

ELE 548: Progress Report

Calvin Higgins

Department of Computer Science and Statistics
University of Rhode Island

November 19, 2025

Project Overview

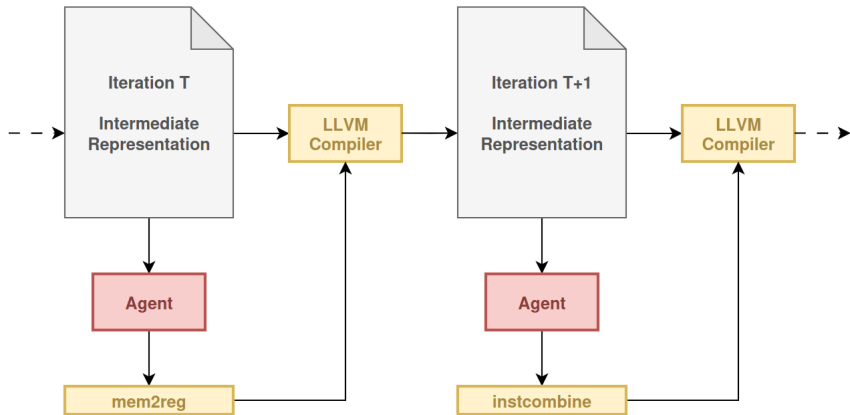


Figure: Reinforcement learning for the phase-ordering problem. Adapted from [4].

CartPole Implementation

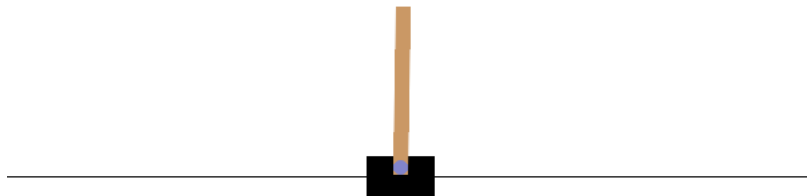


Figure: CartPole, an introductory reinforcement learning environment.

CompilerGym Setup

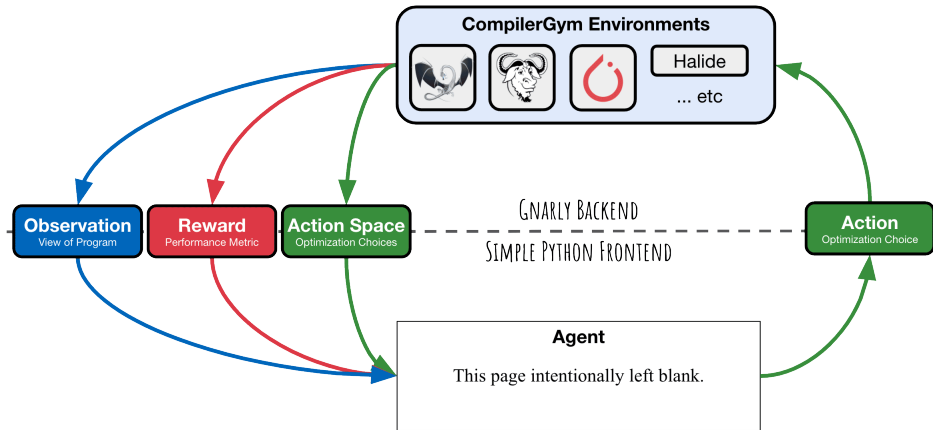
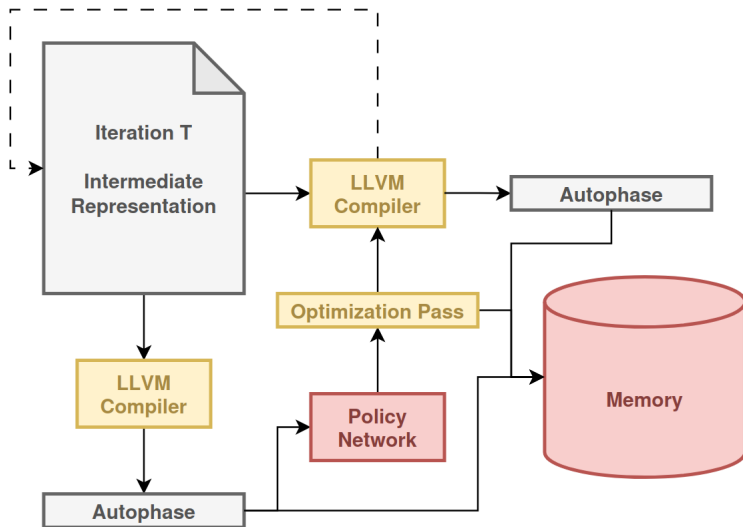
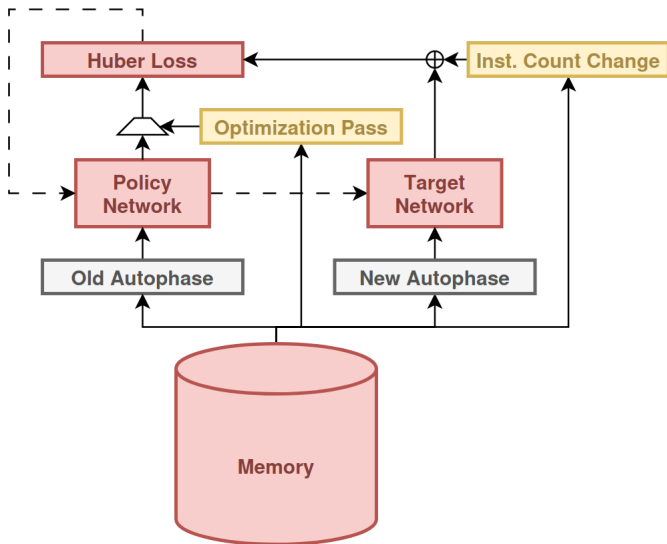


Figure: CompilerGym architecture [3].

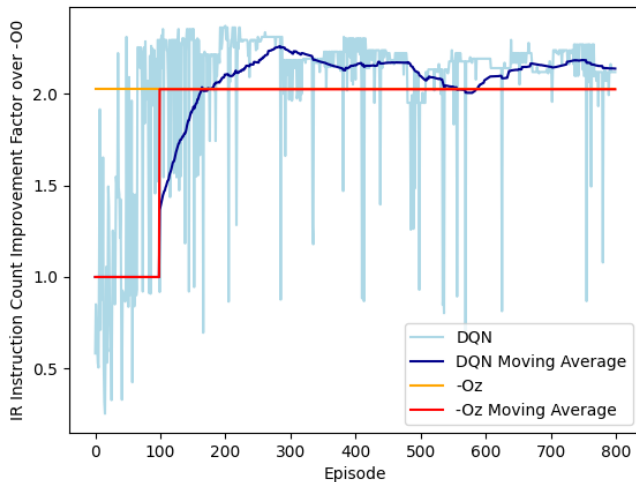
Deep Q-Network Rollout Implementation



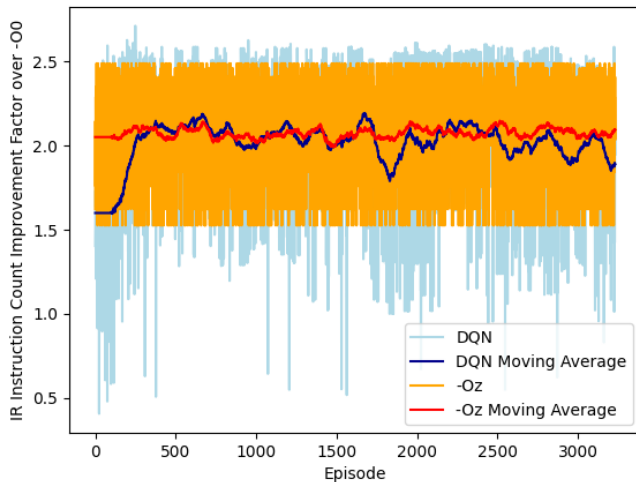
Deep Q-Network Training Implementation



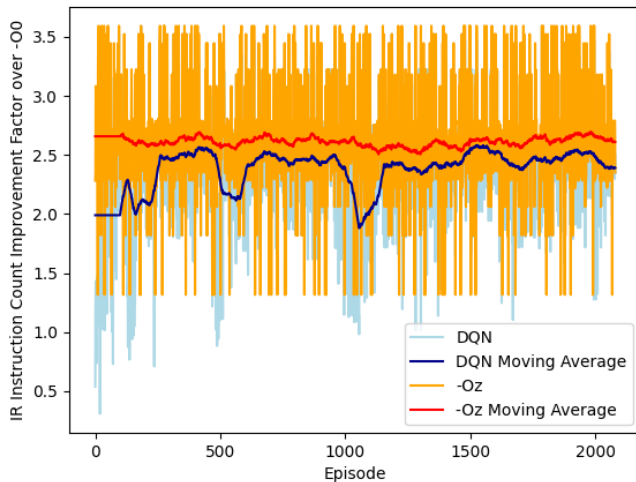
QSort ($n = 1$)



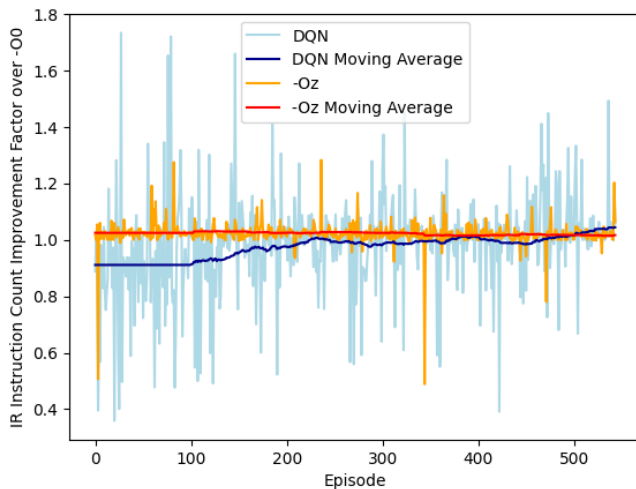
CHStone [7] ($n = 12$)



MiBench [5] ($n = 40$)



Tensorflow [1] ($n = 1985$)



Tensorflow [1] ($n = 1985$)

```
Traceback (most recent call last):
  File "dqn2.py", line 357, in <module>
    main(cfg)
  File "dqn2.py", line 352, in main
    train(sys, cfg, stat)
  File "/workspace/.venv/lib/python3.7/site-packages/torch/autograd/grad_mode.py",
line 27, in decorate_context
    return func(*args, **kwargs)
  File "dqn2.py", line 278, in train
    train_episode_batch(sys, cfg, stat)
  File "/workspace/.venv/lib/python3.7/site-packages/torch/autograd/grad_mode.py",
line 27, in decorate_context
    return func(*args, **kwargs)
  File "dqn2.py", line 200, in train_episode_batch
    for initial_cost, episode_cost in zip(initial_costs, episode_costs)
  File "dqn2.py", line 200, in <genexpr>
    for initial_cost, episode_cost in zip(initial_costs, episode_costs)
ZeroDivisionError: division by zero
```

Remaining Tasks

Monitoring (1 day):

- ① Track additional statistics
 - ① Rewards, actions, Q-values, instruction counts

Feature Engineering (1 week):

- ① Normalize inputs and rewards to baseline program
- ② Pre-train auto-encoder on Autophase [6] features
- ③ Try Transformer encoder on inst2vec [2] embeddings

Evaluation (1 day):

- ① Evaluate on held-out test-sets

Model (1 week):

- ① Implement MuZero
- ② Implement EfficientZero

- [1] Martín Abadi et al. “TensorFlow: a system for large-scale machine learning”. In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2016. ISBN: 9781931971331.
- [2] Tal Ben-Nun, Alice Shoshana Jakobovits, and Torsten Hoefer. “Neural code comprehension: a learnable representation of code semantics”. In: *International Conference on Neural Information Processing Systems (NeurIPS)*. 2018.
- [3] Chris Cummins et al. “CompilerGym: robust, performant compiler optimization environments for AI research”. In: *International Symposium on Code Generation and Optimization (CGO)*. 2022. DOI: 10.1109/CGO53902.2022.9741258.

References II

- [4] Chaoyi Deng et al. “CompilerDream: Learning a Compiler World Model for General Code Optimization”. In: *Conference on Knowledge Discovery and Data Mining (KDD)*. 2025. DOI: 10.1145/3711896.3736887.
- [5] M.R. Guthaus et al. “MiBench: A free, commercially representative embedded benchmark suite”. In: *IEEE International Workshop on Workload Characterization (WWC)*. 2001. DOI: 10.1109/WWC.2001.990739.
- [6] Ameer Haj-Ali et al. “AutoPhase: Juggling HLS Phase Orderings in Random Forests with Deep Reinforcement Learning”. In: *Machine Learning and Systems (MLSys)*. 2020.
- [7] Yuko Hara et al. “CHStone: A benchmark program suite for practical C-based high-level synthesis”. In: *International Symposium on Circuits and Systems (ISCAS)*. 2008. DOI: 10.1109/ISCAS.2008.4541637.