Hybrid Imitation–Reinforcement Learning for Autonomous Control in Xpilot

Student: Muhammad Abdullah
Course: COM 407 - Computational Intelligence

Overview

This project aims to design an autonomous agent for the game Xpilot, capable of surviving and defeating other players by learning optimal control strategies. The proposed approach combines imitation learning and reinforcement learning (RL) in a hybrid pipeline.

The system will first train a neural network controller to imitate human gameplay (learning from recorded demonstrations). This gives the agent a solid initial policy that can perform basic maneuvers such as navigating, avoiding obstacles, and firing at enemies. The trained model will then be fine-tuned using reinforcement learning, allowing it to improve performance beyond the demonstrations by maximizing in-game rewards such as survival time and successful attacks.

Motivation

Training an AI agent entirely through reinforcement learning often requires extensive training time and careful hyperparameter tuning, as the agent starts with no knowledge and learns through trial and error.
By contrast, imitation learning provides a "head start" by teaching the agent from human examples. Combining both methods creates a balance between efficiency and adaptability the agent learns quickly from demonstrations and then improves through self-play.

This approach reflects modern AI training strategies used in complex environments such as robotics and autonomous driving, where agents first imitate and later refine their behavior through reinforcement learning.

Methodology

1. Data Collection
    - Record gameplay data from human or rule-based Xpilot sessions.
    - Each data point will include:
        - State features: ship position, velocity, angle, distances to walls/enemies, etc.
        - Action labels: thrust, rotate left/right, fire, or idle.

2. Imitation Learning
    - Train a feedforward neural network (using PyTorch) with one or two hidden layers.
    - The model learns to predict the correct action given the current state.
    - Use standard supervised learning with cross-entropy loss.
    - Evaluate accuracy on held-out demonstration data.
3. Reinforcement Learning Fine-Tuning
    - Use the trained model as the starting policy for Deep Q-Learning (DQN).
    - Define a reward function based on:
        - Positive rewards for survival time or hitting enemies.
        - Negative rewards for collisions or being destroyed.
    - Train the agent in the Xpilot environment to gradually improve beyond the imitation model.
    - Compare performance (average reward, survival duration) between:
        - The imitation-only model
        - The hybrid (imitation + RL) model
4. Evaluation and Report
    - Measure how well the hybrid model performs relative to baselines.
    - Discuss learning efficiency, stability, and observed strategies.
    - Create a short project website/report summarizing the setup, methods, results, and visuals.