Deep Reinforcement Learning and Transfer Learning with Flappy Bird

Cedrick Argueta[1], Austin Chow[1], Cristian Lomeli[1]

Department of Computer Science[1], Stanford University, Stanford, CA 94305

Motivation

Reinforcement learning is a technique for solving certain decision tasks where an agent learns how to act in a real world environment. In recent years, major breakthroughs in RL have come from common video games. Since we grew up playing tons of video games, we wanted to explore the applicability of RL to the games Flappy Bird and Pixel Copter.

Problem

- We wish to use reinforcement learning to play the games Flappy Bird and Pixel Copter, determining whether RL can beat us and/or an expert in score.
- Q-learning does not generalize well to large state spaces, since many states would be left unexplored. We can use **deep Q-learning**, which uses a neural network to approximate the Q-value function and allows us to generalize to unseen states.
- Instead of using the game's screen as input, we use **feature engineering**, which allows us to achieve the same level of performance without having to train a CNN to learn features for each game.
- We also aim to explore the impact of **transfer learning** to our task. In our case, transfer learning would be starting an instance of Pixel Copter training with weights already trained on Flappy Bird.

Challenges

- Training required extensive time and compute
  - Opted to forgo image input and use feature engineering instead
- Difficult to pull together several packages for this task
  - Used ALE, PLE, Keras-rl, OpenAI Gym, tensorflow
- Transfer learning was traditionally used for CNNs
  - Changed observation spaces of both games to be as similar as possible
- Couldn't use lots of data we collected since we changed our training and testing process several times

References

- Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *Neural Information Processing Systems.* (2013).
- Taylor, Matthew E., and Peter Stone. "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research* 10.Jul (2009): 1633-1685.
- Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8.3-4 (1992): 279-292.

Methods and Models

Deep Q Learning with experience replay

We use a neural network to approximate the Q function, and perform weight updates based on mini batches drawn from a cache of (s, a, r, s') tuples.

For each episode, we do:

Act → Observe → Update

Act	Observe	Update
Linearly annealed ϵ-greedy	Store in experience replay	Perform gradient descent w/ Adam optimizer on minibatch from memory

Neural Network Architecture

- Input in R^{11} and output in R^2, which are the Q-values for the two actions, **jump or not jump**.
- Each hidden layer pictured here has dimensionality of $2^{\text{num nodes pictured}}$
- The input layers and hidden layers are followed by **ReLU layers**, while the last layer is followed by a **linear activation layer**.

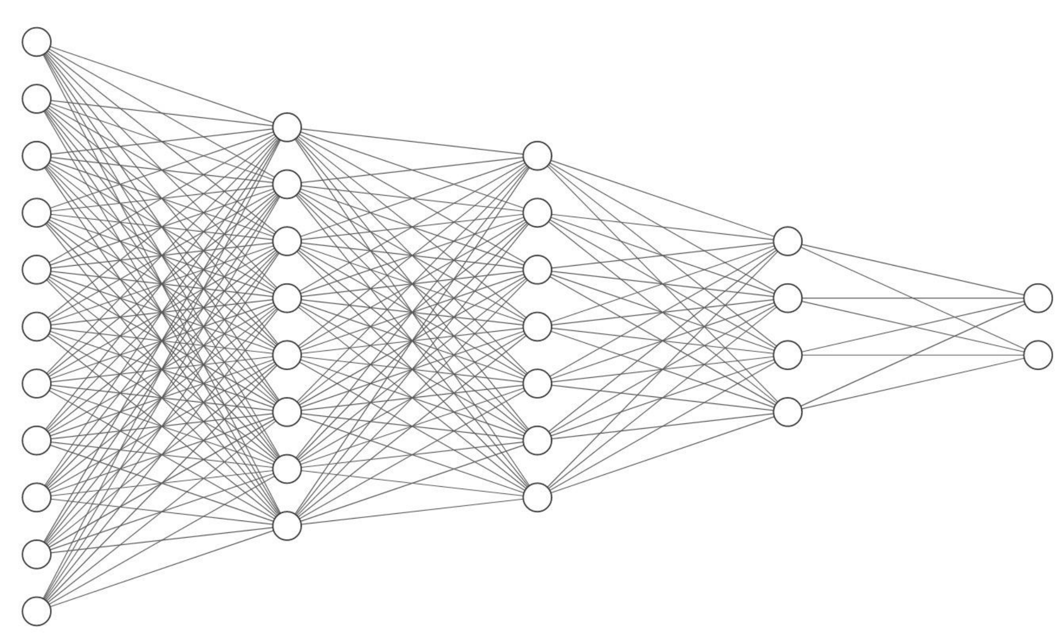


Table for hyperparameters

- For our DQN, we tested several hyperparameters through trial and error to optimize the performance of both games.
- If not noted here, the hyperparameters were left as their default values.

	Flappy Bird	Pixel Copter
Hyperparameters		
Target Model Update	1e-2	1e-2
Learning Rate	1e-3	4e-4
γ	0.99	0.99
Exploration Policy	Linearly annealed ϵ-greedy	
Annealed ϵ	0.2 → 0.05	
Annealing Steps	50,000	150,000
Warm Up Steps	50	100
Training Steps	100,000	300,000
Reward Profile		
Tick	0.1	0.1
Passed Obstacle	1.0	1.0
Collided	-10.0	-10.0

Feature Engineering

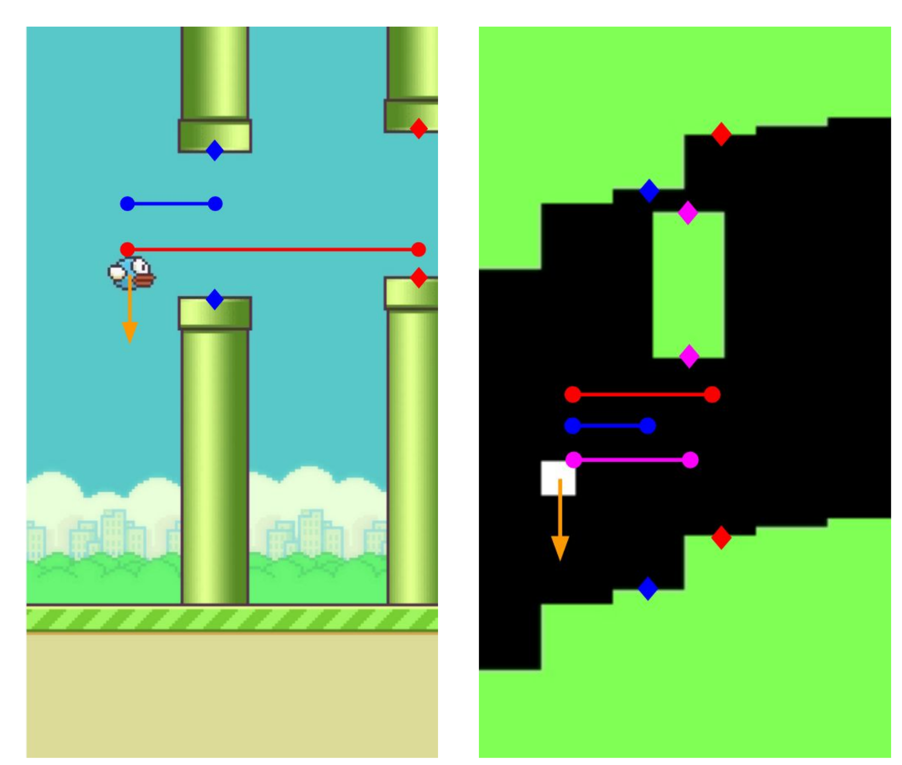
We had to come up with a way to make the state spaces similar to facilitate transfer learning, but didn't want to use images as input.

We used the following features for **both** games:

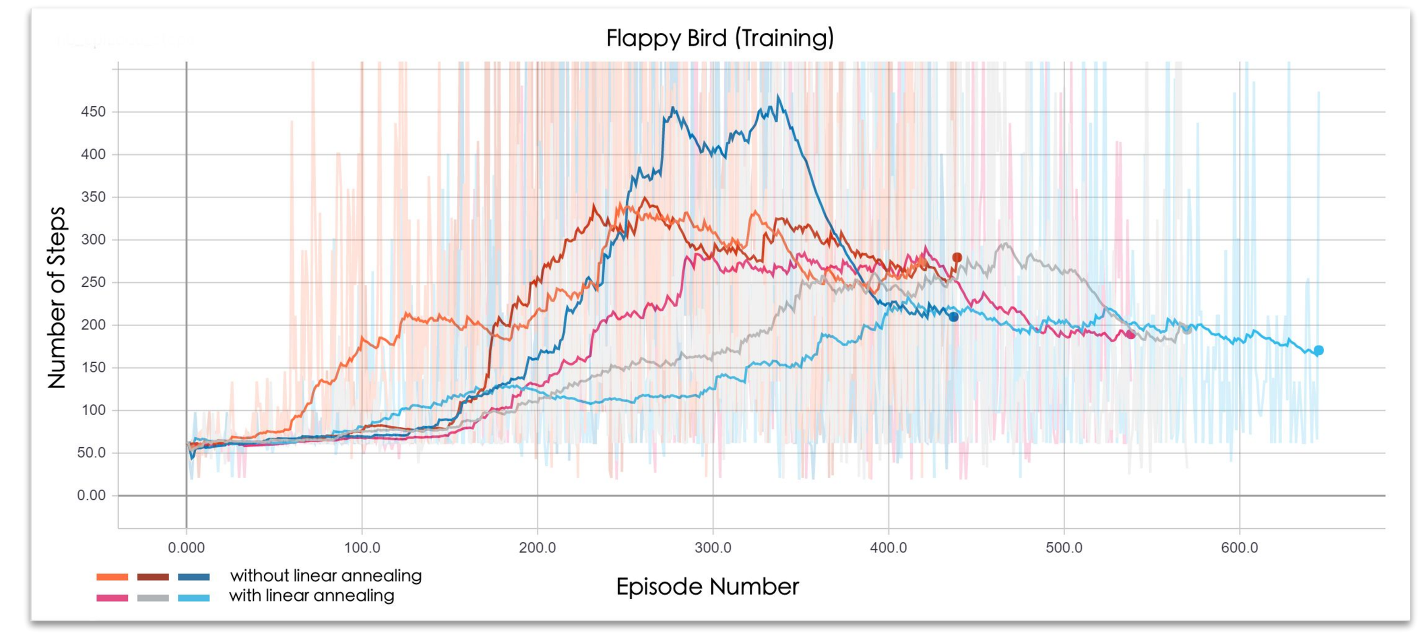
- y position and velocity (orange arrow)
- distance to next terrain (blue line) and next next terrain (red line)
- absolute y positions of the next terrain (blue dot) and next next terrain (red dot)

Additionally, Pixel Copter had obstacles that had no analogue in Flappy Bird:

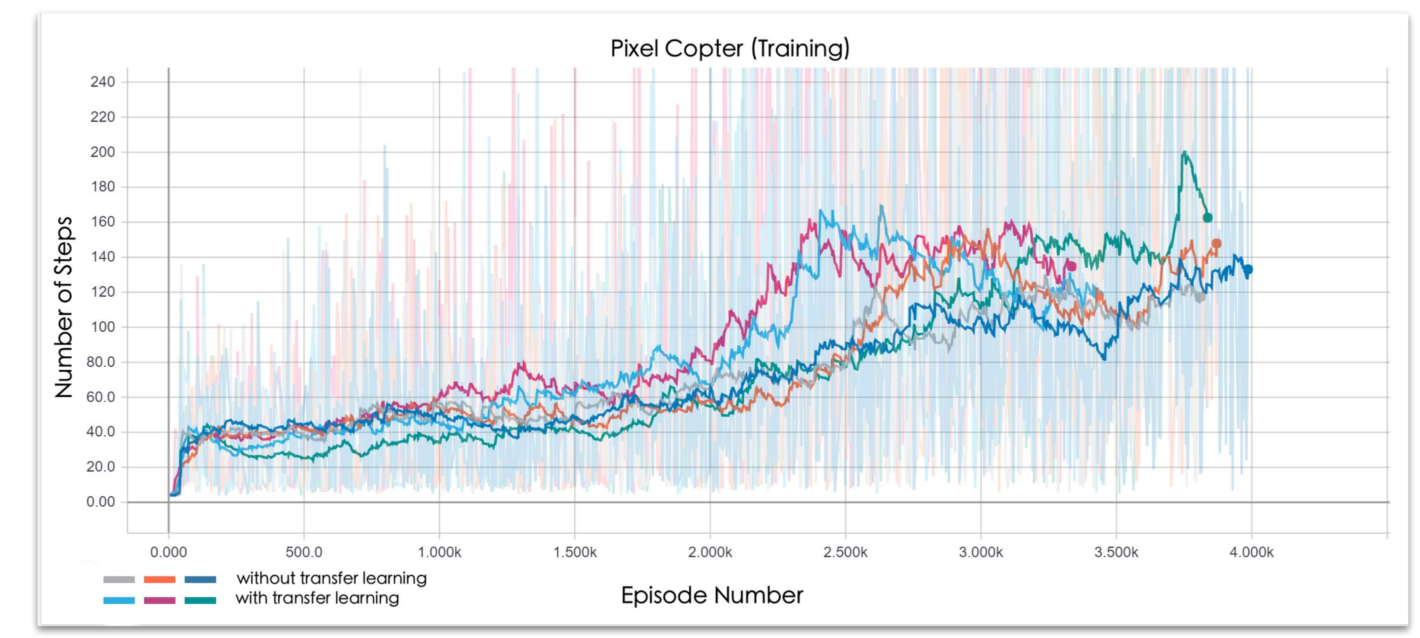
- distance to next block obstacle (purple line)
- absolute y positions of the next block obstacle (purple dots)



Results



6 runs of Flappy Bird training with the hyperparams to the left. We don't show testing data here since each model can run indefinitely.



6 runs of Pixel Copter training with the hyperparams to the left. Test data is shown below, demonstrating little to no improvement for average ability of the agent when using transfer learning, but improvement in the best performance for each agent.

Pixel Copter	w/o Transfer Learning	w/ Transfer Learning
Average # Steps / Episode	518.5	519
Average Top # Steps / Episode	1939	2544

Conclusion and Future Work

- Deep reinforcement learning was able to play both Pixel Copter and Flappy Bird better than we could, and for Flappy Bird in particular our agent reached **superhuman levels of ability**.
- We did see transfer learning improve training times for Pixel Copter and absolute performance slightly, but only after we used some tricks to ease the process.
- We expect that transfer learning when **using images as input** would be much more impactful, since we wouldn't need to relearn as much to interpret the game's screen.
- In the future, we'd like to play around with **games that have higher dimensionality** in terms of observation and action spaces. There are many cool things in RL right now, like OpenAI's Dota bot and DeepMind's AlphaGo Zero!