基于强化学习的AI贪吃蛇

引言：近年来，强化学习（Reinforcement Learning, RL）作为一种重要的机器学习方法，受到了广泛的关注和研究。随着深度学习技术的快速发展，强化学习在多个领域取得了显著的进展，包括游戏、机器人控制、自动驾驶和自然语言处理等。特别是在2013年，DeepMind提出的深度Q网络（DQN）算法，使得强化学习在复杂环境中的应用成为可能，标志着这一领域的一个重要里程碑。本文将基于强化学习实现AI贪吃蛇。

1. 实验原理介绍

1.1 强化学习介绍

强化学习（Reinforcement Learning, RL）是人工智能（AI）和机器学习（ML）领域的一个重要子领域，与监督学习和无监督学习并列。 它模仿了生物体通过与环境交互来学习最优行为的过程。 与传统的监督学习不同，强化学习没有事先标记好的数据集来训练模型。

强化学习主要由智能体（Agent）、环境（Environment）、状态（State）、动作（Action）、奖励（Reward）组成。

智能体：强化学习的本体，作为学习者或者决策者。

环境：强化学习智能体以外的一切，主要由状态集合组成。

状态：一个表示环境的数据，状态集则是环境中所有可能的状态。

动作：智能体可以做出的动作，动作集则是智能体可以做出的所有动作。

奖励：智能体在执行一个动作后，获得的正/负反馈信号，奖励集则是智能体可以获得的所有反馈信息。

本实验采用强化学习中的Q-Learning算法，现对Q-Learning进行介绍

1.2 Q-Learning算法介绍

Q-Learning是Value-Based的强化学习算法，Q-value是由State和Action组合在一起决定的。实际的项目中我们会存储一张表，我们叫它Q表。key是(state, action)，value就是对应的Q-value。每当agent进入到某个state下时，我们就会来这张表进行查询，选择当前State下对应Value最大的Action，执行这个action进入到下一个state，然后继续查表选择action，这样循环。Q-Value的价值就在于指导Agent在不同state下选择哪个action。

其中的重点是更新Q表中的Q-value值，这里用到了Bellman Equation（贝尔曼方程）



图 1 贝尔曼方程

根据贝尔曼方程即可更新Q表中的Q-value值

1. Learning算法流程

输入γ和α

初始化：Q表所有值都设置为0

过程：

agent随机选择一个state

如果state是最终结束的状态：

End

否则：

在当前state下，选择可以使Q(s,α)最大的α值

系统返回reward；

根据贝尔曼方程更新Q表中的Q-value值

继续循环

1.3 epsilon-greedy策略介绍

epsilon-greedy策略以epsilon的概率随机选择动作，以1-epsilon的概率选择模型预测的最优动作，平衡探索和利用。

1. 实验重要代码介绍
   1. model.py介绍

model.py 文件定义了一个神经网络模型和一个用于强化学习代理的训练器。

Linear\_QNet神经网络定义了一个带有一个隐藏层的简单前馈神经网络。

QTrainer训练器管理神经网络的训练过程。

* 1. agent.py介绍

agent.py 文件定义了一个强化学习代理（Agent），用于训练和控制贪吃蛇游戏中的蛇。

Agent类中定义了许多函数：

get\_state(self, game): 获取当前游戏状态，返回一个11维的状态向量。

remember(self, state, action, reward, next\_state, done): 将当前的状态、动作、奖励、下一个状态和是否结束的信息存储到记忆队列中。

train\_long\_memory(self): 从记忆队列中随机抽取一批样本进行批量训练。

train\_short\_memory(self, state, action, reward, next\_state, done): 使用当前的状态、动作、奖励、下一个状态和是否结束的信息进行单步训练。

get\_action(self, state): 根据当前状态获取下一步动作，使用epsilon-greedy策略平衡探索和利用。

* 1. game.py介绍

game.py 文件定义了一个使用 Pygame 库的贪吃蛇游戏。

其中定义了游戏的基本属性和SnakeGameAI类。

SnakeGameAI类中定义了许多函数：

reset(self): 重置游戏状态，包括蛇的位置、方向和食物的位置。

\_place\_food(self): 随机生成食物的位置，并确保食物不生成在蛇身上。

play\_step(self, action): 执行一步游戏逻辑，包括移动蛇、检测碰撞、更新界面和返回奖励、游戏是否结束和当前分数。

is\_collision(self, pt=None): 检测是否发生碰撞（撞到边界或蛇身）。

\_update\_ui(self): 更新游戏界面，包括绘制蛇和食物、显示分数。

\_move(self, action): 根据动作更新蛇的方向和位置。

* 1. train.py介绍

train.py 文件定义了一个用于训练AI贪吃蛇的脚本。

函数 plot\_scores绘制分数曲线。

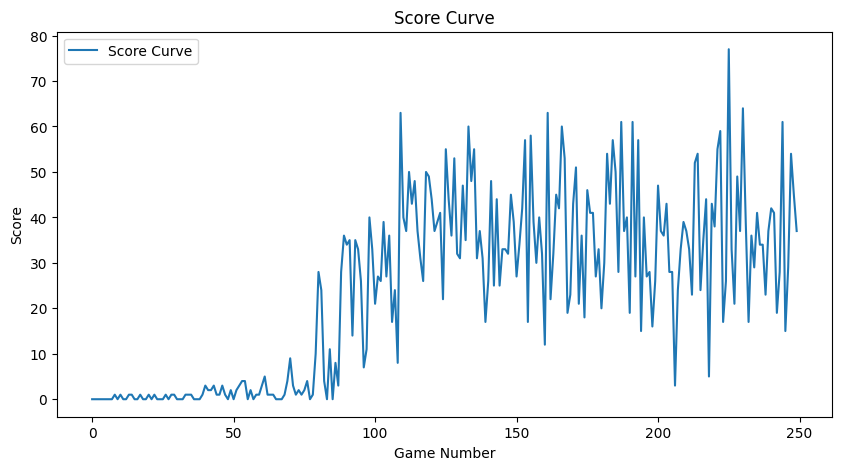
函数 train训练AI贪吃蛇。

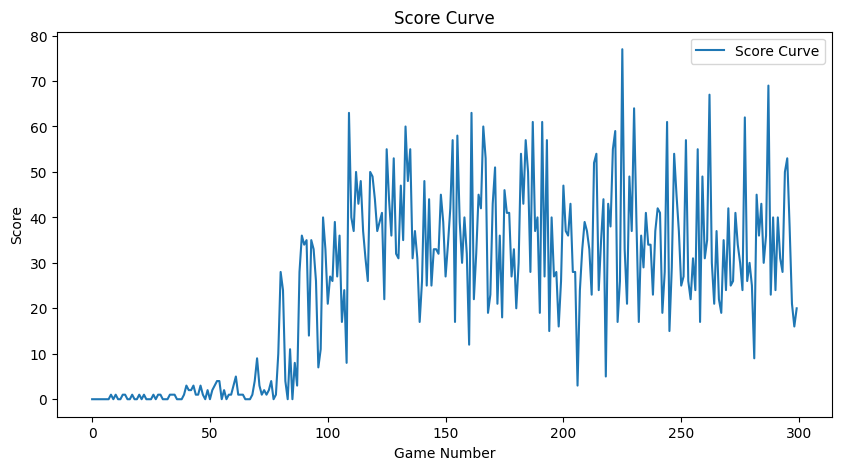
* 1. AI\_snake.py介绍

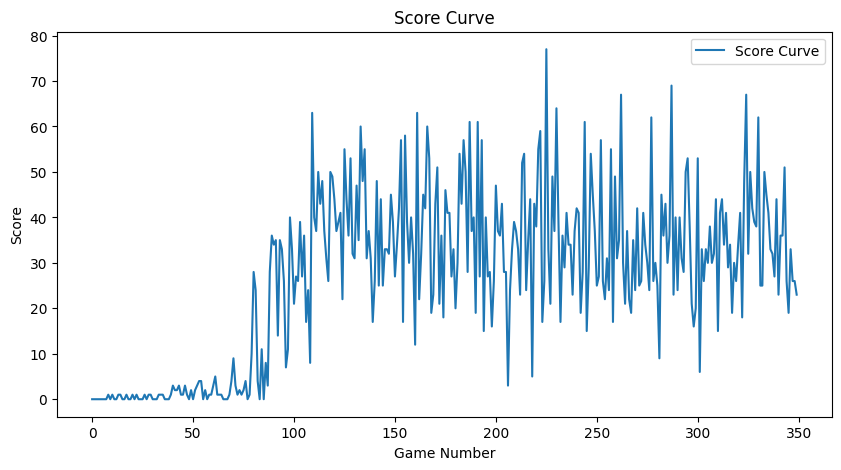
AI\_snake.py 文件定义了一个用于运行AI贪吃蛇游戏的脚本。

调用之前训练得到的表现最好的模型进行贪吃蛇游戏。

1. 实验结果







最好分数为77