AIP Mini-Project - Reinforcement Learning

A3C in VizDoom

Anders Christiansen

Table of Contents

Project Description

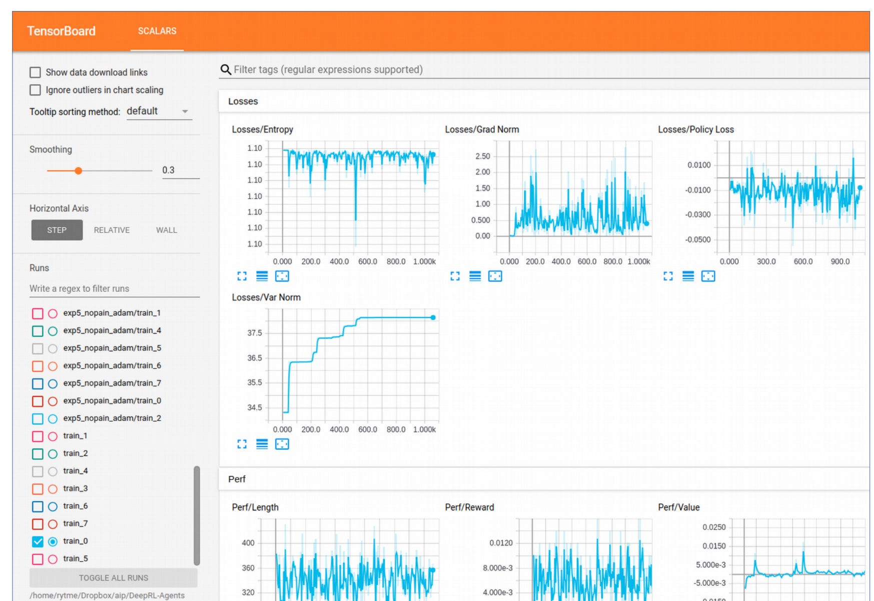
- Running the “Asynchronous Advantage Actor-Critic” method in the VizDoom environment, comparing scenarios and hyperparameters.

Motivation

- VizDoom state-of-the-art within RL
- Great environment for comparing hyperparameters
- A3C paper did not test in VizDoom, but in OpenAI

Tools

- VizDoom
- Tensorflow
- TensorBoard
- Run on NVIDIA GeForce 980 GTX



Resources

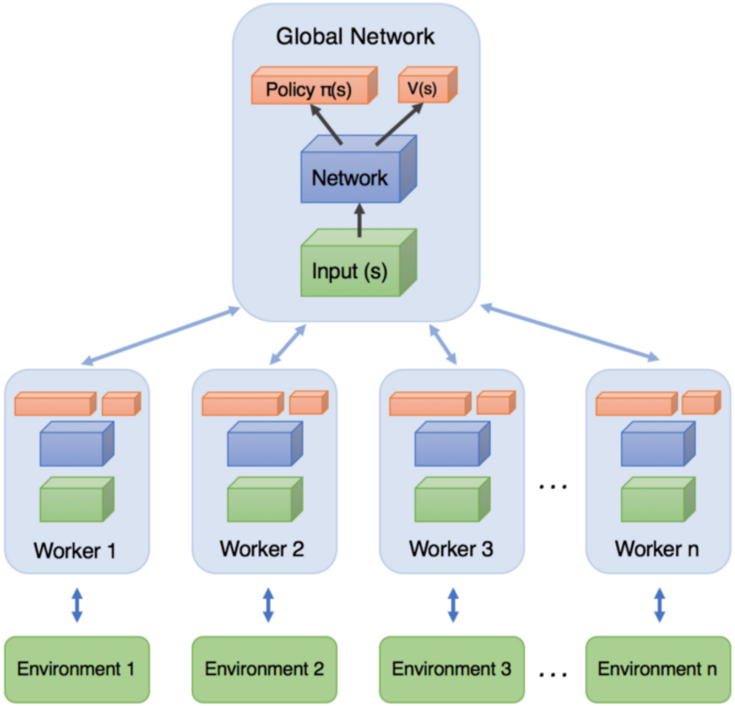
- A3C based on Juliani implementation
  - https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2
- VizDoom - https://github.com/mwydmuch/ViZDoom
- Project Git - https://github.com/RytmeAnders/aip_miniproject

Contribution

- Modifying VizDoom
- Modifying hyperparameters
- Results written to a list, then exported to a .txt

Algorithm - A3C

- **Asynchronous:** Multiple agent running in parallel, each with their unique instance of the environment
- **Advantage:** Calculating how much better a chosen action is than expected (R – V(s))
- **Actor-Critic:** Combining Q-learning and policy gradient methods. The agent (the actor) determines a policy, which is then updated by a state-value-estimate from the environment (the critic)



Algorithm - A3C

- Proposed by Mnih et. al. (2016)
  - https://arxiv.org/pdf/1602.01783.pdf
- Works in discrete and continuous state-spaces
- Much faster than a traditional Deep Q-Network (DQN)

Algorithm - A3C

- Learning by value loss and policy loss

- **Value loss:** The sum of squared advantages across n workers
  - L = Loss function
  - A = advantage
  - discounted return
  - state-value
  - N = number of threads (CPU kernels)

$$L = \sum_{i=0}^{n} A_i^2 = \sum_{i=0}^{n} (R_i - V_i(s))^2$$

- **Policy loss:** Negative logarithm of the product of the policy, the advantage, and the entropy
  - Π(s) = policy for state s
  - A(s) = advantage of state s
  - nt/coefficient
  - ntropy of policy Π

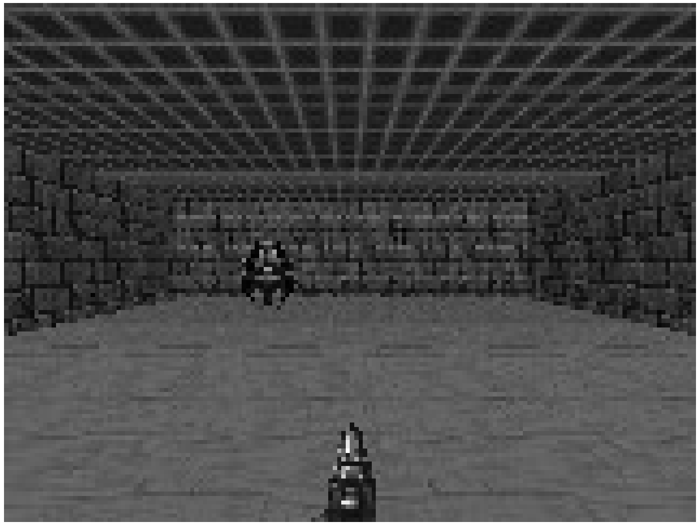
$$L = -log(\pi(s)) * A(s) - \beta * H(\pi)$$

List of configurations

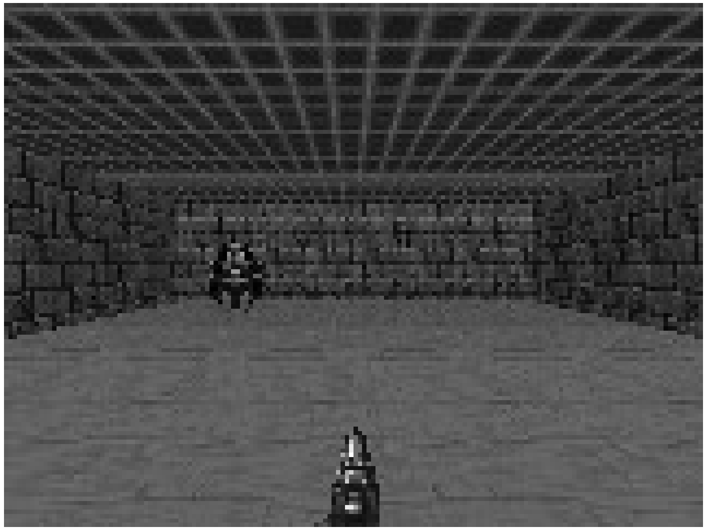
Name	Hyperparameters	Value	Scenario
2: nopain_rms	Living Reward: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	0 RMSprop 7e-4 0.99 0.1	Basic.wad
3: pain_adam	Living Reward: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	-1 ADAM 1e-4 0.99 0	Basic.wad
4: pain_rms	Living Reward: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	-1 RMSprop 7e-4 0.99 0.1	Basic.wad
5: nopain_adam	Living Reward: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	0 ADAM 1e-4 0.99 0	Basic.wad
6: defend	Death Penalty: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	1 ADAM 1e-4 0.99 0	DefendTheCenter.wad
7: lowLR	Living Reward: Optimizer: Learning Rate: Discount Factor: Epsilon-Greed:	-1 ADAM 1e-10 0.99 0	Basic.wad

Results

- RMSprop episode 50

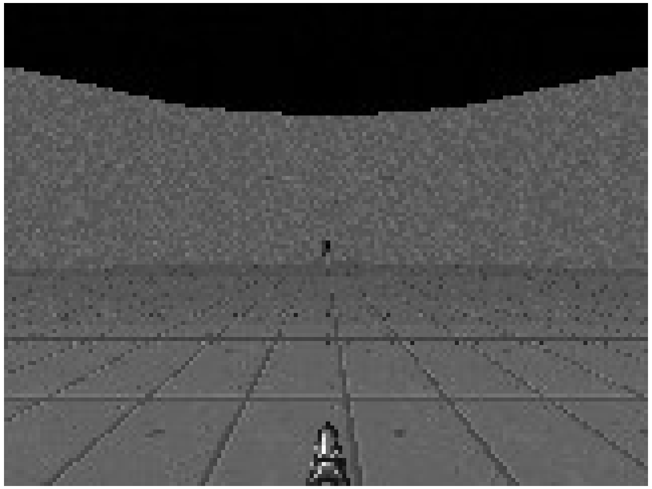


RMSprop episode 2100

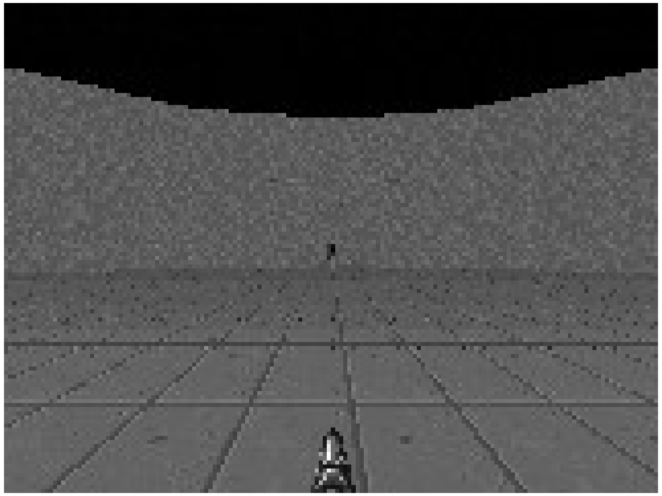


Results

- ADAM episode 25



ADAM episode 1875



Results

- Tensorboard

Discussion

- In contrast to the A3C paper, ADAM optimization seems best for VizDoom
- Very small learning rate does not seem to converge
- Also -1 reward on movement is not that necessary in the current configuration