Using Transfer Learning Between Games to Improve Deep Reinforcement Learning Performance

Chaitanya Asawa[1], Christopher Elamri[2], David Pan[2]

[1] Department of Computer Science, Stanford University, [2] Department of Electrical Engineering, Stanford University

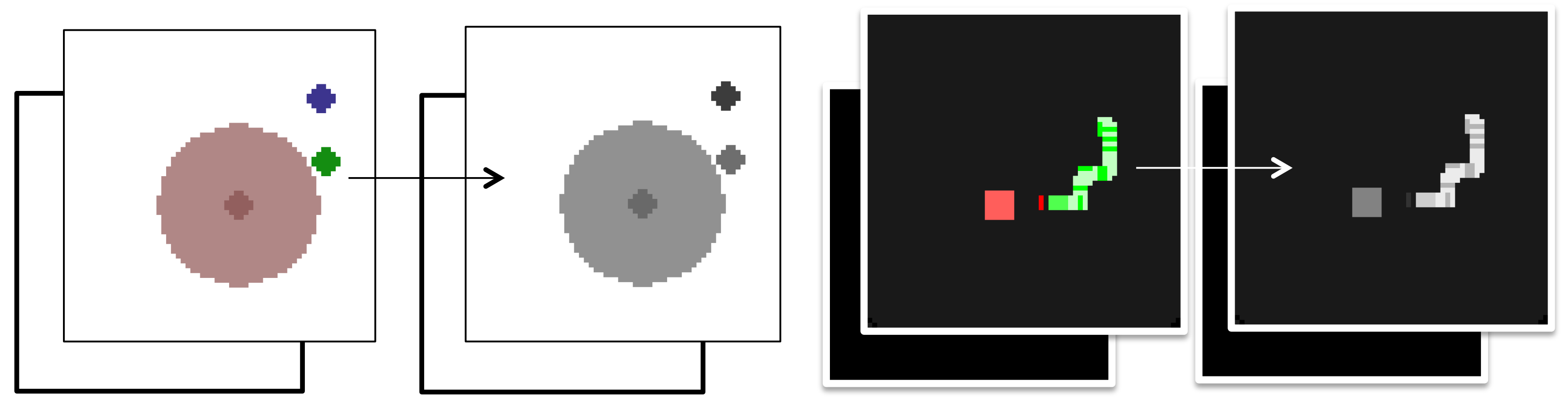
Motivation

The ability to successfully transfer learn is useful as it allows us to avoid having to train specific models for every task we may have (which takes much computational power and time), and rather achieve high performance quickly using what we have generally learned.

Background

- We aim to play two games with Deep Q-learning, Snake and PuckWorld, and hope to learn from one game for another
- Deep Q-learning has performed very well on Atari games in the past; it can generalize across huge state spaces
- In PuckWorld, an agent aims to stay around some target while avoiding a large red puck that slowly follows it
- In Snake, an agent collects food, growing longer each time, while trying to avoid running into itself or a wall

Preprocessing



- To have a better sense of direction (and velocity in the case of PuckWorld), we maintain a state history
- To reduce the input size, we use gray scale images instead of RGB images
- We skip a constant number of frames to play more games during training

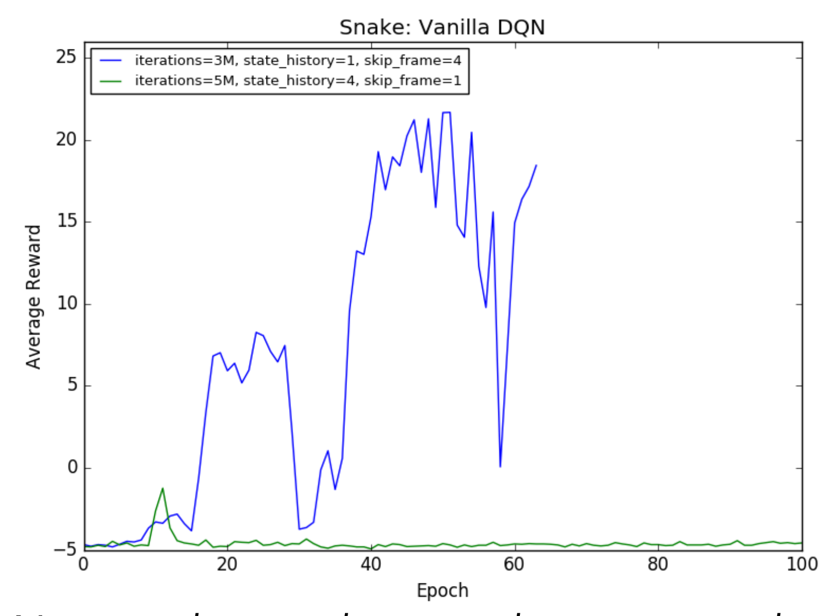
Rewards

- Snake: -5 for running into itself/wall, +1 for collecting reward
- PuckWorld:
  - Function of distance to the green circle (want to be closer)
  - Negative reward proportional to distance from the red puck's center if we are within the red puck's radius.

Vanilla DQNs

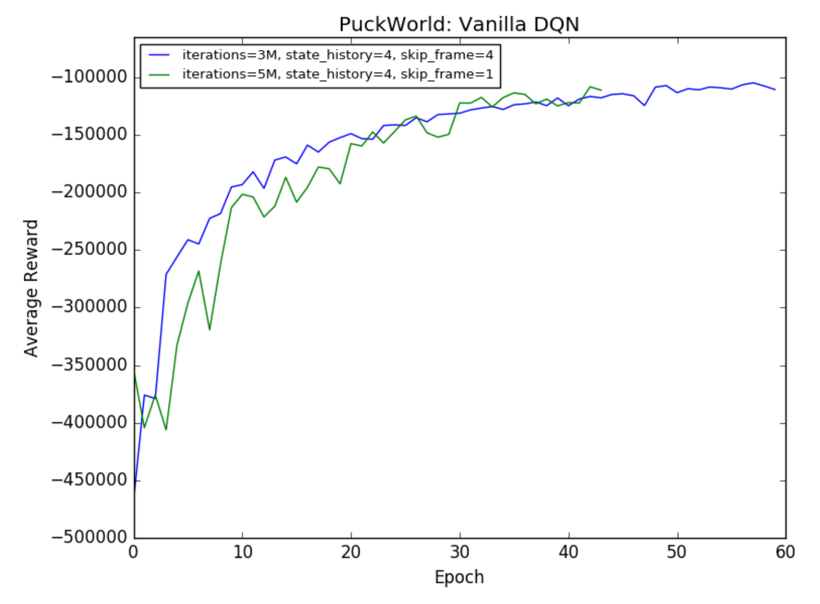
- We used Deep Q-Networks that had 3 convolutional layers of 32 filters (size 3 by 3) with stride 1, followed by FC-256, and a final FC to the number of actions.
- In addition, we use experience replay and a target Q-network.

Snake



We see that with more history, it is harder for Snake to learn. In addition, when it does learn well, there is a lot of oscillation.

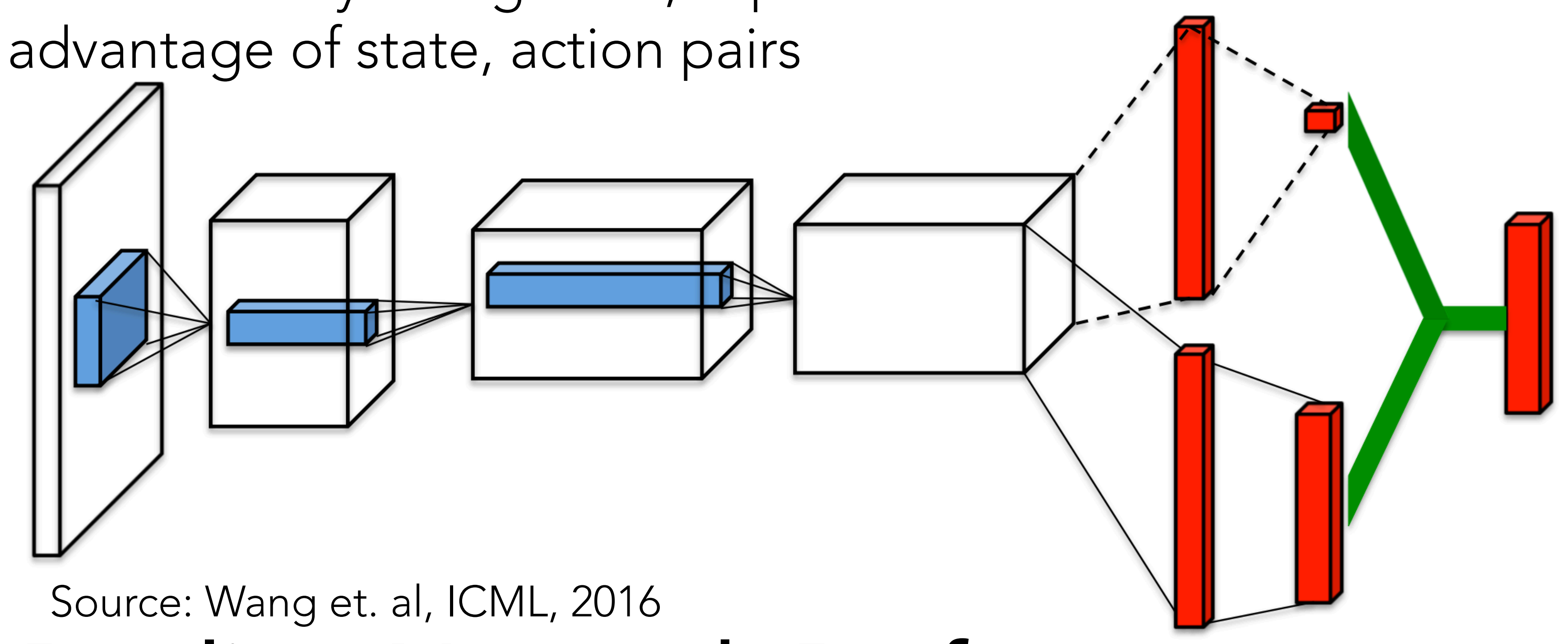
PuckWorld



PuckWorld seems to converge smoothly in a relatively similar fashion regardless of the size of history.

Dueling Network

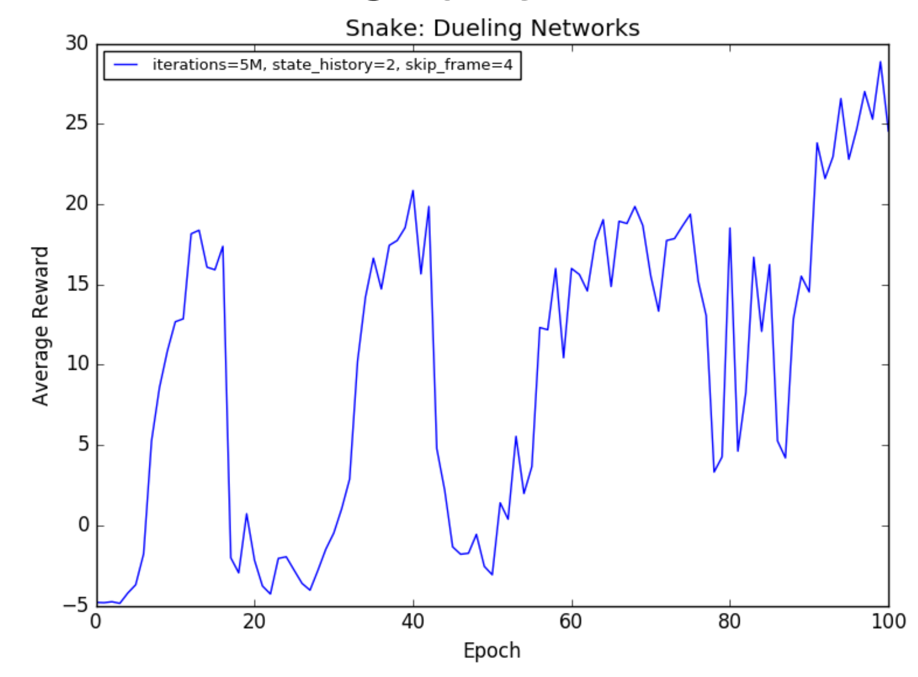
Introduced by Wang et. al, separates value of state from advantage of state, action pairs



Source: Wang et. al, ICML, 2016

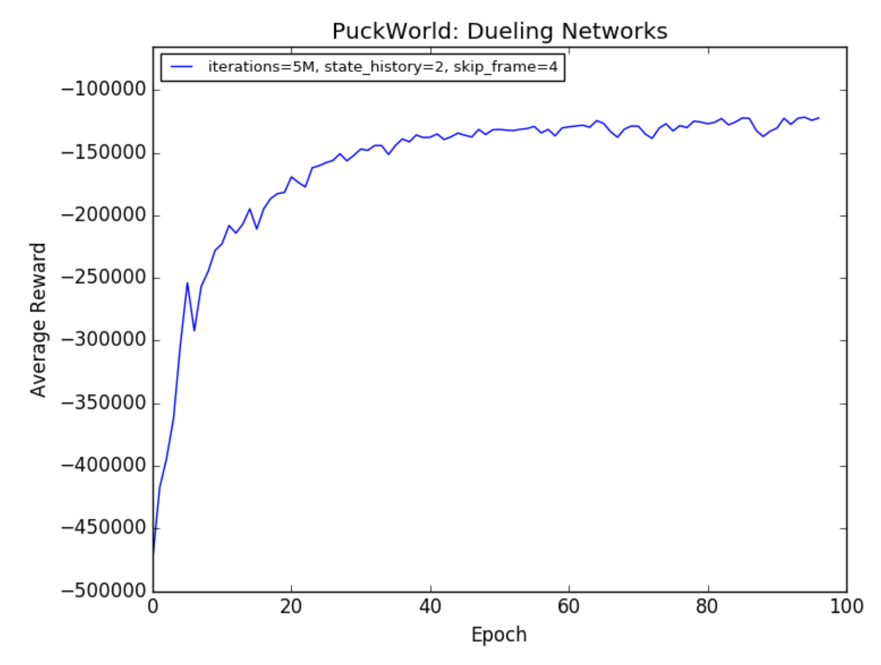
Dueling Network Performance

Snake



We still see oscillation in the rewards, but looks like greater final performance and overall performance.

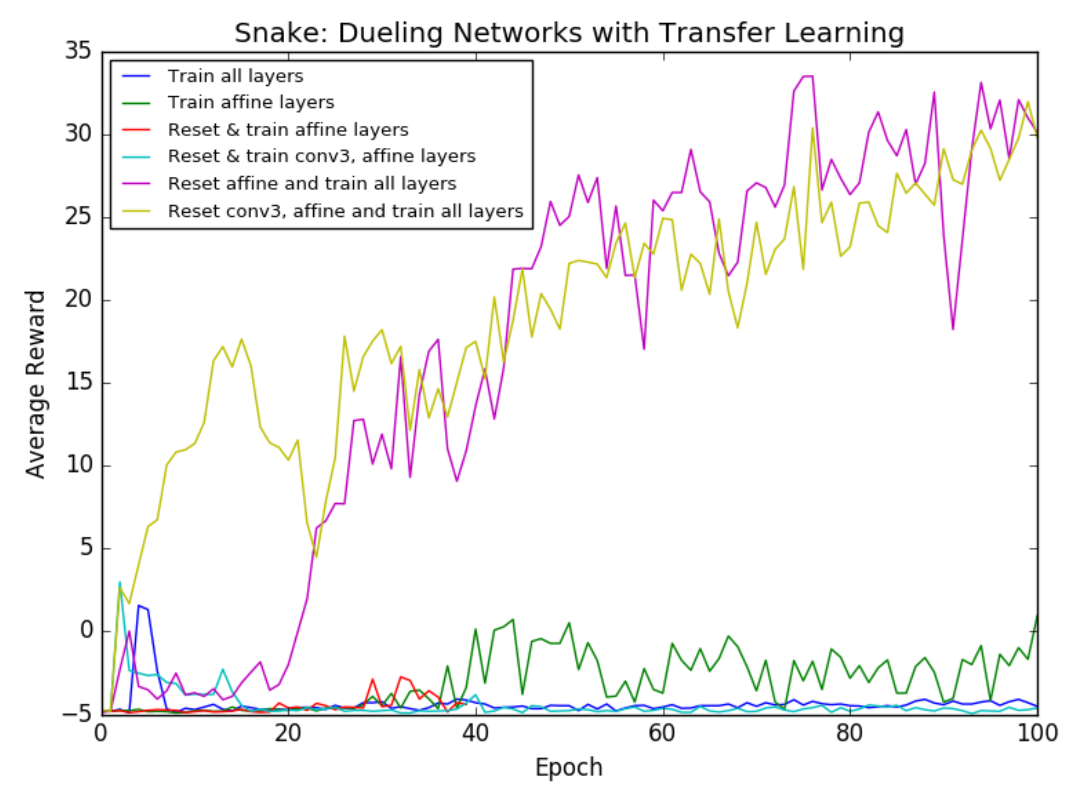
PuckWorld



PuckWorld seems to converge around the same as final reward as Vanilla DQNs – maybe even slightly worse.

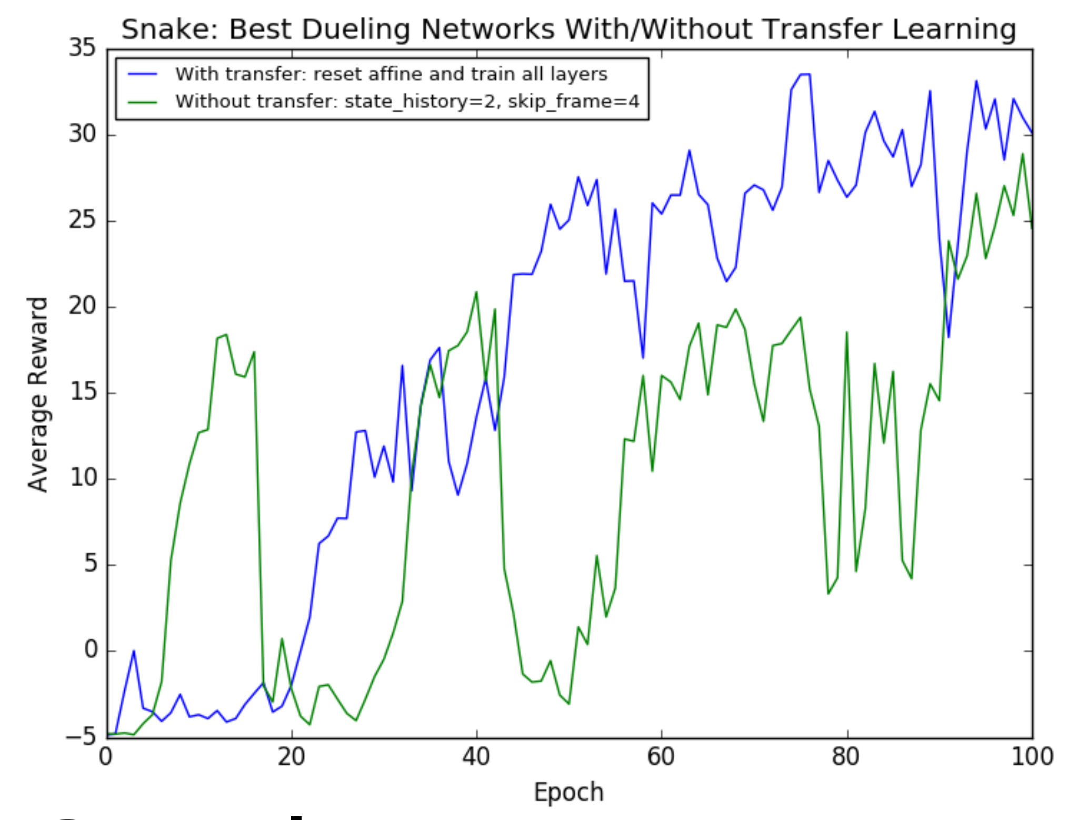
Transfer Learning

We then experiment with using the weights we learned from PuckWorld in terms of usefulness for playing Snake. We explore various combinations of two approaches:
- Retraining layers (so the weights can change)
- Reinitializing layers (starting from totally fresh weights)



We find greatest success when retraining all layers in addition to reinitializing some of the final layers. Otherwise, we seem to fail to learn.

Benefit of Transfer Learning



It seems that transfer learning made significant improvements in terms of preventing the rewards from oscillating, in addition to earning higher rewards as it started learning better and in the end.

Conclusion

- Factors such as size of state history can make it more challenging for DQNs to learn
- We see in playing Snake oscillating rewards over time. We hypothesize that the Snake has difficulty adjusting to its longer length and takes less optimal actions.
- Transfer learning can help with both reward oscillation and improving performance.

Future Directions

While we found success in transferring PuckWorld weights to Snake, we'd ideally like to see if this scheme generalizes to more games using the same weights – both similar ones and different ones.