_old 和新策略 π 的截断比率 r(θ)=π(a|s)/π_old(a|s)。

引入剪切目标：L_clip=min(r*A,clip(r,1-ε,1+ε)*A)保持更新稳定。

广义优势估计 GAE(λ)减少方差，兼顾偏差。

通过 mini-batch+多 epoch 重复利用采样数据，提高样本效率。

(3) 卷积神经网络

输入：(batch,2,15,15)，两通道分别表示黑子与白子位置。

架构：四层 3x3same-padding 卷积保持空间尺寸，随后全连接层输出 225 维动作 logits 及一个标量价值。

激活函数：ReLU；价值头使用 Tanh 限制[-1,1]区间。

(4) 蒙特卡洛树搜索

Selection：采用 UCB1 或 PUCT 公式选子，以策略先验作为引导，提升搜索效率。

Expansion：当遇到新节点，根据当前策略 logits 对合法动作做 softmax 分布作为先验概率。

Simulation：调用价值网络对叶子节点估值。

Backpropagation：从叶子向上回传价值并累计访问次数，最终选择访问次数最多的动作。

二、实践过程

1. 模块划分

reinforce_learning.py：定义网络、PPO 训练流程、模型保存/加载、动作采样、缓冲区管理。

game_thread.py：承担自博弈线程、人机界面逻辑、训练对局管理、MCTS 搜索。

main.py：提供命令行入口，默认训练，--play 进入人机对弈。

net.py：CNN 架构定义。

models/latest.pt：全局唯一的模型权重。

2. 实现步骤

(5) PPO 实现

select_action：对合法动作掩膜后重新归一化概率，线上采样时可 deterministic（用于 MCTS）或 stochastic（用于自博弈探索）。

record_episode：根据最终赢家为整局填充奖励，同时保存 log_prob、value 用于 PPO 算法。

train：实现 mini-batch 循环，包含策略损失、价值损失、熵正则；优化器选 Adam，学习率 3e-4；梯度裁剪上限 1.0。

save_model：只保留最新权重；利用临时文件写入成功后原子替换，防止训练中断导致文件损坏。

(6) 自博弈流程重写

play_training_game：在训练线程中循环调用，自动加载最新checkpoint，启动新的对局；每步都记录状态与策略信息直至终局。

handle_game_end：终局时record_episode到train再到save_model形成闭环，确保训练与保存同步进行；同时打印局数和胜者信息方便调试。

(7) 人机对战

loop_human_vs_ai：启动pygame界面，读取最新模型，与玩家（默认执黑）对战；玩家落子后AI自动调用ai_move。

ai_move：优先调用_mcts_select，在合法动作上展开若干次模拟（默认100次），如树搜索失败则退化为PPO策略的greedy决策。

load_latest_model()：保证对战时使用的权重为最新训练成果。

(8) 鲁棒性提升

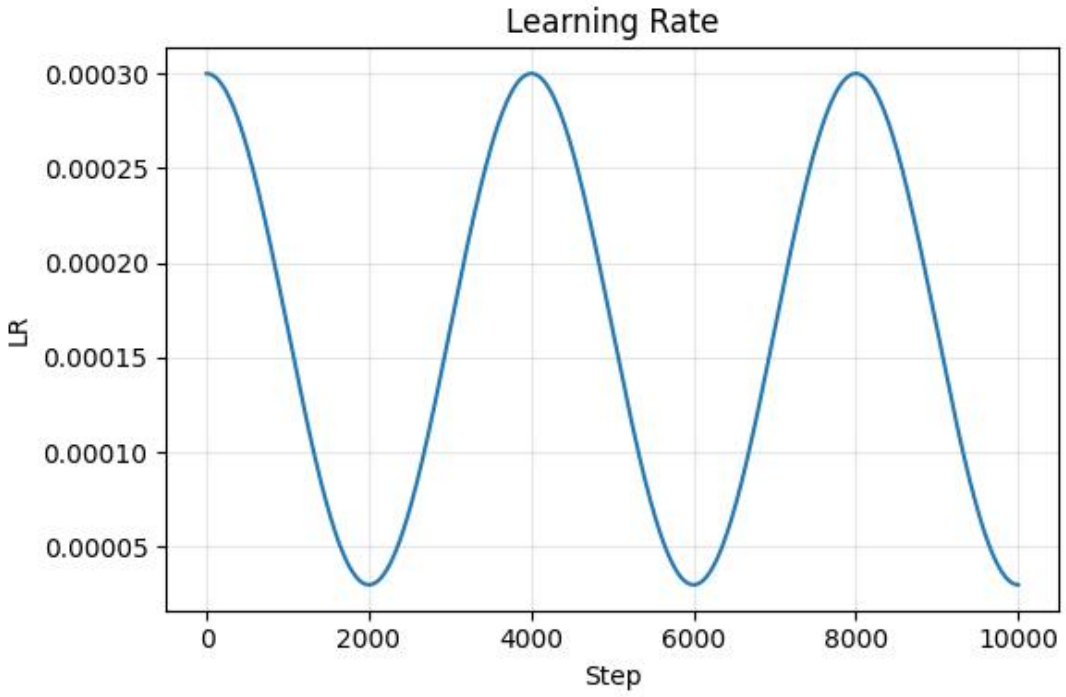
模型加载失败不再抛出异常，而是提示并返回False，防止程序崩溃。

增加GPU信息打印及CUDA安装指导，方便在不同环境部署。

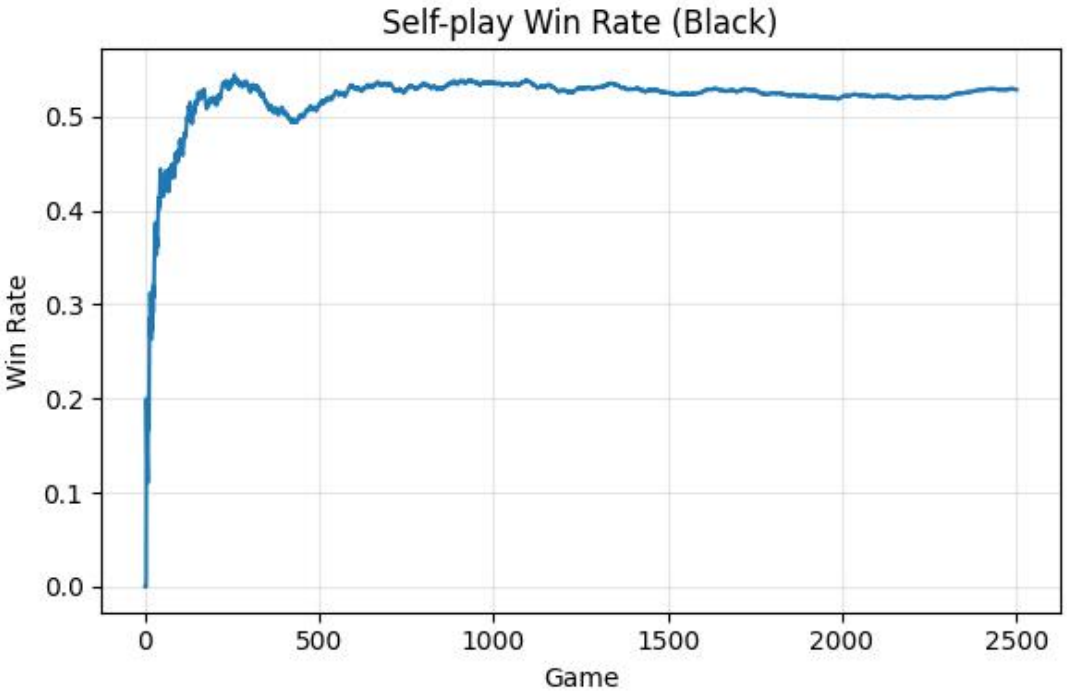
Save与load操作均加try/except，确保在停电或中断后能感知损坏文件并提醒用户删除。

3. 实验与结果

(1)余弦学习率曲线



(2)执黑胜率统计



三、心得体会

(1)对 PPO 的理解更深入

通过手动实现剪切损失、优势归一化、熵正则，理解了 PPO 在稳定性与效率间的平衡。对 ε 、value_loss_coef、entropy_coef 等超参数的敏感性也有直观认识。

(2)工程实现的重要性

理论再好，如果模型保存损坏或加载失败，系统就会中断。实践中强调了异常处理、日志输出、文件操作的鲁棒性。

模型只保留最新版本，同步训练和推理逻辑，避免不同线程出现参数“漂移”。

(3)MCTS 与神经网络的结合

在人机对战中，纯策略采样容易犯错，加入 MCTS 后决策稳定性提升明显。说明在有限搜索深度下，策略网络的先验和价值网络的评估是一种有效组合。

(4)人机交互体验的提升

pygame 虽简洁但足够展示棋盘和落子过程；通过热键、提示信息和响应速度优化，使用户体验较为流畅。

(5)持续迭代的意义

从 DQN 到 PPO，再到 MCTS，每一步迭代都需要重新审视系统架构和依赖；课程实践鼓励不断尝试和迭代改进。

四、总结展望

本次《人工智能导论》课程实践通过对五子棋 AI 系统的深度改造，完整体验了从理论推导、算法实现、工程调试到功能验证的全过程。实践中不仅掌握了 PPO、GAE 和 MCTS 等核心算法，也体会到模型持久化、异常处理、实时交互等工程细节对系统可靠性的决定性影响。经过多轮自博弈训练和人机对战评估，AI 的棋力显著提升，能自主掌握中心先手、活三造势、紧急防守等策略。但在算法和硬件性能利用上仍有提升空间。

(1)训练效率与棋力提升

目前自博弈采用单线程顺序进行，效率有限；未来可考虑采用多线程或多进程并行自博弈，配合共享网络参数更新。

可引入经验回放 Buffer，增加数据多样性；或加入自对弈中对最优策略的惩罚奖励，提高探索质量。

结合更深的网络结构（Residual blocks）和策略头/价值头分离的架构，增强表达能力。

(2)MCTS 搜索参数

现阶段模拟次数较少，遇到复杂局面仍有误判。可以动态调整模拟次数或结合温度参数，平衡响应速度和棋力。

(3)对战体验

当前人机对弈仅支持本地鼠标操作，可扩展网络对战、AIvsAI

观摩模式，甚至加入语音提示、历史对局复盘等功能。

(4)可视化与分析

最好添加训练统计面板，如胜率曲线、策略熵变化图、落子热度地图等，帮助更直观地分析训练效果。

(5)算法扩展

可以引入分层策略、组合神经网络与 Alpha-Beta 剪枝、或者在 PPO 基础上加入 Value Clipping、Trust Region 等改进，以探索更多强化学习算法与棋类 AI 的融合。