# 基于 VizDoom 的 FPS 智能体实战

**核心考察点** ：卷积神经网络 (CNN) 视觉特征提取、循环神经网络 (LSTM) 时序记忆处理、PPO/DQN 算法的工程实现、奖励函数设计 (Reward Shaping) 以及 Docker 容器化交付能力。

## 一、 任务内容

本次作业要求每位同学完成一个完整的**端到端强化学习系统**，从原始图像输入到智能决策输出，最终交付一个能够在复杂迷宫中生存并击杀敌人的 AI 模型。

### 核心任务：构建具备“视觉+记忆”能力的 FPS 智能体

#### 1. 环境构建与数据流 (Environment & Data Pipeline)

* **环境封装 (Gym Wrapper)** ：
  + 编写适配器类，将 VizDoom 的底层 C++ 接口封装为标准的 Gym/Gymnasium 接口（reset(), step()）。
  + **数据清洗**：实现图像预处理流水线，将原始 320x240 RGB 屏幕图像裁剪、灰度化、降采样至 84x84 单通道张量。
  + **动作空间映射**：将离散动作（如：[0,0,1]）映射为游戏内的具体指令（如：TURN\_LEFT）。

#### 2. 网络架构设计 (Model Architecture)

* **视觉感知层 (CNN)** ：
  + 设计并实现基于 CNN 的特征提取器（推荐参考 NatureCNN 或 ImpalaCNN 结构），从预处理后的图像中提取空间特征。
* **时序记忆层 (LSTM/GRU)** ：
  + **强制要求**：在全连接层前必须引入 **RNN (LSTM 或 GRU)** 单元，用于处理 POMDP（部分可观测）问题。
  + **状态管理**：在训练循环中正确处理 Hidden State 的初始化与传递，确保智能体能“记住”刚才看到的敌人或物资。

#### 3. 策略训练与算法实现 (Training & Algorithm)

* **场景指定**：训练必须在 **Deathmatch (死斗模式)** 或 **Health Gathering Supreme (迷宫寻宝)** 复杂场景下进行。
* **功能需求 1：算法实现**。基于 PyTorch 实现 PPO (Proximal Policy Optimization) 或 Dueling DQN 算法。
* **功能需求 2：奖励函数设计 (Reward Engineering)**。
  + 设计合理的稠密奖励（Shaping Reward），解决稀疏奖励导致的难以收敛问题。
  + *示例*：除了击杀奖励外，需设计“移动奖励”（防止蹲坑）、“拾取奖励”或“朝向敌人奖励”。

#### 4. 系统工程与可视化 (System & Visualization)

* **训练监控**：打通 TensorBoard 或 WandB，实时记录并展示 Reward Mean (平均奖励)、Value Loss、Entropy (策略熵) 等关键指标曲线。
* **模型持久化**：实现自动 Checkpoint 机制，每隔固定 Episode 保存模型权重，并支持从断点恢复训练。
* **结果可视化**：编写评估脚本，加载训练好的模型，自动录制并保存一段 **MP4 格式** 的第一人称战斗回放视频。

## 二、 评分标准 (Total: 100%)

本次作业侧重于**系统的完整性**、**算法的深度**和**工程化落地能力**。

|  |  |  |
| --- | --- | --- |
| **维度** | **占比** | **详细说明** |
| **功能完成度** | 80% | 完成从环境封装到模型收敛的全流程；智能体在指定场景下表现出显著优于随机策略的智能（如主动寻路、瞄准）。 |
| **技术深度** | 10% | **核心考点**：成功引入 LSTM/GRU 并在实验报告中通过对比实验（Ablation Study）证明其对生存率的提升；奖励函数设计逻辑严密。 |
| **工程质量** | 5% | 代码结构清晰（网络、算法、环境解耦）；显存管理合理（无内存泄漏）；Docker 镜像构建规范。 |
| **文档与分析** | 5% | 实验报告逻辑清晰，能解释训练曲线的波动原因；对“Bad Case”（失败案例）有深入的归因分析。 |

## 三、 交付与验收标准

### 1. 交付物格式

请提交一个完整的工程文件夹（压缩包），命名格式：学号\_姓名\_VizDoom\_Project.zip。

### 2. 验收标准

助教将使用以下命令进行**黑盒测试**，请确保你的环境符合标准：

cd your\_project\_folder  
# 1. 构建镜像  
docker build -t vizdoom\_final\_submission .  
# 2. 运行评估脚本 (需支持 GPU)  
docker run --gpus all -v $(pwd)/outputs:/app/outputs vizdoom\_final\_submission python evaluate.py

* **运行验收**：运行上述命令后，程序应不报错，并在控制台输出平均得分。
* **结果验收**：运行结束后，宿主机的 outputs 文件夹下应生成至少一个 .mp4 战斗回放视频和一份 eval\_log.txt。

### 3. 特别说明

* **依赖管理** ：Dockerfile 中必须包含项目所需的所有系统依赖（如 libboost, cmake 等）和 Python 库依赖，**禁止**要求助教手动安装任何环境。
* **硬件适配** ：代码需包含 device = "cuda" if torch.cuda.is\_available() else "cpu" 逻辑，具备在不同硬件上的兼容性。