基于 NumPy 手写深度强化学习框架的迷宫 AI 设计

黄鹤翔高恒斌都奕宁
信息与电子工程学院
浙江大学, 杭州, 中国

摘要—本报告详细阐述了一个基于深度强化学习（Deep Q-Network, DQN）的迷宫自动寻路及对战 AI 系统的设计与实现。项目的核心挑战在于不依赖现有的深度学习框架（如 PyTorch 或 TensorFlow），而是仅使用 NumPy 库从底层构建卷积神经网络（CNN）及优化器。报告首先介绍了迷宫生成算法（DFS 与 Prim），随后深入探讨了自定义深度学习框架 `simple_nn` 的实现细节，包括 `im2col` 卷积加速、反向传播算法及 Adam 优化器。在此基础上，搭建了 DQN 模型，设计了包含墙壁信息、位置信息及访问记录的 7 通道状态空间，并通过经验回放（Experience Replay）和目标网络（Target Network）机制稳定训练过程。最后，展示了 AI 在迷宫寻路测试及人机对战模式中的表现，验证了手写框架及算法的有效性。

***Index Terms*—深度强化学习，DQN，卷积神经网络，NumPy, 迷宫生成, 自动寻路**

I. 引言

项目的设计灵感源于 B 站 up 河南葫季果和 L_Shy_P 的视频，视频中出现了一个近乎无敌的 ai，能够追踪敌人躲避子弹。

本项目旨在旨在构建一个具备自主学习能力的坦克对战 AI 系统。与常规项目不同，本项目的核心要求是 “去框架化”，即不调用成熟的深度学习库，而是利用 Python 和 NumPy 从零实现神经网络的底层算子（卷积、全连接、激活函数）及训练机制。这不仅考验对强化学习算法的理解，更要求对神经网络反向传播及矩阵运算有深入的掌握。

II. 设计任务和要求

本项目的具体设计任务和要求如下：

1) **迷宫环境构建**：实现基于深度优先搜索（DFS）和 Prim 算法的随机迷宫生成器，支持可视化展示。
2) **底层框架实现**：仅使用 NumPy 库，实现卷积层（Conv2d）、全连接层（Linear）、激活函数（ReLU）、损失函数（MSELoss）及优化器（Adam）。
3) **AI 算法实现**：基于手写框架，在简单的环境中搭建 DQN 框架训练 ai，在复杂的环境中使用遗传算法迭代 ai。
4) **人机对战模式**：集成训练好的模型，实现玩家与 AI 在迷宫中的实时对战（坦克大战模式）。

III. 算法原理

A. 迷宫生成算法-*DFS*

DFS 迷宫生成算法本质上是一种递归回溯过程。它从起始格子出发，随机选择一个未访问的相邻格子进行 “打通”（移除中间墙壁），然后递归地对该新格子执行相同操作。当当前格子的所有邻居均已被访问时，算法回退至上一层，直至遍历完整个网格。

- 将每个网格单元视为图中的一个节点；
- 初始时所有节点标记为 “未访问”，所有相邻节点间存在 “墙”；
- 从任意起点（如左上角 (0,0)）开始 DFS 遍历；
- 每次移动到新节点时，移除当前节点与目标节点之间的墙，并将目标节点加入连通图；
- 递归完成后，整个网格形成一棵生成树，对应一个无环连通迷宫。

B. 迷宫生成算法-*prim*

Prim 算法原本用于求解最小生成树（Minimum Spanning Tree, MST）。在迷宫生成中，我们将其改造为随机 Prim 算法：不考虑边权，而是随机选择待扩展的边。

算法维护一个“边界集合”（frontier set），包含所有与已生成区域相邻但尚未加入的格子。每一步从边界集中随机选取一个格子，将其与已生成区域中的某个邻居连接（打通墙壁），并将该格子的新邻居加入边界集。重复此过程直至覆盖全部格子。

Prim 生成的迷宫通常具有更多短分支和较短的主路径，整体结构更“开阔”，有利于测试智能体在多岔路口的决策能力。

- 将网格视为完全图，每条潜在通道（相邻格子间）是一条边；
- 任选一个起始格子加入生成树；
- 将其所有相邻格子加入“边界集”；
- 循环：从边界集中随机选一格子，随机选择其一个已在生成树中的邻居，打通两者之间的墙，并将该格子纳入生成树，同时将其未访问邻居加入边界集；
- 直至所有格子被纳入。

DFS 生成的迷宫通常具有长而曲折的走廊和较少的分支,Prim 生成的迷宫通常具有更多短分支和较短的主路径

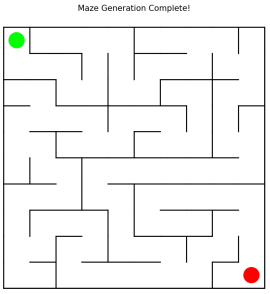

图 1: DFS 生成迷宫

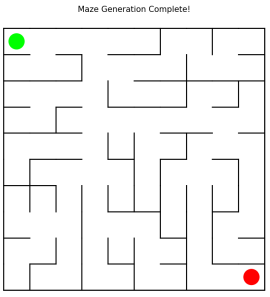

图 2: prim 生成迷宫

C. 解迷宫算法-A*

A* （A-Star）算法是一种广泛应用于路径规划和图遍历的启发式搜索算法。它结合了 Dijkstra 算法的完备性与贪心最佳优先搜索的效率，能够高效地找到从起点到终点的最短路径。

A* 算法通过评估函数 f(n) 来决定下一个要探索的节点：

$$f(n) = g(n) + h(n)$$

其中：

- g(n)：从起点到当前节点 n 的实际代价。
- h(n)：从当前节点 n 到目标点的启发式估计代价。

在本项目中使用曼哈顿距离 $abs(x_1 - x_2) + abs(y_1 - y_2)$ 进行启发式搜索。我们在初始化时将起点 (0,0) 加入 open 列表中，然后进入循环，直到 open 列表为空或找到终点为止。在每一步迭代中，我们从 open 列表中选择 f 值最小的节点作为当前节点。然后检查当前节点是否为目标节点，如果是则结束循环；如果不是目标节点，则将其标记为已处理并将其相邻的未处理节点加入 open 列表中。同时，将已处理的节点加入 closed 列表中以避免重复处理。

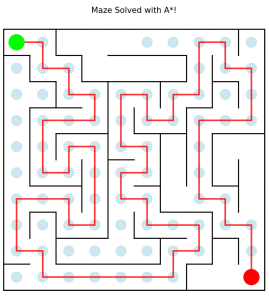

图 3: A* 算法解迷宫

D. 自定义深度学习框架 *(simple_nn)*

为了替代 PyTorch，我们实现了 `simple_nn.py`。

1) 卷积层实现: 卷积操作的核心在于高效地提取局部特征。为了避免低效的多重循环，采用了 `im2col`（image to column）技术，将输入特征图展开为矩阵，从而将卷积运算转化为矩阵乘法（GEMM）。

$$Y = W_{col} \times X_{col} + b \tag{1}$$

其中 X_{col} 是展开后的输入矩阵，W_{col} 是重排后的权重矩阵。

2) 优化器: 实现了 Adam 优化器，结合了动量法（Momentum）和 RMSProp 的优点，自适应调整学习率。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{4}$$

E. AI 训练-DQN

DQN (Deep Q-Network) 将卷积神经网络与 Q-Learning 结合，解决了高维状态空间的强化学习问题。

1) 核心机制:

- 经验回放 (Experience Replay)：构建一个回放池 D，存储转移样本 $(s, a, r, s', done)$。训练时随机采样一个小批量（Batch），打破了数据间的相关性，提高了训练的稳定性。
- 目标网络 (Target Network)：引入一个参数滞后的目标网络 $\hat{Q}$ 计算目标值，避免了“自举”（Bootstrapping）导致的目标值震荡。目标网络每一定时间从策略网络中复制参数。

通过贝尔曼最优方程来计算期望 Q 值，即执行该动作后获得的即时奖励 r 加上对未来最大 Q 值的折扣期望

$$Q^*(s,a) = \mathbf{E} * s' \sim \mathcal{P}\left[r + \gamma \max * a' Q^*(s', a')\right]$$

目标函数为最小化均方误差：

$$L(\theta) = \mathbf{E}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{5}$$

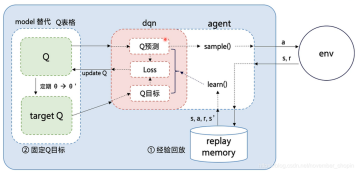

图 4: dqn 算法流程结构

2) Q 值估计网络设计-输入层: 输入层设计为 $7 \times H \times W$ 的张量，包含 7 个通道：

- 通道 0-3 (墙壁信息)：分别编码每个格子上、下、左、右四个方向的墙壁存在情况
- 通道 4-5 (位置信息)：分别使用 One-hot 编码智能体位置和目标位置
- 通道 6 (历史轨迹)：记录智能体访问过的路径，赋予智能体记忆

3) Q 值估计网络设计-卷积层: 网络包含 4 层卷积层，通道数分别为 32, 64, 128, 128，主要为卷积层与 relu 激活层的循环。网络中没有加入传统 CNN 网络中常见的池化层，主要是因为池化层的加入会模糊智能体与目标点的位置信息，影响训练效果。

通过堆叠 4 层卷积层，深层神经元的感受野逐渐扩大，能够感知更大范围的迷宫结构，从而规划长距离路径。通道数逐层增加（$32 \to 128$），使得网络能够组合低级几何特征.

表 I: 卷积神经网络各层参数配置

	输入通道数	输出通道数	卷积核大小	步长	填充
第一层	7	32	3	1	1
第二层	32	64	3	1	1
第三层	64	128	3	1	1
第四层	128	128	3	1	1

4) Q 值估计网络设计-全连接层: 卷积层输出展平后，连接 3 层全连接层（$1024 \to 512 \to 4$），提供了强大的非线性拟合能力，将提取的迷宫特征转化为上、下、左、右四个离散动作的 Q 值。

F. 超参数设计

```python
# Hyperparameters
WIDTH, HEIGHT = 10, 10
EPISODES = 10000
BATCH_SIZE = 32
GAMMA = 0.9 # 对未来奖励的关心程度，越大越关心
# 使用线性下降的EPSILON，EPSILON越大越倾向于随机探索
EPSILON_START = 1.0
EPSILON_END = 0.05
EPSILON_DECAY_EPISODES = 4000
LR = 0.0001
TARGET_UPDATE = 200 #目标网络的更新频率
MEMORY_SIZE = 50000 #经验池的大小
```

G. AI 训练-遗传算法

遗传算法是一种受达尔文生物进化论启发的元启发式优化算法。它模拟了自然界“物竞天择，适者生存”的过程，通过在解空间中维护一个种群（Population），并利用选择（Selection）、交叉（Crossover）和变异（Mutation）等遗传算子，迭代地演化出越来越优的解决方案。

在这个项目中，我们使用一个全连接神经网络来对智能体的下一步动作进行决策，通过遗传算法迭代更新神经网络的参数。同时在训练过程中，我们借鉴了 DQN 的训练思路，每隔一段时间将训练目标的参数从智能体中复制下来，以求获得更加强大的智能体。

- 选择算子：锦标赛选择
- 交叉算子：均匀交叉
- 变异算子：高斯扰动
- 进化策略：精英保留，将每一代中适应度最高的前 10% 个体（精英）直接复制到下一代，不经过交叉和变异

IV. 主要仪器设备

- **开发语言**：Python 3.10
- **核心库**：NumPy (用于矩阵运算), Matplotlib (用于可视化)
- **硬件环境**：AMD CPU
- **操作系统**：Linux

V. 设计结果记录与分析

VI. 结论

VII. 参考文献

1) https://www.bilibili.com/video/BV17EswzLEKw/