A3C Methods for Game Playing

Rahul Palamuttam (rpalamut), William Chen (wic006)
Stanford University

Introduction

The asynchronous advantage actor-critic model is demonstrated to be an effective reinforcement learning algorithm at solving various game playing environments (BeamRider, Breakout, Space Invaders). This is due to the fact that parallel actors are leveraged to accumulate gradient updates and achieve learning stability. However, when running the A3C reinforcement learning algorithm on driving games like DuskDrive and NeonRacer, the optimal behavior is going straight, i.e. pressing the 'up'-key. It's interesting because the A3C algorithm is effective in generalizing across a variety games in open ai universe (BeamRider, Breakout, SpaceInvaders). Unfortunately in driving games it resorts to trivially going straight to maximize reward. We believe this occurs for two reasons:

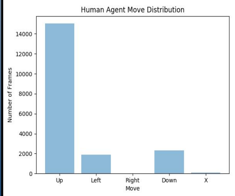
1. Network does not represent feature coordinates and suppress distractors.
2. No reward signal for accomplishing intermediate tasks.

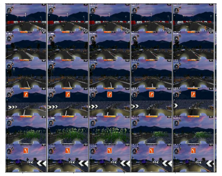
Problem Statement

We seek to improve the performance of the OpenAI universe starter agent on the flashgame DuskDrive. We attempt to use spatial-softmax to enable the network to represent feature coordinates and suppress distractors. Our evaluation consists of implementing a spatial-softmax layer and comparing the reward curve with that of the baseline agent and evaluating various learning rates. We also evaluate an additional metric which is reward variance to study the distribution of rewards per step as the models train.

Dataset

Our dataset consists of frames from the first level of the DuskDrive game. We also evaluated a human agent and the corresponding move distribution for 10 games. We were able to achieve a score of ~75k by manually playing the game while the model converged to ~35k.





Methods & Models

A3C Dueling Network Controller:

- 4 Convolutional layers
  - 32 (3,3) filters with (2,2) strides and 'same' padding
- LSTM with cell size of 256
- Spatial softmax layer
- Expected pixel position layer
- Fully connected layer for policy
- Fully connected layer for value

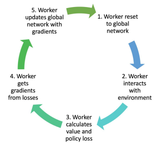
A3C Loss Functions:

- Value loss: $L_V = \Sigma(R - V(s))^2$
- Policy loss: $L_p = \log(\pi(s)) * A(s) + \beta * H(\pi)$

Generalized Advantage Estimator:

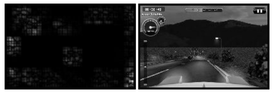
- Discounted sum TD residuals: $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$
- K-step discounted advantage: $\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V$
- Generalized advantage estimator: $\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$

Worker Training Flow:

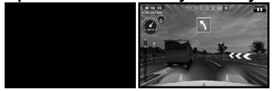

5. Worker updates global network with gradients
1. Worker reset to global network
2. Worker interacts with environment
3. Worker calculates value and policy loss
4. Worker gets gradients from losses

Results

Baseline : Policy Saliency

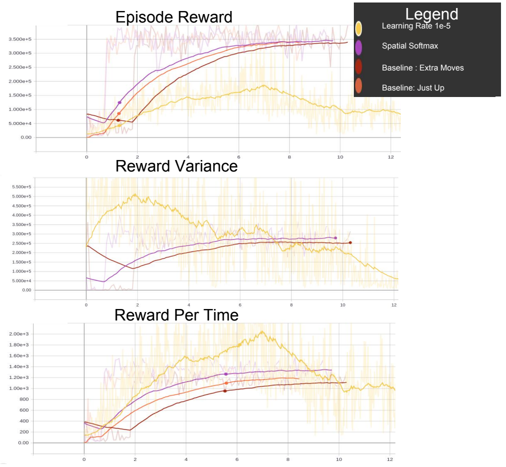


Spatial Softmax : Policy Saliency



Policy Saliency

- Baseline activation regions correspond to the bushes, score, and speedometer.
- Spatial softmax showed no activations. Note these are taken once training has converged.
- No value saliency.

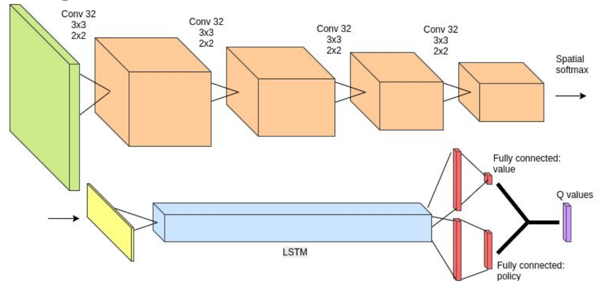


Performance Results

- With the additional spatial softmax layer, model achieved convergence slightly faster than baseline.
  - SS intended to suppress distractor features
- We see that with a lower learning rate the model yielded a lower episode reward but a higher reward per time. "Tried many moves"
  - Exploration heavy: reduced speed and episode length
- With a higher learning rate the baseline achieved a higher reward but slightly lower reward per time. "Learned to just go straight"
  - Exploitation heavy: increased speed and episode length

Verdict : The best model still learns to just go straight

A3C Dueling Network Architecture



Conclusions

- The A3C model optimizes for speed and in driving games that essentially means the model exploits rather than explores.
- The ideal game should have a nearly constant reward per time measure. However, for all four experiments the reward per step variance was very large. We need to have a lower reward variance.

Future Work:

- Due to technical challenges we weren't able to incorporate intermediate signals such as the type of turn to take for a given image. This would essentially be a second convolutional neural network that maps to a (left, right, up, down) vector.
- We also want to incorporate experience replay

References

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. CoRR, abs/1602.01783, 2016.
[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. CoRR, abs/1504.00702, 2015.
[3] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. CoRR, abs/1506.02438, 2015.
[4] Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. CoRR, abs/1511.06581, 2015.