Reinforcement Learning for the Petoi Bittle in an Embedded Systems Lab

TEAM MEMBERS: TYLER INGEBRAND, AMY WIELAND, SEAN MCFADDEN, NATHAN BRUCK, YI TING LIEW, CHRIS HAZELTON, NAYRA LUJANO
SDMAY22-45 | FACULTY ADVISOR & CLIENT: DR. ROVER

Problem Statement

- Demonstrate embedded ML on a machine learning application
- Make recommendations for incorporating embedded ML in a CPRE course

Our Solution

- Train a robot to walk using reinforcement learning in virtual simulations
- Apply the neural network in real life to generate walking
- Make recommendations for course learning objectives about RL

Functional Requirements

- Reinforcement learning done virtually
- Neural network loaded on to the robot should be computationally fast enough to run at a reasonable frame rate
- Walk stably based on the neural network

Non-Functional Requirements

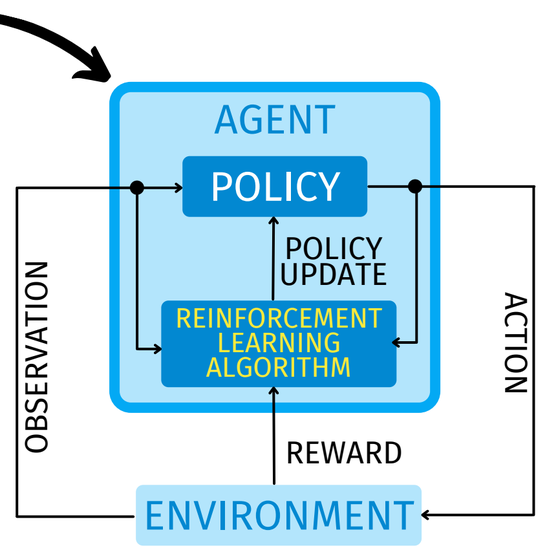
- Compile a list of useful ML resources for a class
- Modular development for maintainability
- Use C++ and Python (common languages)
- Use free tools (OpenAI Gym, PyBullet, PyTorch)

Constraints

- Limited processing power for training the neural network
- Limited sensory information provided by the Petoi Bittle
- Raspberry Pi for all real-time processing, including using the neural network

What is Reinforcement Learning?

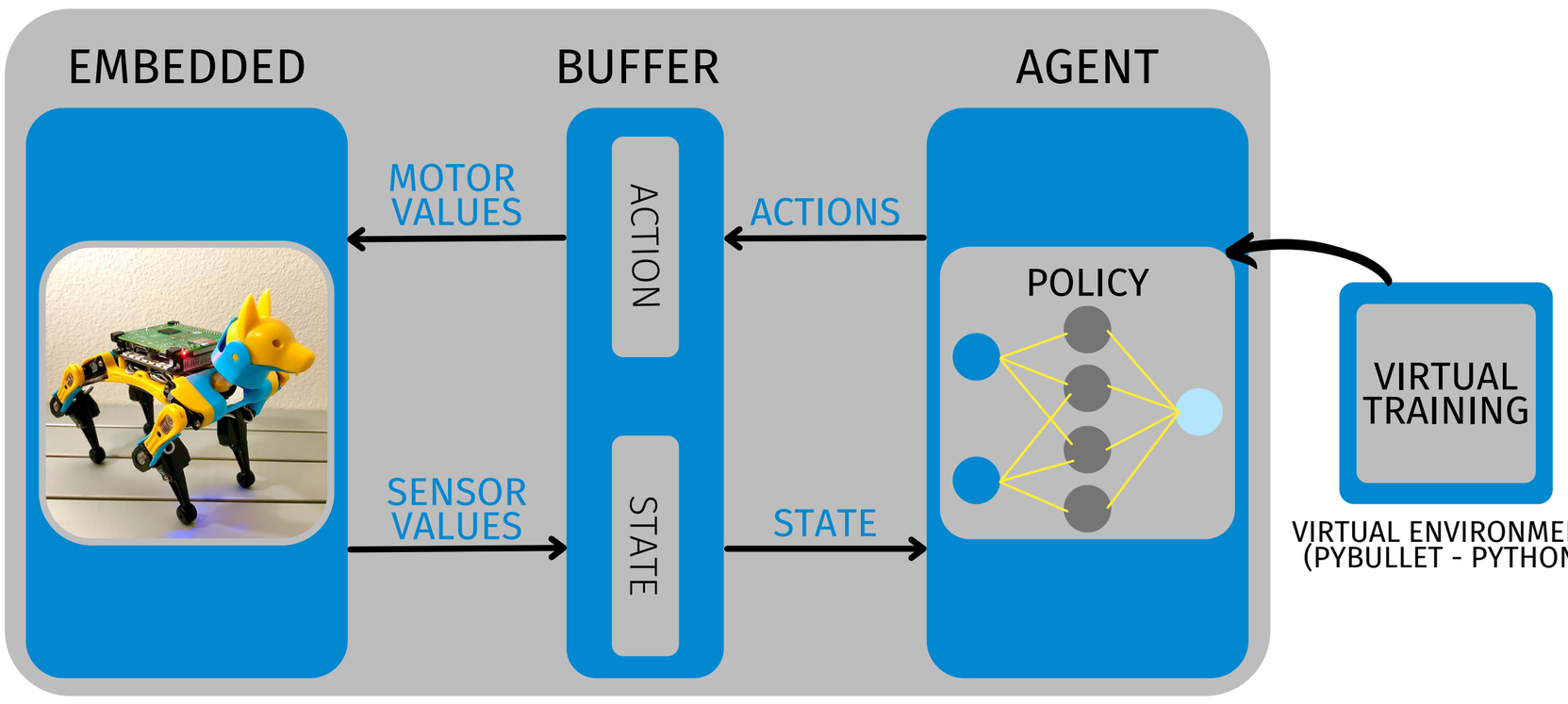
- A method of machine learning where the computer plays a game repeatedly
- Over time, it learns which choices are good and which are bad
- Eventually, it can learn to only make good decisions and “solve” the game



Intended Users & Uses

- Designed for undergraduate students and faculty
- Provide an introduction to a growing field of machine learning for future professionals

Design Approach



Operating Environment

- Training and testing are done on a smooth, flat surface
- More complicated surfaces could be learned given more sophisticated sensors

Security Concerns

- Robot falling and breaking its parts
- Failsafe in the code that would turn off the robot in the case of malicious actions

Technical Details

VIRTUAL TRAINING

- Train in Python using PyBullet, a free physics simulator
- Uses DDPG and TD3 (RL algorithms) for learning, and PyTorch for the neural networks
- Outputs a neural network capable of stable walking in simulation
- Lab module on reward shaping

AGENT

- Load a PyTorch neural network trained in Python via LibTorch library
- Given the robot’s current state, output an action calculated via the neural network
- Lab module on using agents to interact with the embedded system

EMBEDDED SYSTEM

- Interfaces the C++ code to the robot’s physical control
- Reads the sensors to create the robot’s state
- Applies an action to change the robot’s servo motor positions
- Limited to joint angle values
- Lab module on sensor noise reduction

Standards

- *IEEE P2940* - Standard for measuring robot agility
- *IEEE P1872.2* - Standard for autonomous robots ontology
- *IEEE 1725-2021*- Standard for rechargeable batteries
- *IEEE 802.11* - Standard for wireless LANs
- I2C Bus Protocol

Testing & Results

- UNIT TESTING:
  - Each module has unit tests to ensure individual functionality
- INTEGRATION TESTING:
  - “Dummy” modules can implement the interfaces to support testing and integration
- SYSTEM TESTING:
  - Stable walking in simulation
  - Reasonable motion on the testing stand
  - Unstable walking when supporting its own weight

Next Steps

- Accelerate development cycle by using a GPU for actual use in a lab setting
- Improve physics environment which leads to better performance in real life
- Design a robot for reinforcement learning from the ground up
- Further develop example labs which teach basic RL concepts