Rogues Gallery – OpenAI Gymnasium Team

Sri Ranganathan, Tejeshwar, Rahul, Mugdha, Sanjana, Chacko, Han, Dorsa

Introduction

The Neuromorphic Team at Rogues Gallery specializes in neuromorphic, also known as brain-inspired, applications. We have explored Q-Learning previously and currently working on Deep Q-learning models for the Mountain car environment - a classic problem in reinforcement learning provided by the OpenAI gymnasium library to demonstrate the effectiveness of our models. Our experimentation and results show that Deep Q-Learning converges much faster and is more consistently performant in learning the optimal policies than Q-Learning.

Methodology

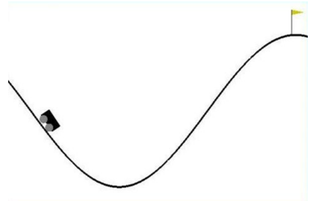
Deep Q-Learning

- An extension of Q-Learning that combines reinforcement learning with deep neural networks to handle complex, high-dimensional state spaces.
- Employs a Q-Network (deep neural network) to approximate the Q-value function
- Experience Replay: During training, the agent's experiences are stored in a replay buffer. Mini-batches of experiences are randomly sampled from the buffer to train the Q-Network, improving stability and reducing correlations between samples.
- Target Network: To stabilize learning, a separate target network is used to calculate the target Q-values. The target network's weights are periodically updated with the main Q-Network's weights, helping to mitigate the issue of moving targets.

Key Experiments & Results

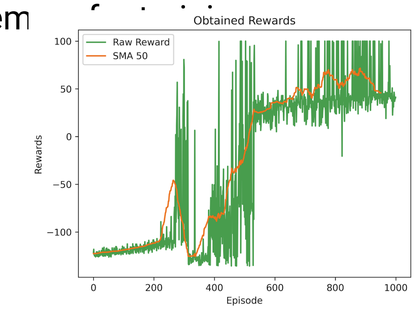
Mountain Car Problem

- The mountain car environment requires a car (agent) with insufficient power to ascend the steep hill to reach the flag.
- The only available actions for the car are accelerations on either directions.
- The car must learn optimal actions to strategically enough momentum and reach the flag placed on top of the right hill.



Model Optimizations

- Implemented an epsilon-greedy exploration strategy to balance exploration and exploitation. The value of epsilon decays over time, leading to explore more initially and exploit learned knowledge later.
- Clipped the gradients to a maximum norm to ensure stable updates of the network's weights.
- Implemented experience replay using deques to store transitions and randomly sampled mini-batches from the replay mem
- Fine-tuned training hyperparameters to improve stability and faster convergence of the Deep Q Network.



Challenges & Next steps

Challenges

- Encountered difficulties in rendering the environment due to reliance on outdated documentation for the OpenAI Gym framework instead of the updated Gymnasium version.
- Training took erroneously wrong since most development was done locally; network didn't converge since the architecture wasn't complex enough to cover the complexities of the Mountain Car environment.

Next steps: Neuromorphic implementation

- Learn about Lava, a software framework designed for neuromorphic computing. Evaluate technical challenges of mapping the Deep Q Learning model and network architecture to an Spiking Nueral Network (SNN) using Lava's programming constructs.
- Explore scalability of Lava implementation to other gymnasium environments.

Rogues Gallery – NVIDIA Jetson Team (Spring 2024)

Srijan Ponnala, James White, Suzan Manasreh

Introduction & Goals

Introduction

The Neuromorphic Team specializes in neuromorphic, also known as brain-inspired, computing applications. This semester, we've continued exploring the implementation and performance of machine learning algorithms for object detection on the NVIDIA Jetson platform.

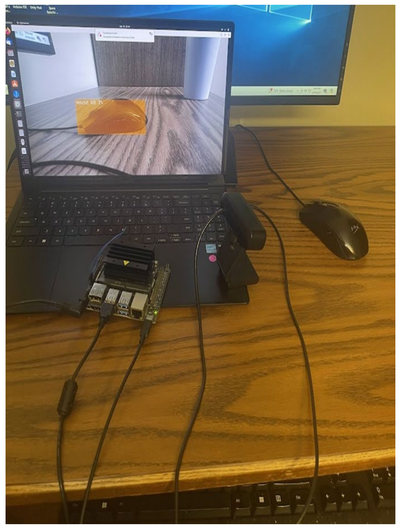
Semester Goals

- Benchmark machine learning algorithms on the NVIDIA Jetson platform and compare their run-times to traditional hardware.
- Utilize YOLO v5 and DetectNet for training precise and specialized object detection models

Key Concepts & Results

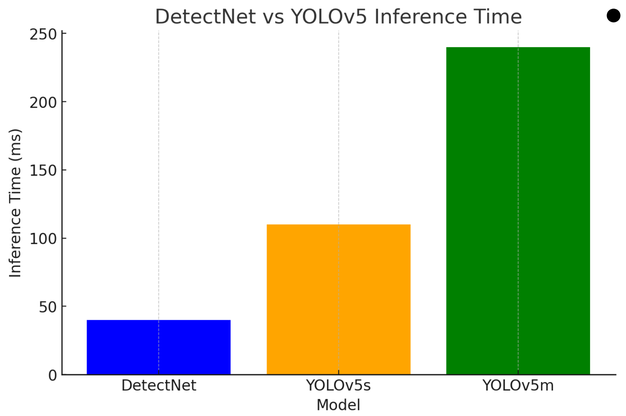
Overview

- The Nvidia Jetson platform allows for use of NVIDIA TensorRT, which is specifically designed to take full advantage of the Jetson chip's architecture
- TensorRT implements GPU acceleration for machine learning algorithms, which makes real-time object detection possible, even on the Jetson nano (the smallest and weakest chip)
- We found the benefits of using DetectNet vs YOLO v5 and measured metrics depicting these differences



DetectNet vs YOLO

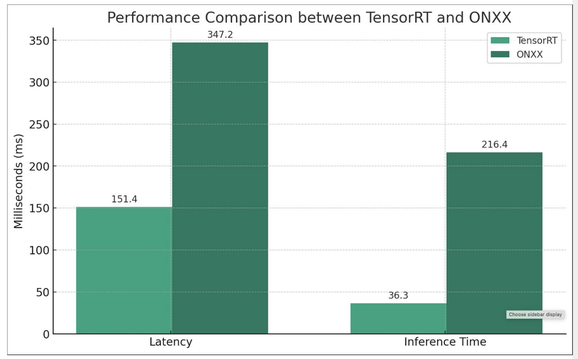
- DetectNet is a DNN designed for object detection and localization in images, optimized for performance and accuracy on NVIDIA platforms like Jetson.
- Yolo v5 is a object detection model designed to run on traditional hardware. However, the model can be converted to take advantage of tensorRT



- We found that on the Jetson, DetectNet was far faster than Yolo, even when TensorRT was enabled, likely due to DetectNet being specifically designed to take advantage of the underlying hardware

YOLO on TensorRT vs ONNX

- We were able to benchmark YOLO v5 on TensorRT vs ONNX
- TensorRT optimizes deep learning models on Jetson, which significantly boosted inference speed and reduced latency compared to ONXX
- TensorRT applies various optimizations like layer fusion and precision calibration (we used FP32, FP16)



- TensorRT compresses DNN and then streamlines its operation so it runs faster and uses less memory, without significantly impact accuracy
- It merges as many layers and operations as possible
- TensorFlow -> ONXX -> TensorRT

Challenges & Next Steps

Challenges

- We had difficulty setting up optimizations for our algorithms as a lot of the features like precision calibration require consistent sudo access to test and change
- Object detection models are somewhat shaky when processing at high frame rates. Objects tend to come in and out of boxes

Next Steps

- Run more complex and specific image recognition algorithms on the Jetson
- Develop our own image detection algorithm with TensorRT
- Make the object detection less sporadic

Credits
(Advisor) Jeff Young: jyoung9@gatech.edu
(Advisor) Aaron Jezghani: ajezghani3@gatech.edu
Srijan Ponnala: sponnala6@gatech.edu
James White: jwhite398@gatech.edu
Suzan Manasreh: smanasreh6@gatech.edu

Real-Time Object Detection