**实验二实验报告**

**实验背景**

**1.1 研究动机与意义**

五子棋作为一种经典的策略类棋类游戏，具有规则简单但策略复杂的特点，是验证人工智能算法效果的理想测试平台。本实验旨在通过深度强化学习技术，特别是结合蒙特卡洛树搜索（Monte Carlo Tree Search, MCTS）和深度神经网络的方法，构建一个具有较强棋力的五子棋AI系统。

这种方法的核心思想来源于AlphaGo系列算法，通过自我对弈的方式不断提升模型的决策能力，无需依赖大量的人类专家棋谱数据。这对于理解和实现通用人工智能具有重要的理论价值和实践意义。

**1.2 技术路线概述**

本项目采用了策略价值网络（Policy-Value Network）与蒙特卡洛树搜索相结合的技术路线：

**策略价值网络负责两个核心功能：**

* **策略评估（Policy）**：给出当前棋盘状态下每个可行落子位置的概率分布，指导搜索过程中的节点选择
* **价值评估（Value）**：评估当前棋盘状态对当前玩家的胜负期望，用于终止搜索时的局面评估

**蒙特卡洛树搜索作为决策引擎：**

* 利用策略网络的输出进行智能的树搜索，平衡探索（exploration）与利用（exploitation）
* 通过大量模拟对弈来评估不同落子选择的优劣
* 最终选择访问次数最多或胜率最高的着法作为实际落子

**自我对弈训练机制：**

* 系统与自身进行对弈，生成训练数据
* 每局对弈的结果用于更新网络参数，实现持续学习和改进
* 这种方式避免了对人类棋谱的依赖，能够发现人类可能忽略的策略

**1.3 实验环境与设置**

本实验在8×8的棋盘上进行五子棋对弈，相比传统的15×15棋盘，这种设置能够：

* 降低计算复杂度，提高训练效率
* 减少搜索空间，使得模型能够更快收敛
* 便于实验验证和调试

实验支持GPU加速训练，使用SGD优化器，并实现了完整的检查点保存和恢复机制，确保训练过程的稳定性和可持续性。

**2. 模型设计**

**2.1 整体架构设计**

本项目的核心架构包含两个主要组件：策略价值网络和蒙特卡洛树搜索算法。这种设计充分体现了深度学习与传统搜索算法结合的优势。

**系统工作流程：**

1. **状态表示**：将当前棋盘状态转换为三维张量，包含对手棋子分布、己方棋子分布和当前玩家标识
2. **网络推理**：策略价值网络对当前状态进行评估，输出落子概率分布和状态价值
3. **树搜索**：MCTS利用网络输出进行多轮模拟，构建决策树
4. **动作选择**：根据搜索结果选择最优落子位置
5. **状态更新**：执行落子，更新棋盘状态，进入下一轮决策

**2.2 策略价值网络详细设计**

**2.2.1 网络架构选择**

项目实现了两种不同复杂度的网络架构：

**简化网络（Easy\_model）：**

输入 → Conv2d(32) → BatchNorm → ReLU →

Conv2d(64) → BatchNorm → ReLU →

Conv2d(128) → BatchNorm → ReLU → 输出特征

**ResNet架构（resnet18）：** 基于残差网络设计，包含多个BasicBlock，每个块包含两个3×3卷积层和跳跃连接。这种设计能够：

* 解决深层网络的梯度消失问题
* 提供更强的特征表达能力
* 保持训练稳定性

**2.2.2 双头输出设计**

网络采用共享特征提取器加双头输出的设计：

**策略头（Policy Head）：**

* 输入：共享特征图（128通道）
* 处理：1×1卷积降维至16通道 → BatchNorm → ReLU → 全连接层
* 输出：棋盘大小平方的概率分布（如8×8=64维）
* 作用：指导MCTS中每个位置的选择概率

**价值头（Value Head）：**

* 输入：共享特征图（128通道）
* 处理：1×1卷积降维 → 全连接层 → Tanh激活
* 输出：标量值，范围[-1,1]
* 作用：评估当前局面对当前玩家的胜负期望

**2.2.3 状态表示方法**

棋盘状态被编码为3×H×W的三维张量：

* **第0层**：对手棋子位置（1表示有子，0表示无子）
* **第1层**：己方棋子位置（1表示有子，0表示无子）
* **第2层**：当前玩家标识（全1表示黑棋，全0表示白棋）

这种表示方法确保了网络能够理解当前的完整游戏状态，包括棋子分布和行棋权。

**2.3 蒙特卡洛树搜索算法设计**

**2.3.1 树结构设计**

MCTS使用节点-边的树结构：

**节点（MonteTreeNode）：**

* 存储当前玩家信息和访问统计
* 管理所有可能的后续动作（边）
* 负责节点扩展和价值回传

**边（MonteTreeEdge）：**

* 连接父节点和子节点
* 存储先验概率（来自策略网络）
* 维护访问次数和累积价值
* 实现UCB（Upper Confidence Bound）选择策略

**2.3.2 UCB选择策略**

边的选择基于UCB公式：

UCB = Q + c\_puct × P × √(N\_parent) / (1 + N\_edge)

其中：

* Q：平均价值（累积价值/访问次数）
* P：先验概率（来自策略网络）
* N\_parent：父节点总访问次数
* N\_edge：当前边访问次数
* c\_puct：探索常数

这个公式巧妙地平衡了：

* **利用（Exploitation）**：选择当前价值高的动作
* **探索（Exploration）**：给访问较少的动作更多机会

**2.3.3 搜索流程**

每次MCTS搜索包含四个阶段：

**1. 选择（Selection）**：从根节点开始，使用UCB策略选择路径直到叶节点 **2. 扩展（Expansion）**：如果叶节点不是终局状态，使用策略网络扩展新节点 **3. 评估（Evaluation）**：使用价值网络评估新节点的价值 **4. 回传（Backpropagation）**：将评估结果沿路径向上回传，更新统计信息

**2.4 训练机制设计**

**2.4.1 自我对弈数据生成**

训练数据完全来自系统的自我对弈：

* 每步对弈都记录当前状态、MCTS输出的概率分布和最终胜负结果
* 胜负标签采用对称设计：赢家的所有状态标记为+1，输家标记为-1
* 这种设计确保网络学习到正确的价值判断

**2.4.2 损失函数设计**

训练使用复合损失函数：

Loss = CrossEntropy(策略输出, MCTS概率分布) + MSE(价值输出, 胜负标签)

**策略损失**：使用交叉熵损失，让网络学习模仿MCTS的决策分布 **价值损失**：使用均方误差损失，让网络准确预测局面胜负

**2.4.3 训练流程控制**

* **数据缓存机制**：使用固定大小的缓存池存储最新的训练数据
* **批量训练**：每积累一定数量的对弈局数后进行批量训练
* **学习率调度**：采用指数衰减调度器，随训练进行逐步降低学习率
* **检查点管理**：定期保存模型状态，支持训练中断后的恢复

**2.5 实验验证系统**

项目还实现了完整的对弈验证系统：

* **Web界面**：提供直观的人机对弈平台
* **移植网络支持**：可加载外部预训练模型进行效果对比
* **多线程处理**：支持并发的对弈请求处理
* **实时反馈**：提供训练过程的实时日志和性能监控

这种完整的实验环境确保了算法效果的可验证性和实用性。

**实验结果**

**图表, 气泡图

AI 生成的内容可能不正确。**

可以看到刚刚开局的时候，白棋的应对相对比较盲目，落子在一个与黑棋距离较远的地方

**图表, 散点图, 气泡图

AI 生成的内容可能不正确。**

黑棋下在了原来的相邻位置，白棋也在有意识的连成一线

**图表, 散点图, 气泡图

AI 生成的内容可能不正确。**

当黑棋连成三颗的时候，白棋发现了威胁并采取了正确的应对措施

**图表, 气泡图

AI 生成的内容可能不正确。**

后续白棋还尝试主动发起进攻，以及在黑棋斜对角连成一线的时候及时进行了围堵