# DSIR Capstone 2021
# Intro to Reinforcement Learning With A Custom PyGame

Jennifer Fong

## Games People Play

Well, really games machines play...

https://www.pinterest.com/pin/331577591315324099/

## Problem Statement

- Create a simulation game where actors within the game would learn adaptive behaviors from experience and improve performance

Reality forced resetting of expectations

- Provide a simple, basic Python game as the data for a machine learning model
- Use reinforcement learning model to onboard itself on the custom app

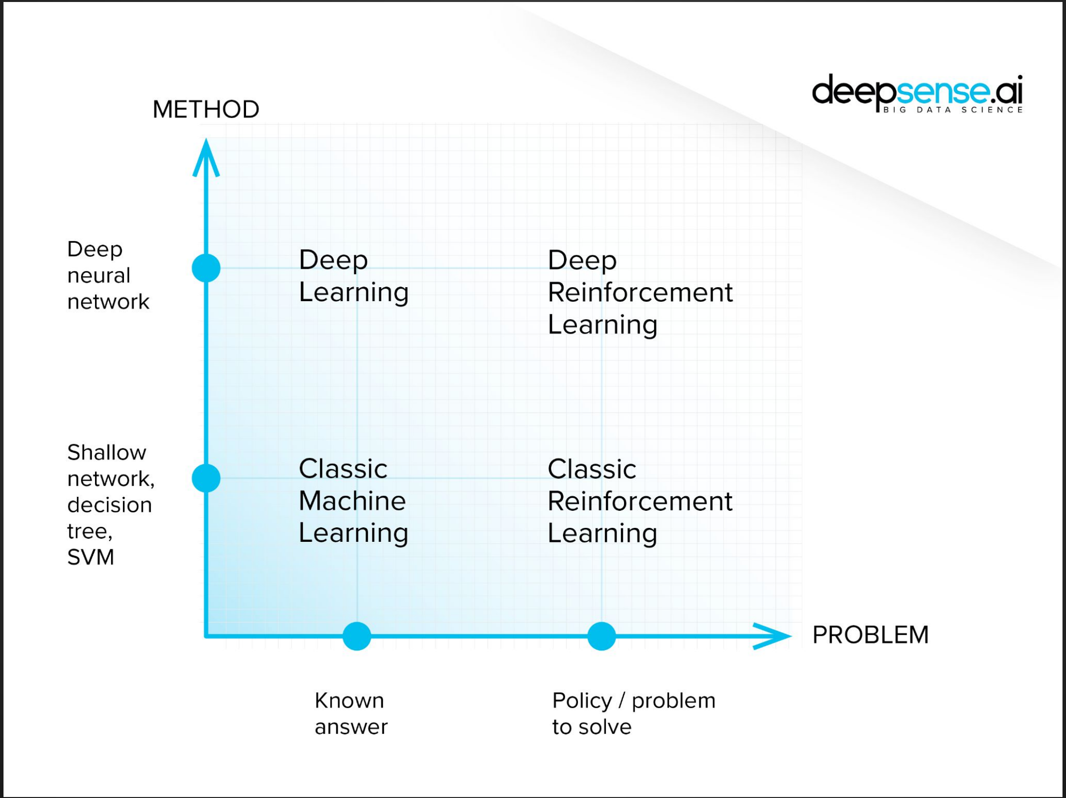
## Reinforcement Learning

- Training machine learning models to make decisions
- Machine learns in an uncertain, possibly complex environment
- Computer employs trial and error
- Learns through reward system
- Model is not given hints
- Unlike humans, machines can learn in parallel trials

https://www.forbes.com/sites/davidthier/2020/05/02/how-to-solve-the-may-day-maze-in-animal-crossing-new-horizons-and-the-rewards/?sh=74b56db5331a

https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

# Ok, where does reinforcement learning fall?

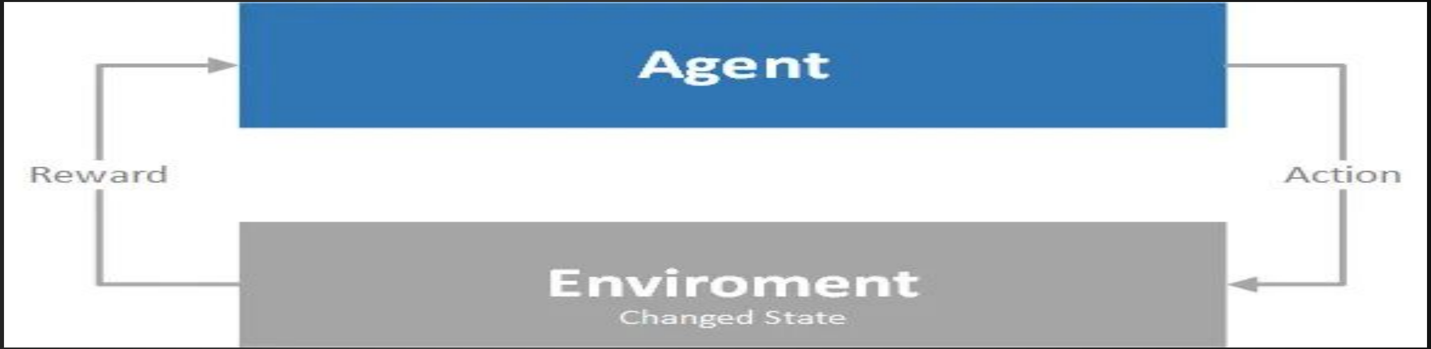

deepsense.ai BIG DATA SCIENCE
METHOD
Deep neural network
Shallow network, decision tree, SVM
Deep Learning
Deep Reinforcement Learning
Classic Machine Learning
Classic Reinforcement Learning
PROBLEM
Known answer
Policy / problem to solve

## So what does this look like?

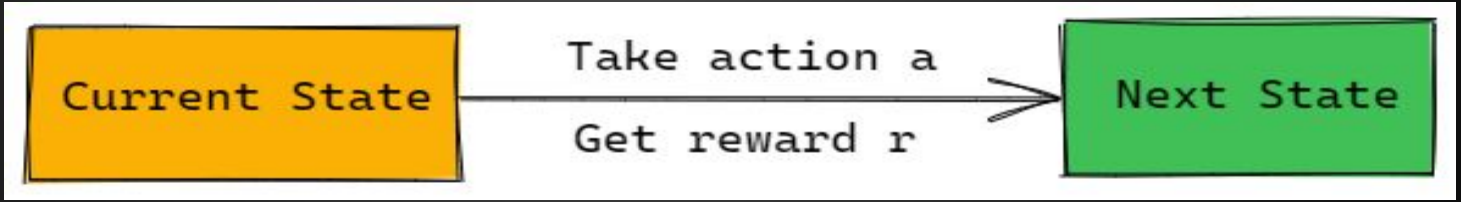
From Kaggle tutorials

- Define an agent/player
- Can provide game strategies (decision tree) and heuristics (assign points)
- Help agent look n steps ahead (minimax algorithm to weight the losses against gains)
- More advanced: can use neural networks with weights adjusted with each reward/penalty
- Note that CNNs can also be used to read images or the data output to a screen
- Success is not measured by single high score but averaged score
- One caveat: Tutorials use outdated TensorFlow version 1.x

## Vocabulary



https://rubikscode.net/2019/12/23/ultimate-guide-to-deep-q-learning-with-tf-agents/

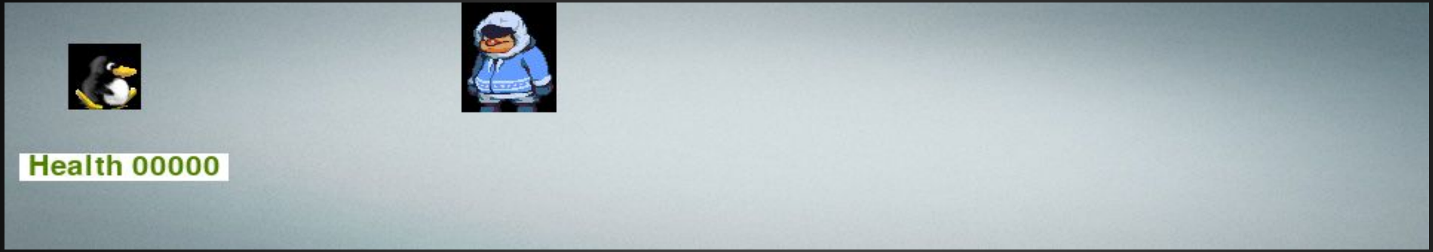


https://levelup.gitconnected.com/dqn-from-scratch-with-tensorflow-2-eb0541151049

# Popular Frameworks

| Environments | Framework | model |
|---|---|---|
| Open AI | Open AI + PyTorch | PPO<br>- Proximal Policy Optimization<br>- Used for robots<br>A2C<br>- Actor Critic |
| TF + Py Environment | TensorFlow/Keras | DQN (Deep Q Network)<br>CNN + DQN |
| Open AI Gym | Ray (UCB RISE lab)<br>- Supposed to be very easy | Deep Q Learning<br>● Did not work with, but worth the mention |

# Rules of Engagement



Simple Game

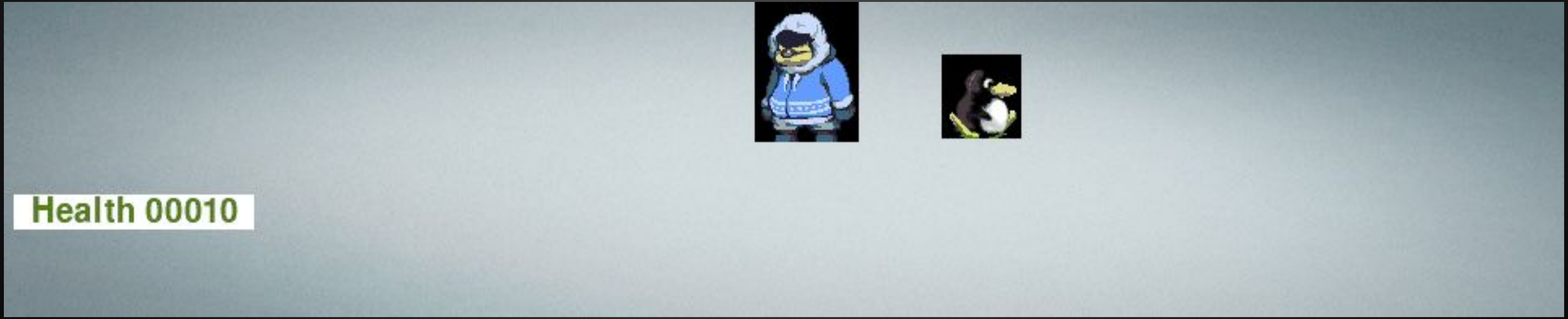
- Based off PyGame Tutorials
- Time span (5 seconds per game) for multiple iterations
- 1 agent:  animated penguin sprite with up to 2 movement options
  - Left, Right
  - For a while, there was a “nothing” option, but some models refused to move even without rewards
- 1 obstacle: enemy sprite (Man) that moves left and right
- Feedback/Rewards provided with points

## Before shot:  Agent acting randomly

We see the Penguin sprite fluttering around.  It tends to only go in one direction for very short distances.  It rarely approaches the enemy sprite.

## PPO

- Open AI environment + Algorithms take in pixels and discrete reward information
- Open AI default model due to ease of use and requires little tuning
- Here we see the penguin sprite trying to be close, but it also gets run over.  It loses 100 points if it collides, but gets 25 extra points if it can get close without contact.

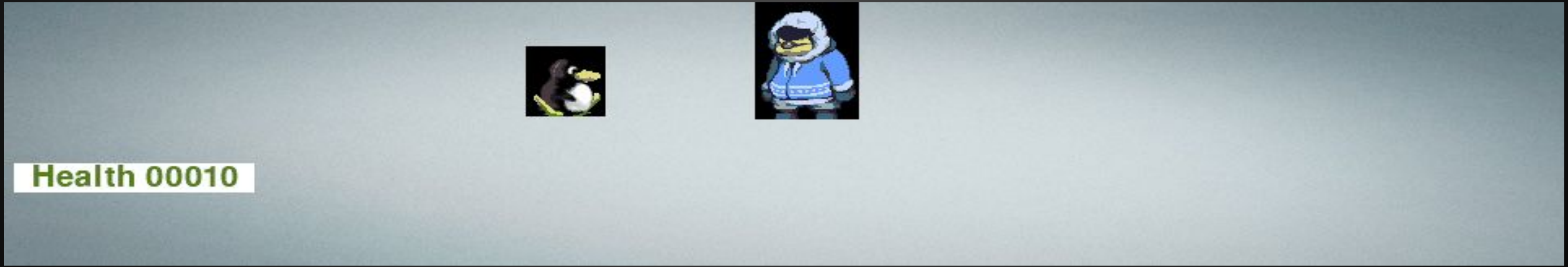


## A2C Trained

Open AI gym environment + algorithms take visual input and discrete reward information

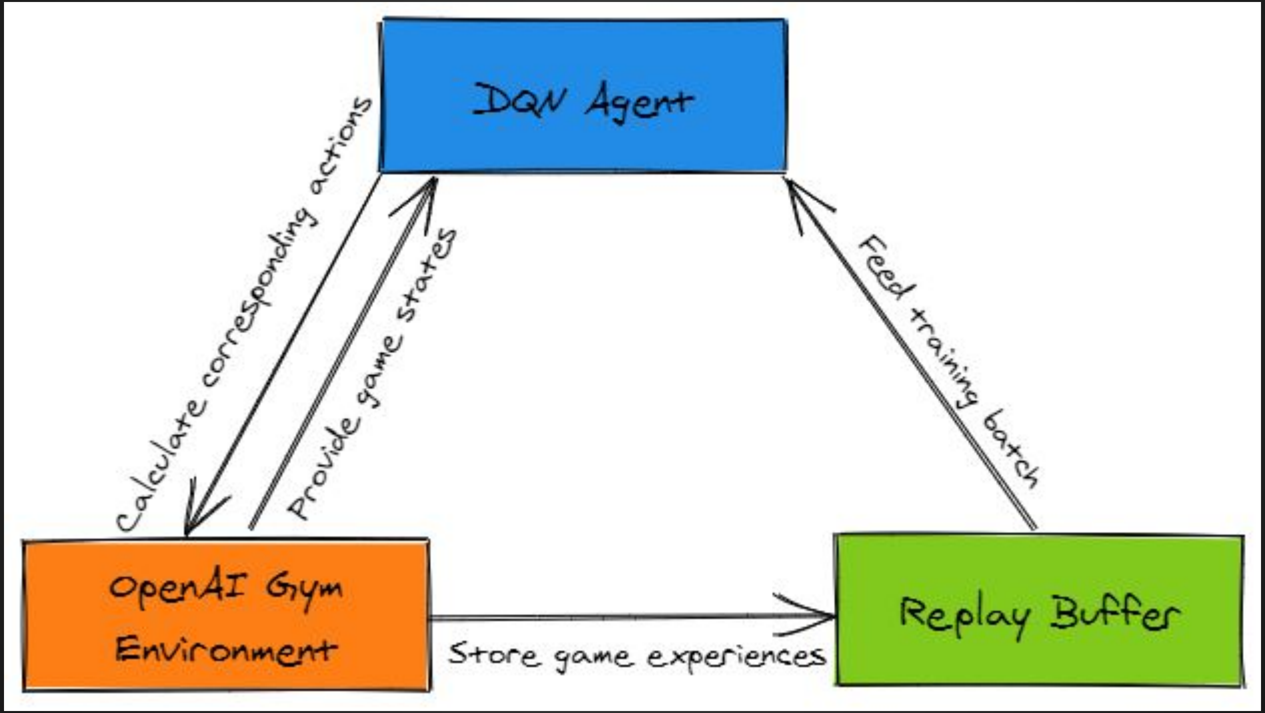
Actor-Critic: Critic estimates the value and actor updates policy distribution

Note that the penguin does try to go in the vicinity of the person, even if it is not successful at avoiding a collision.  Without the higher risk reward, it would huddle to one side away from the "enemy"



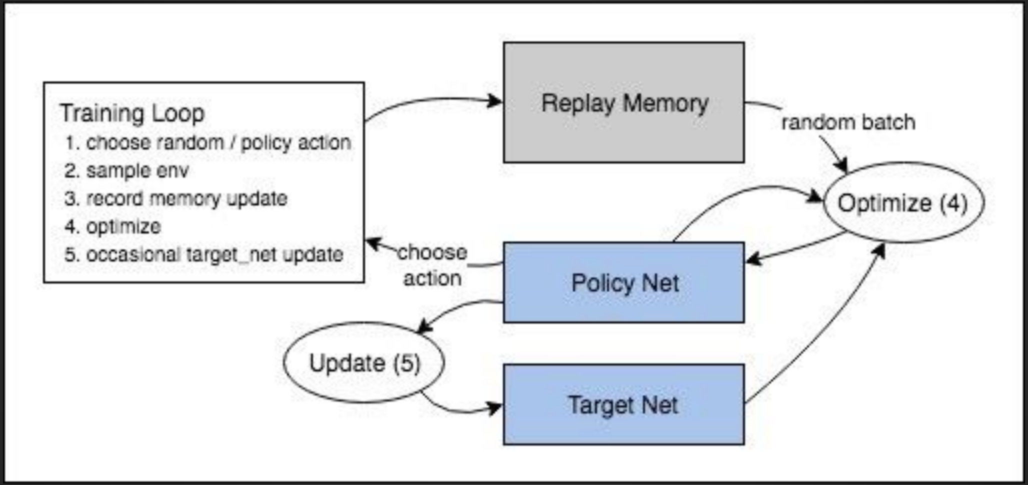
https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f

## Some Deep QN background



https://levelup.gitconnected.com/dqn-from-scratch-with-tensorflow-2-eb0541151049

## Replay Buffer



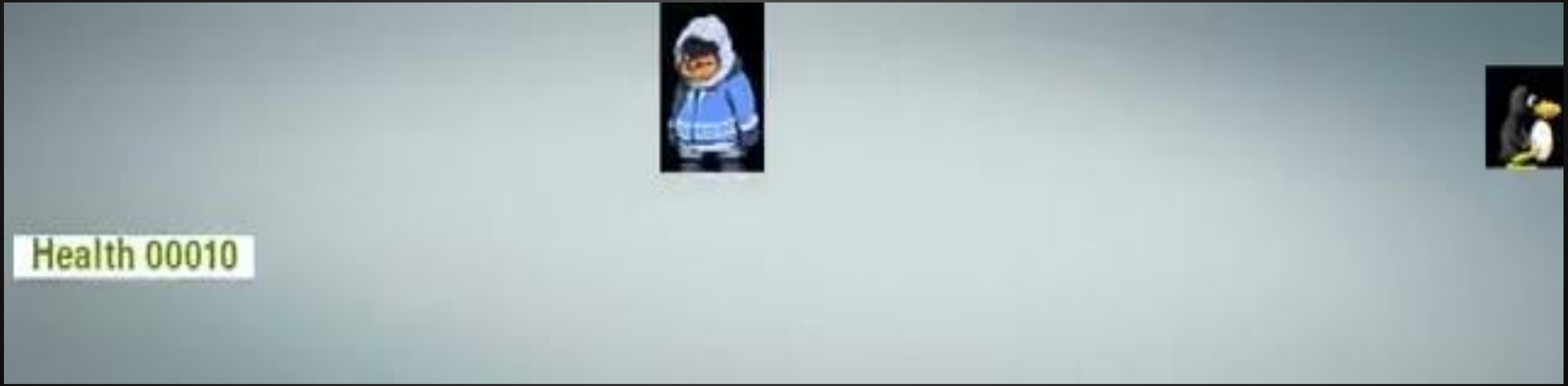
https://levelup.gitconnected.com/dqn-from-scratch-with-tensorflow-2-eb0541151049

## Deep QN + TF Environment + Python Environment

- No screen images used
- Uses 3 discrete state features to train model
- Memory constraints - could not use raw image like OpenAI
- Note that the penguin runs away with clear decisive motion

## CNN + Deep QN

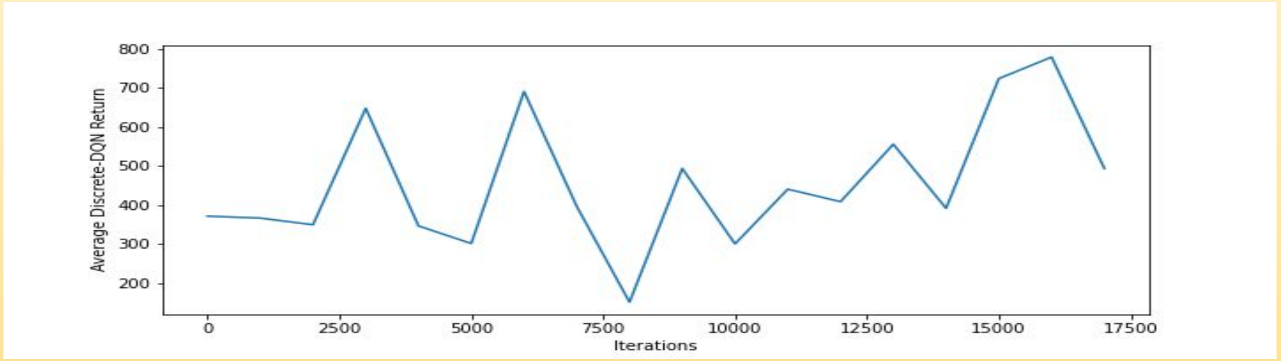
- Reduced training due to memory constraints
- Note that the penguin sprite moves with clear determination, but hasn’t figured out that it is stuck at the boundaries



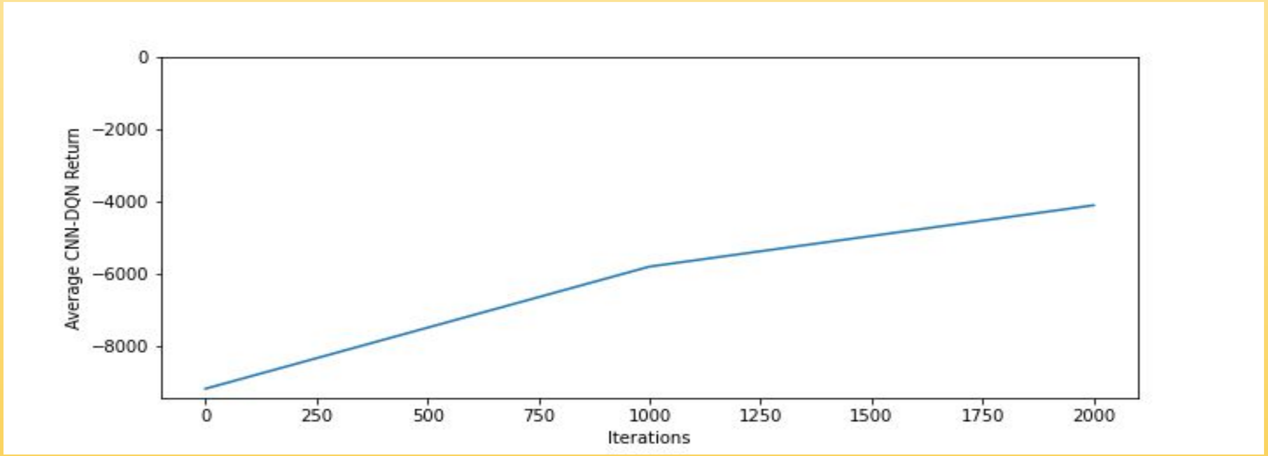
## Average scores

| Model | Baseline Randomized Agent range | Training | Trained model |
|---|---|---|---|
| PPO | 4243 (100 trials) | 20480 time steps/actions | 582  (100 trials) |
| A2C | ~ 5254 (100 trials) | 20000 time steps/actions | 1424 (100 trials) |
| Deep Q Discrete | 13242 (10 trials) | 17000 trials | Trained 665 for 10 trials<br>Gif shows low positive scores |
| CNN + DQN | 11737 (10 trials) | 2000 trials - small number due to memory issues | --6456 over 30 trials<br>Penguin video more positive |

# Average in Training (X coordinate-DQN) and (CNN-DQN)



X coord - DQN
During training, average scores are positive over 17,000 trials.
Average scores are low overall



CNN-DQN
During training, average scores are negative over 2000 trials.
This model needs more work and memory

## Conclusions

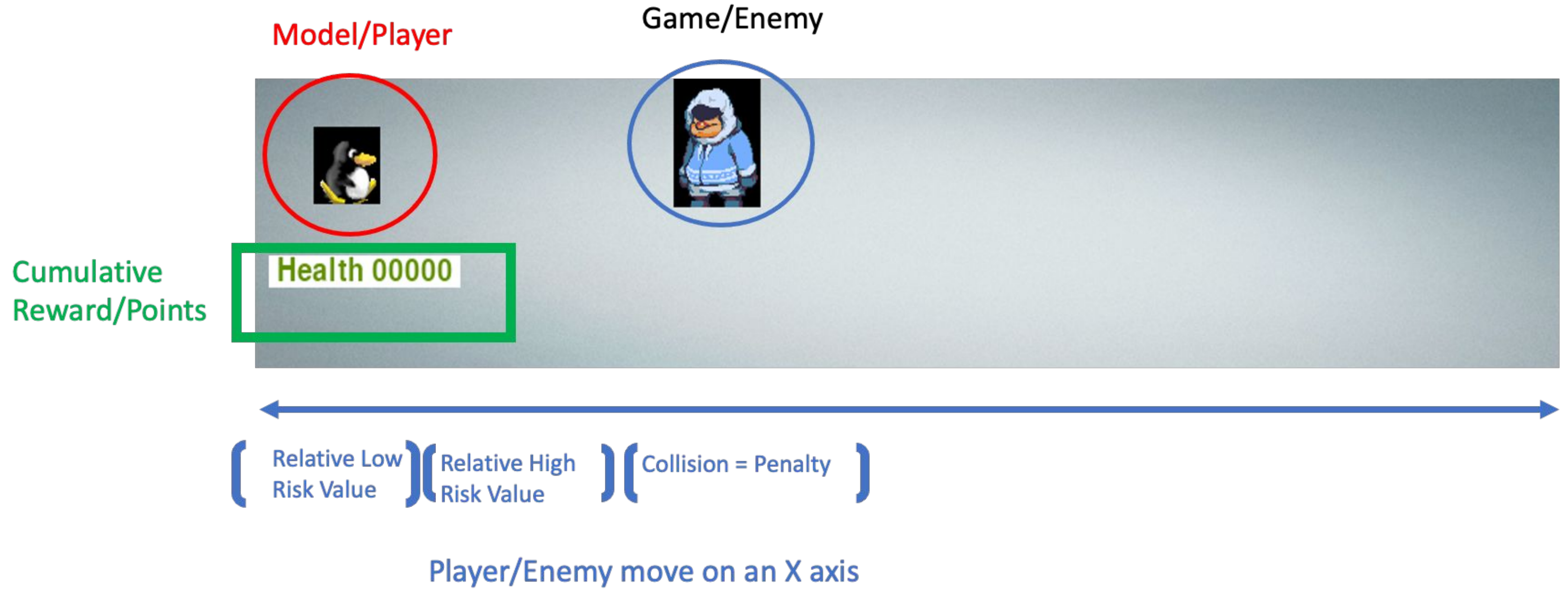
- The models show clear signs of learning, but need more tuning and hardware
- A lot of experimentation was done to adjust rewards
- Average training scores tend to be lower than what you see in actual performance video/gif files
- Most of the work presented is following the tutorials closely
- Hard part is cobbling together the tutorials and customization
    - Open AI Gym: had to create separate python package to install to register environment;
    - weird spec issues with TF agents
- Several Atari Games and Python Games are already supported by different Reinforcement Learning frameworks
- The human learning resembled the progress of the randomized penguin.  It makes you appreciate the amount of work that goes into a prepared lesson

## Recommendations/Next Steps

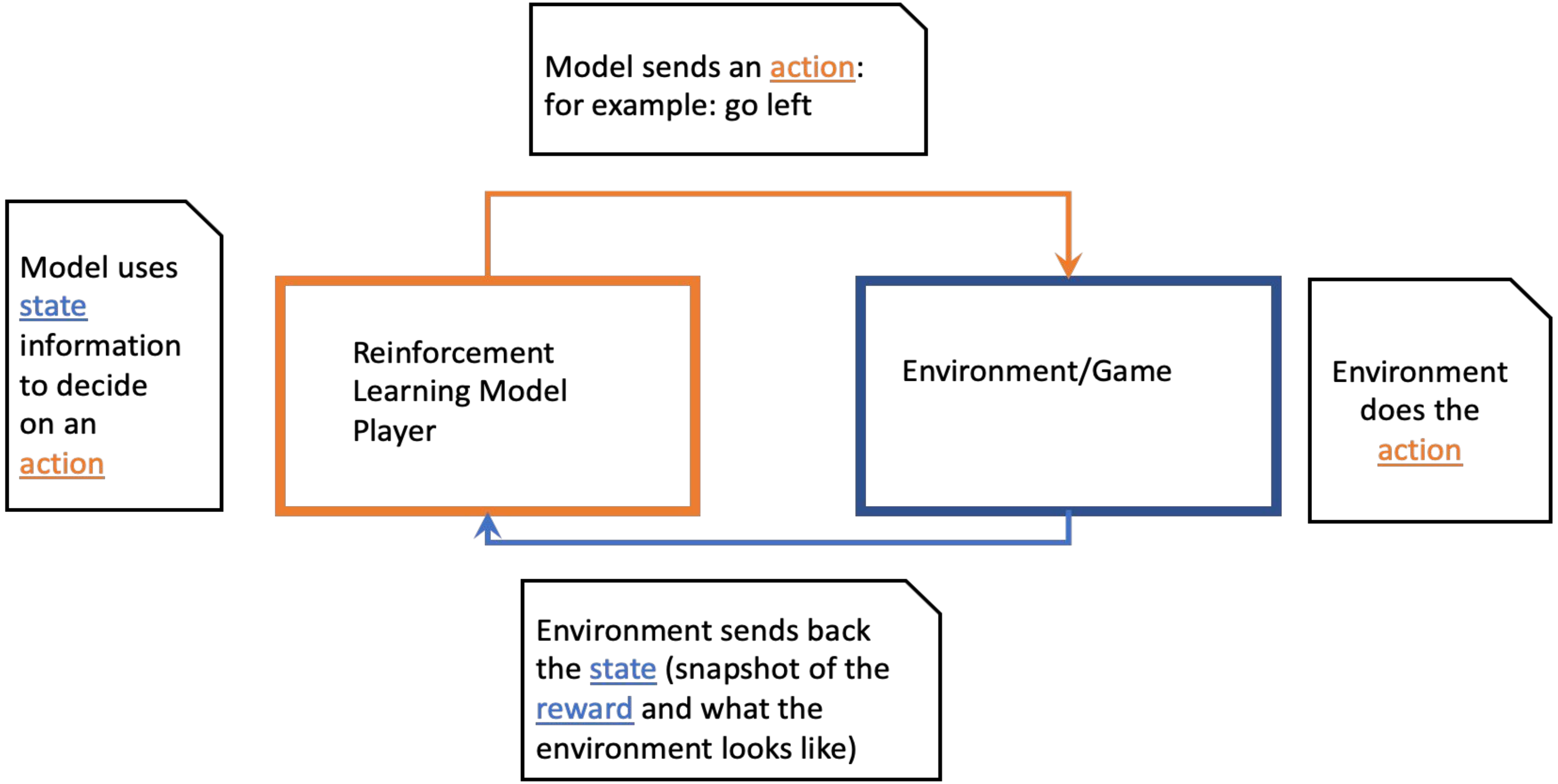
- Phase II: Recommend moving on to more sophisticated applications/games
- Keep working on developing more complex simulations
- Other continuous variable models remain to be tested for user controls such as dials or sliders
- Keep tuning the networks and reward system: learn better strategies for improving model outcomes
- Reinforced learning does show models are able to self-onboard to interpret and respond to a UI and API like a game; however integrating continuous learning is another problem to be addressed

## Appendix

# RL Environment/Simple PyGame



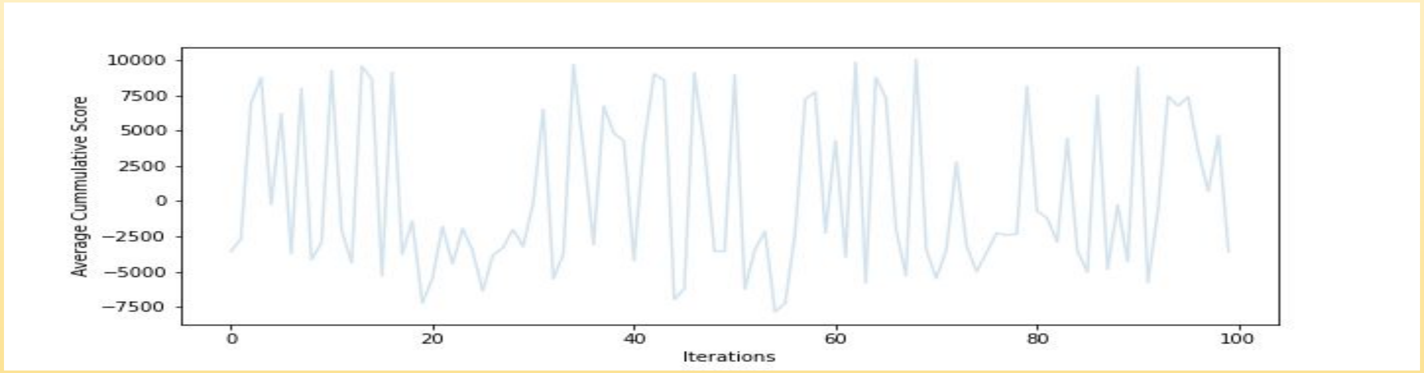
## Reinforcement Learning Cycle of State, Action, Reward


Model sends an action: for example: go left
Model uses state information to decide on an action
Reinforcement Learning Model Player
Environment/Game
Environment does the action
Environment sends back the state (snapshot of the reward and what the environment looks like)

## Some of the attempted training adjustments

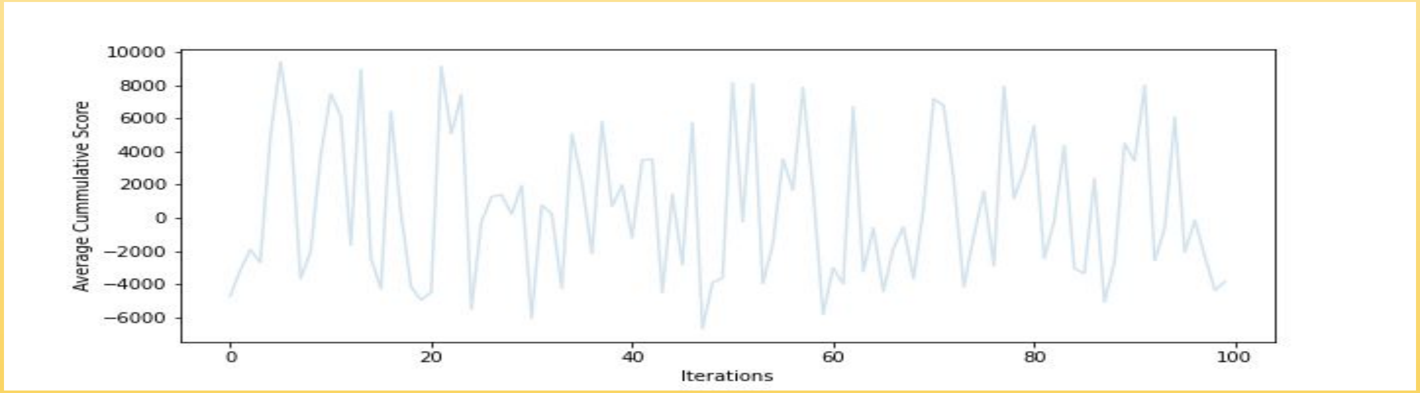
| Type | Description | Outcome |
|---|---|---|
| Rewards | No “risk reward” | Player would lurk on one side completely avoiding “Enemy” |
| Rewards | High collision penalty | Player never recovered and still wouldn’t avoid the “Enemy” |
| Actions | No-op action included | ● Deep Q Models would just sit there without even trying for a reward; ● Other models would still go for the rewarded actions ● Eventually removed this to get DQN to perform better |
| Screen | Player could run off screen | Difficult to assess what was going on. This was removed. Agents didn’t always understand when they stopped at a border. |
| Game Time | Longer games 30 seconds | No significant improvement |

# Average (Before training) PPO



PPO
100 Episodes

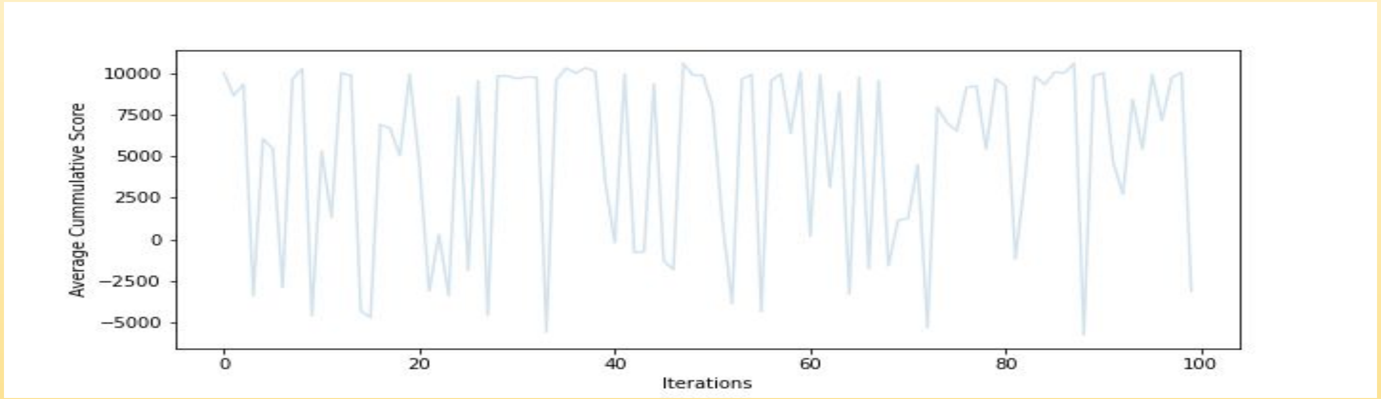
Overall mean
(4243)



PPO (post training)
Overall mean (582)

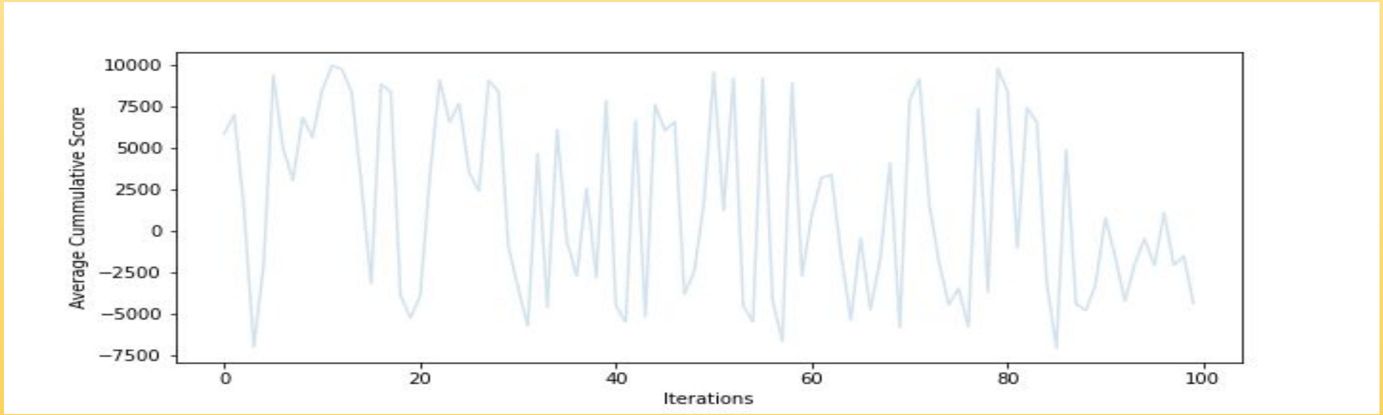
Plot shows more negative scores, so player is spending more time near the “enemy”

# Average (Before and After Training) A2C



Before
100 Episodes

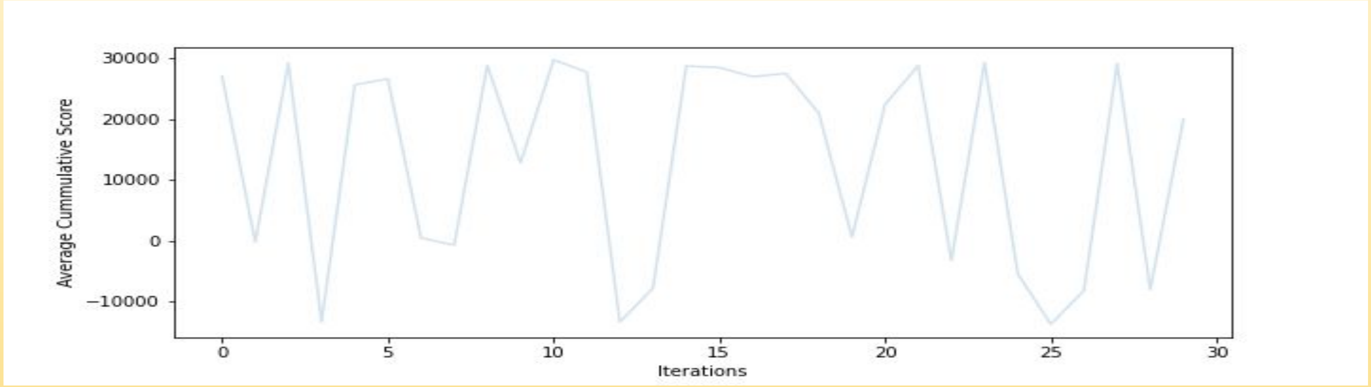
Overall mean (949)



After
100 Episodes
Overall mean (5932)

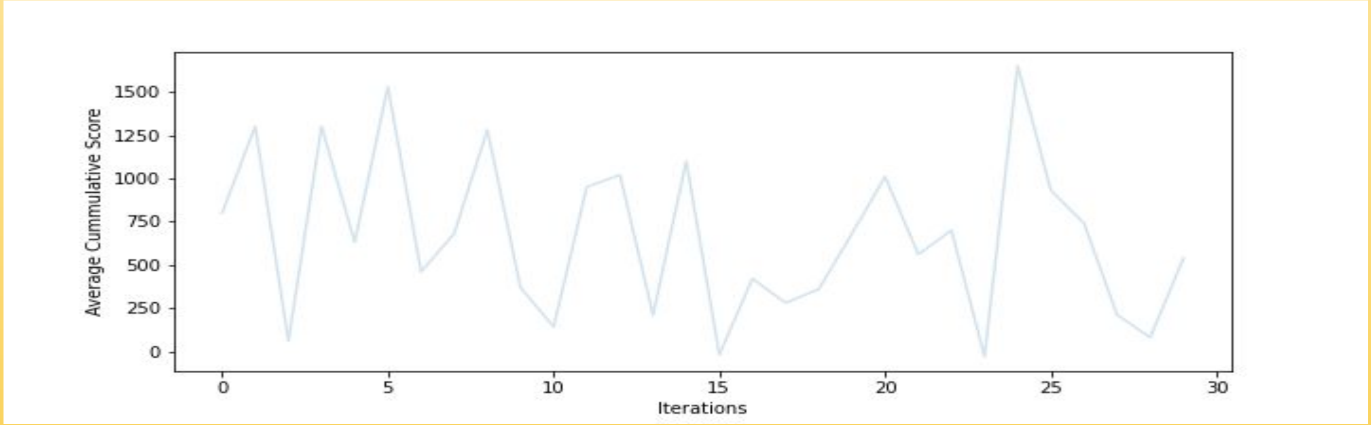
Scores are more positive and in earlier gif, the model is staying closer to the Enemy

# Average (Before and After Training) Coordinate - DQN



Before
30 Episodes

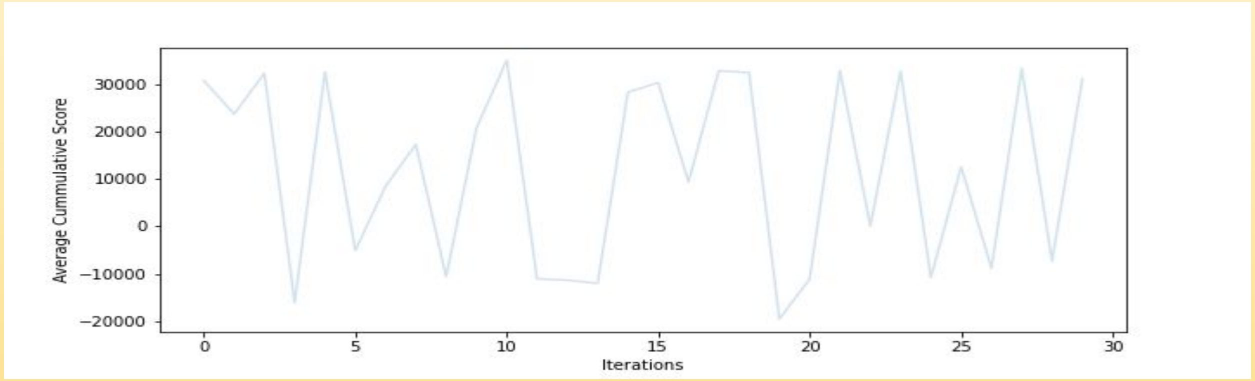
Overall mean
(13242)



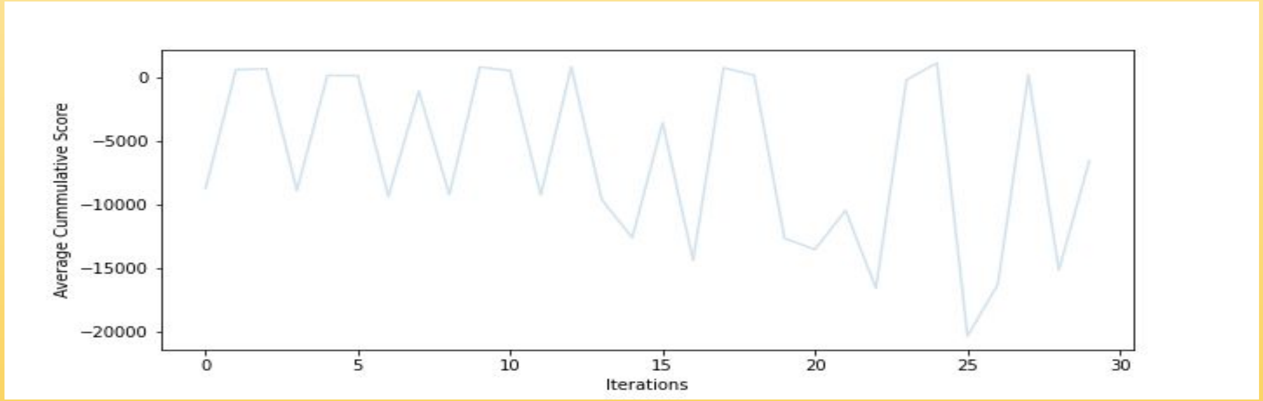
After
100 Episodes
Overall mean (665)

Scores are lower and in the video, the player  heads decisively away from the Enemy

# Average (Before and After) CNN-DQN Training



(Before) CNN-DQN
Only 30 Episodes due to memory constraints
Overall mean



After CNN-DQN
Only 30 episodes due to memory constraints - note the lower scores possibly due to penalties incurred while pursuing higher risk rewards

## Before A2C training, the model spends more time in the lower risk, lower reward area