Reinforcement Learning

Group 5

Bisma Haroon

Asma Saif

Abiha Farid

Abbas Mustafa

Objectives

- Train an AI agent to play street fighter game.
- Learn attacks, blocks, and combos through trial and error.
- Adapt to opponent moves (defensive/aggressive modes).
- Track wins/losses, moves, and mistakes to improve the AI.
- Make the AI challenging but not unfair for human players.

Reinforcement Learning

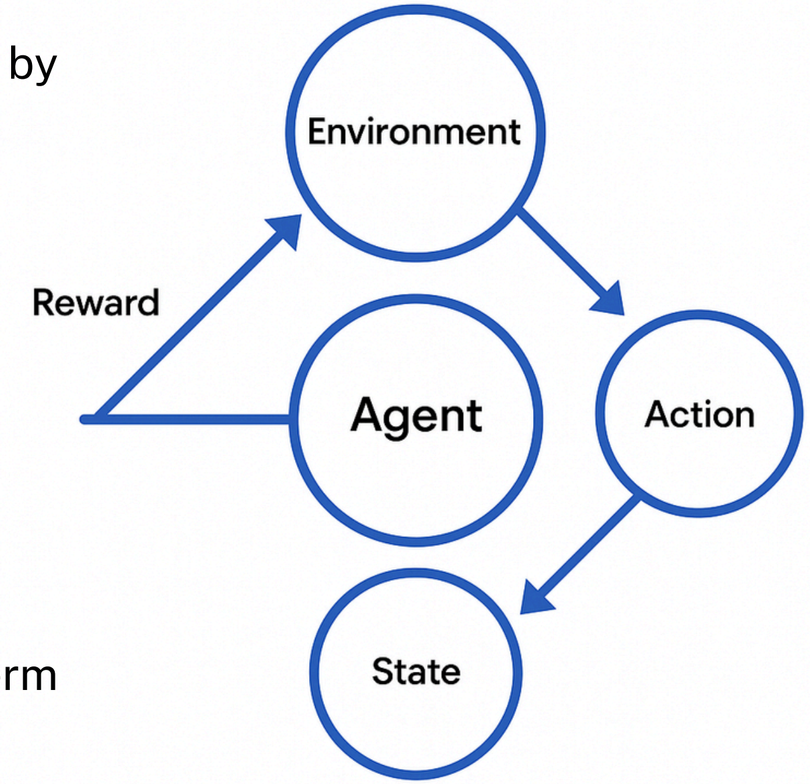
A learning paradigm where an agent learns to make decisions by interacting with an environment.

Core Elements:

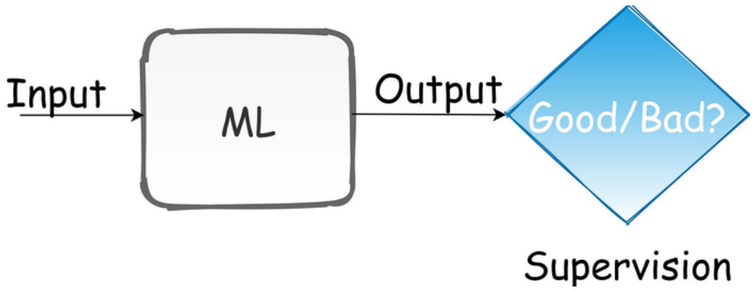
- Agent
- Environment
- State
- Action
- Reward

Goal:

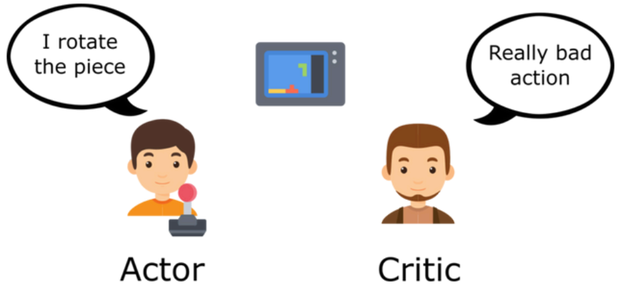
- Learn a policy that maps states to actions to maximize long-term rewards.



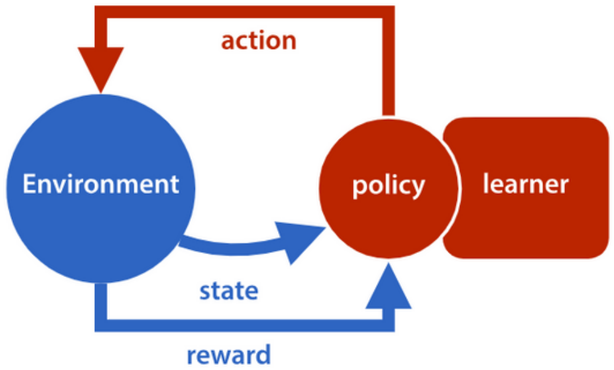
Types of RL Algorithms



Value based



Actor Critic

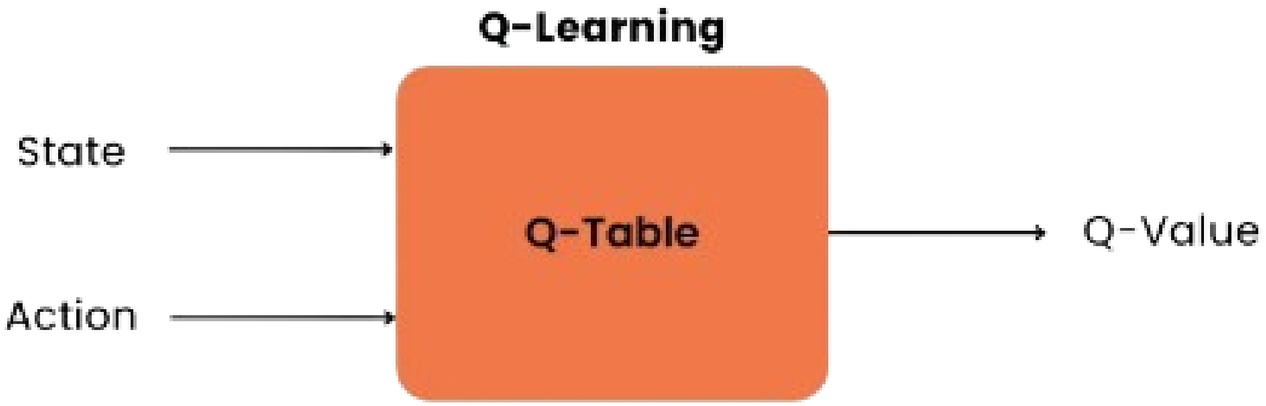


Policy based

RL Beyond Gaming

- **Robotics:** Automation control for manipulation/navigation
- **Self-Driving Cars:** Real-time decision making in dynamic environments
- **Algorithmic Training:** Adaptive strategies for market fluctuations
- **Cybersecurity:** Intelligent threat detection/response systems

Value Based Models: Q-Learning

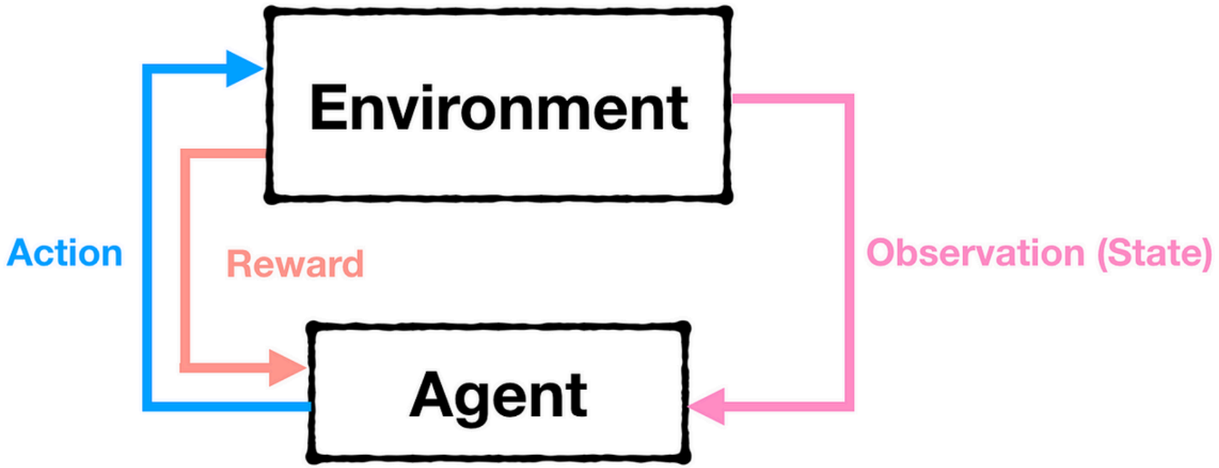


A value-based RL method:

- Learns action values (Q-table) through exploration
- Updates estimates using observed rewards + future potential
- Off-policy: Learns optimal actions regardless of behavior

Key parameters: learning rate (α), discount factor (γ)

Value Based Models: Deep Q-Network

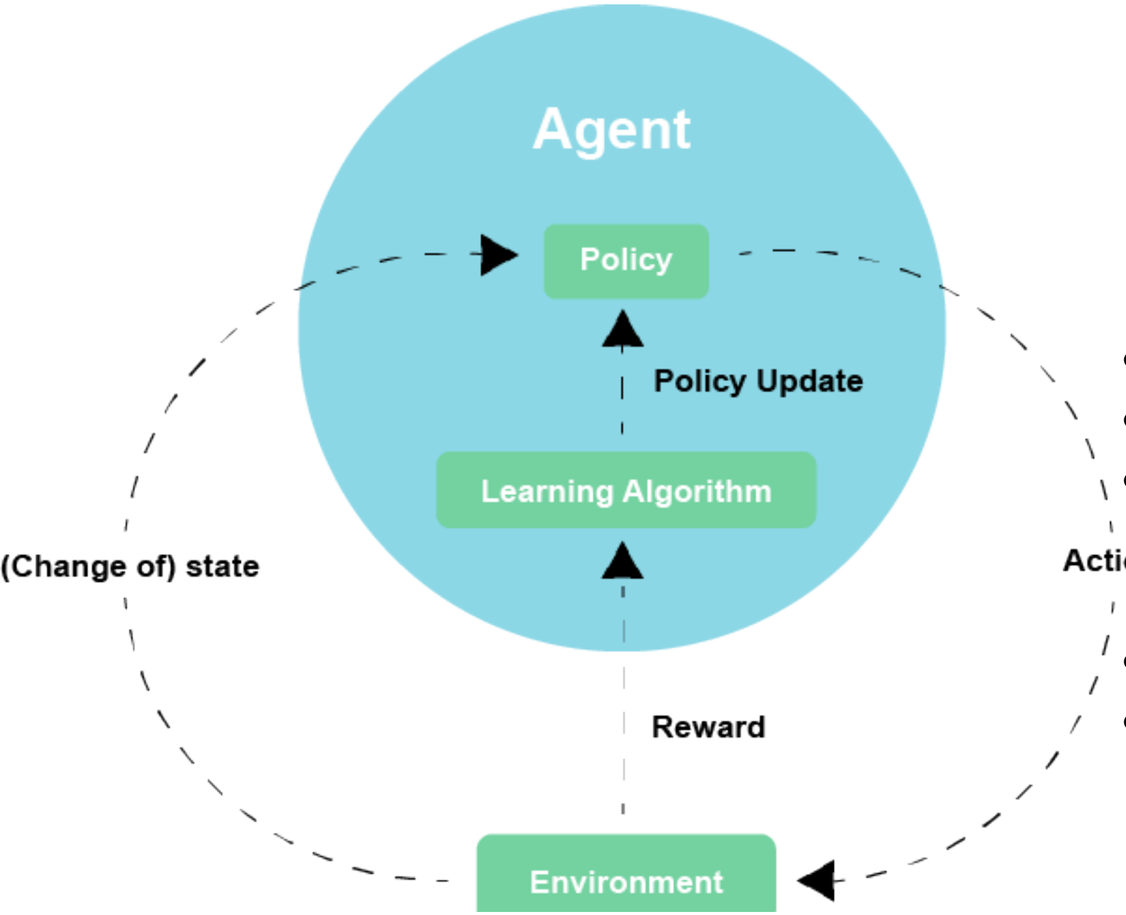


- Value-based reinforcement learning algorithm
- Uses a neural network to approximate the Q-function

Key Features:

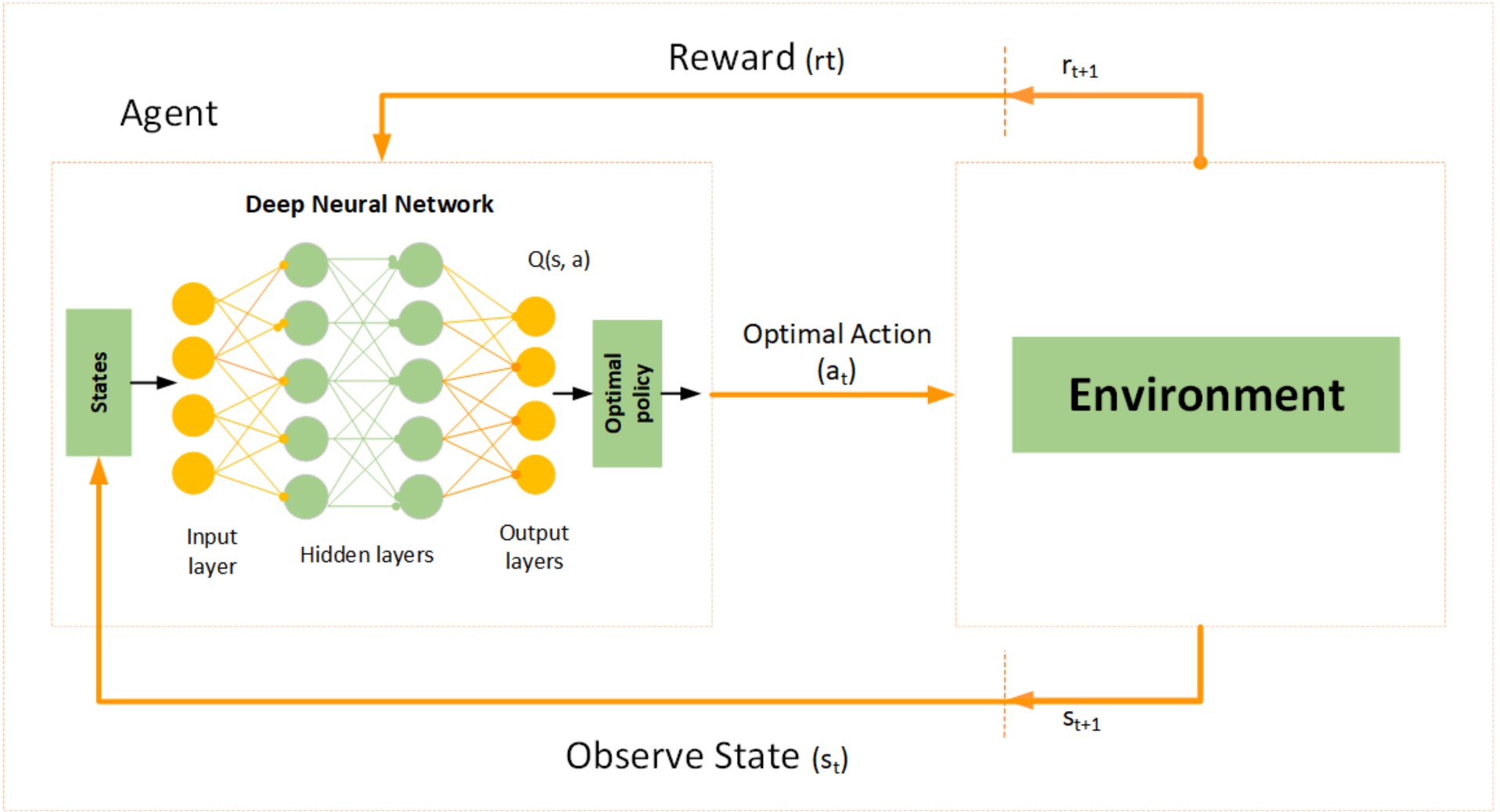
- Experience Replay
- Target Network
- Epsilon-Greedy Policy

Agent-Environment Loop



- Agent observes the current state of the environment
- Agent selects an action based on a policy
- Environment returns:
  - A reward (positive or negative)
  - The next state
- The agent uses this information to update its knowledge
- Loop repeats until an end condition is met

Technique Overview: How DQN Works

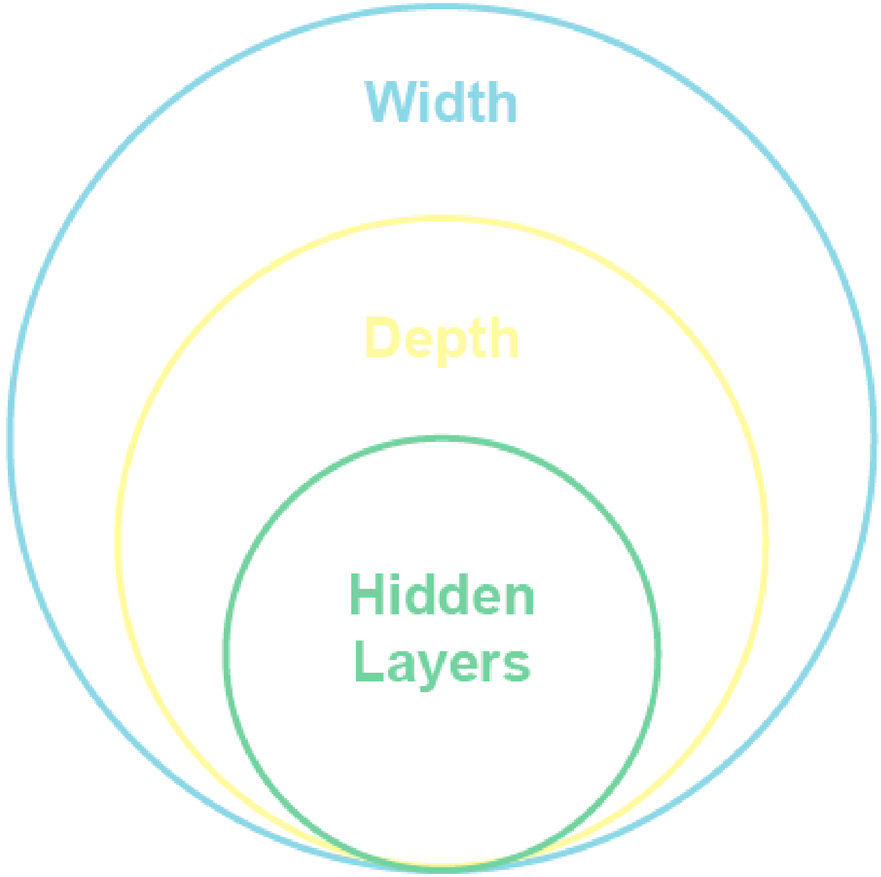


Technique Overview: DQN Structure

Neurons per layer influencing capacity

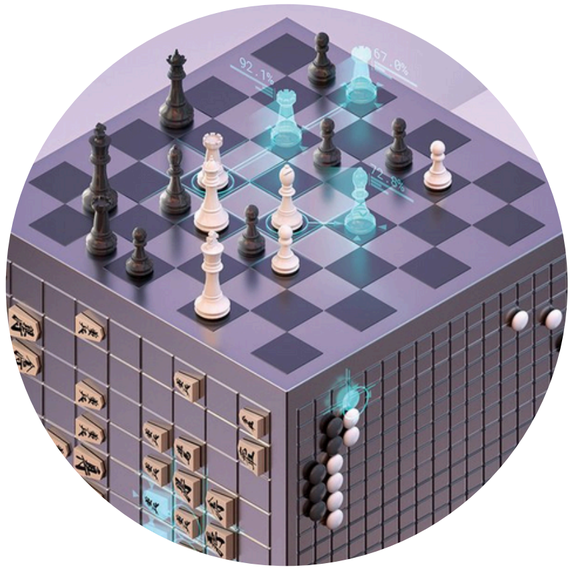
Number of layers affecting complexity

Core component for data transformation



Technique Overview: Famous Applications

Minecraft – Project Malmo



AlphaGo / AlphaZero (Go, Chess, Shogi)

Atari Games

Python Tools & Packages

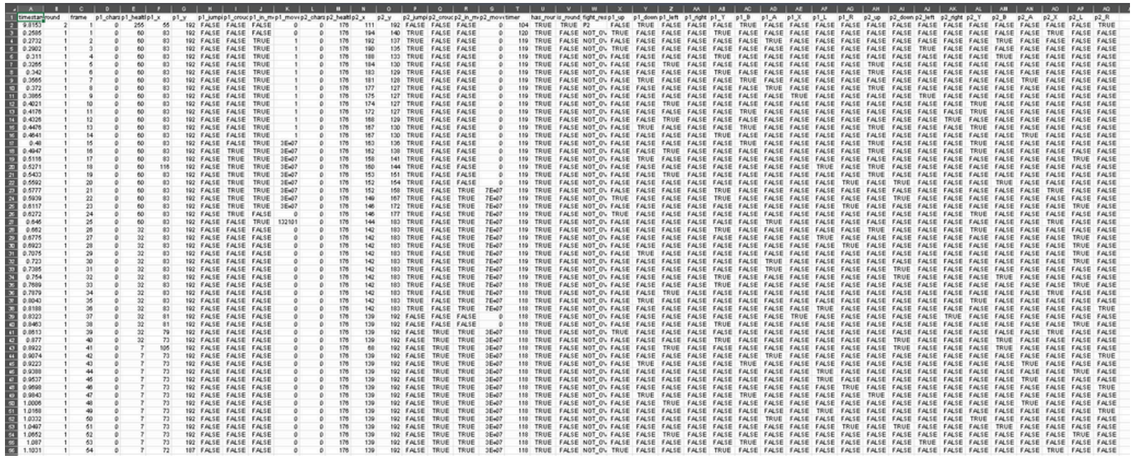
PyTorch

websockets

flake8

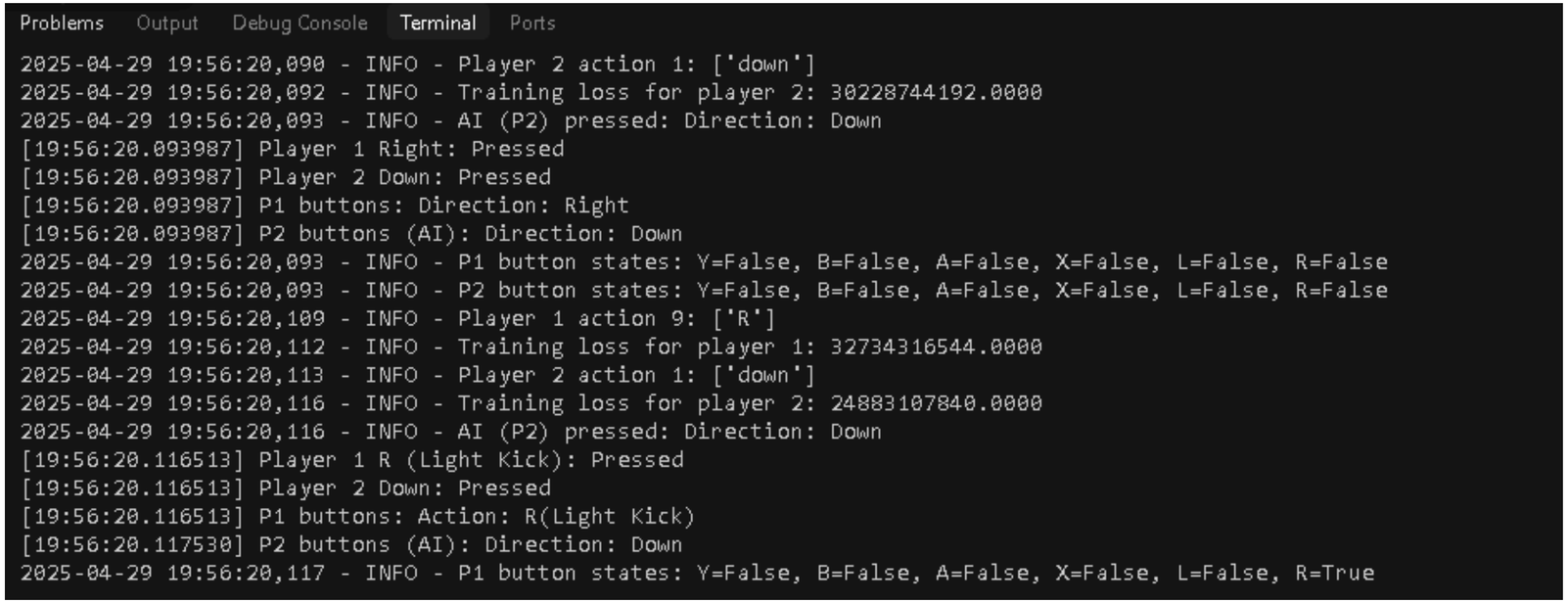
: mypy

Running Example: Dataset & Setup



- **Environment:** A 2D fighting game run in BizHawk emulator.
- **State Input:** Extracted as JSON — includes player health, position, action state, etc.
- **Actions:** Mapped to game controller buttons (e.g., jump, punch, block).

Running Example: Workflow



- **controller.py** receives game state and forwards to the DQN bot.
- **bot.py** selects action via epsilon-greedy policy.
- Action is sent back to BizHawk; results are logged by **data_recorder.py**
- Experience is stored, and the model is updated periodically

Running Example: Results

Trained agent was able to:

- Learn basic attacks and movement
- Defeat random/weak AI opponents
- Show emergent behaviors like avoiding attacks
- Performance improved significantly

Conclusion

- DQN agent effectively learned from visual input in a controlled emulator environment.
- Modular design ensures flexibility and scalability across different games.
- Future work will enhance actions, visualization, and user interaction.

Thank You