**Checkpoint #1: Proposal (due Friday, 03/22, 11:00 am)**

**Team Members:** Liza Trundle (emt8kdn), Ann Li (al7gc), Bailey Phillips (bmp5qd), Shelton Kwiterovich (vsk9hw)

**Title**: Enhancing Reinforcement Learning Using DQN for More Advanced Game Playing with Pac-Man

**Motivation and Problem Statement:**

The motivation behind this project is to extend the capabilities of reinforcement learning (RL) models for playing advanced games such as Pac-Man. While RL has shown promising results in simple environments like Lunar Lander in Assignment 3, the challenge lies in extending the scope of A3 to train more complicated RL models to play more advanced games with higher-dimensional state and action spaces. We chose to use Pacman for our game to train in this project, because the commands are more intricate than the lunar lander game (just moving the space ship left, right, up, and down). Pacman includes enemies and more movement commands. It is also constrained in a maze setup that it must navigate around. We want to develop an RL framework capable of training high-quality models to excel in playing advanced games while optimizing resource usage to reduce training costs. In this project, our goal is to build a model that plays Pac-Man and leverage the DQN framework and distributed training with Ray to parallelize computation across many nodes, therefore accelerating training and achieving higher average rewards and improved training stability.

**Approach:**

Our initial approach involves building upon the actor-based RL framework used in Assignment 3 for Lunar Lander. We plan to extend this framework by utilizing more nodes and finely tuned hyperparameters. We will utilize the DQN framework and distributed training setup with Ray. However, we plan to partition the training process across more nodes than in Assignment 3, to parallelize the computation and accelerate training. Additionally, we will further explore techniques for optimizing hyperparameters in a systematic manner. Our experiment design will involve measuring average reward and training stability metrics across different training runs to assess the effectiveness of our approach.

From preliminary research, we need to set up the PacMan environment states to include: position of pac man, maze dimensions, ghosts, pellets, and the unique maze layout. Depending on the preliminary performance of our model, we can easily customize the difficulty of game play with Pac-Man to test the performance of our models. We will still need to do extra research on how many actors we want to use, number of cpus, virtual machines, training episodes, training time, etc.. Some metrics we can look at when evaluating our model is the average reward and training stability. By implementing the DQN algorithm along with Ray, we hope to increase the training quality for a more advanced game.

**Timeline/Expected Milestones:**

1. [3/22] — Checkpoint 1
2. [3/23 - 3/29] Week 1: Set up environment and update DQN to meet the requirements for Pacman
3. [3/30 - 4/5] Week 2: Test the training algorithm distributed on Ray and record preliminary results
4. [4/6 - 4/12] Week 3: Tune hyperparameters and work on visualizing the game

* [4/12] – Checkpoint 2

1. [4/13 - 4/19] Week 4: Finishing touches on the technical component and work on presentation
2. [4/20 - 4/26] Week 5: Work on writeup of results in paper format

* [4/24-4/29] — Project Presentations

1. [5/01] — Project Due

**Preliminary Results:**

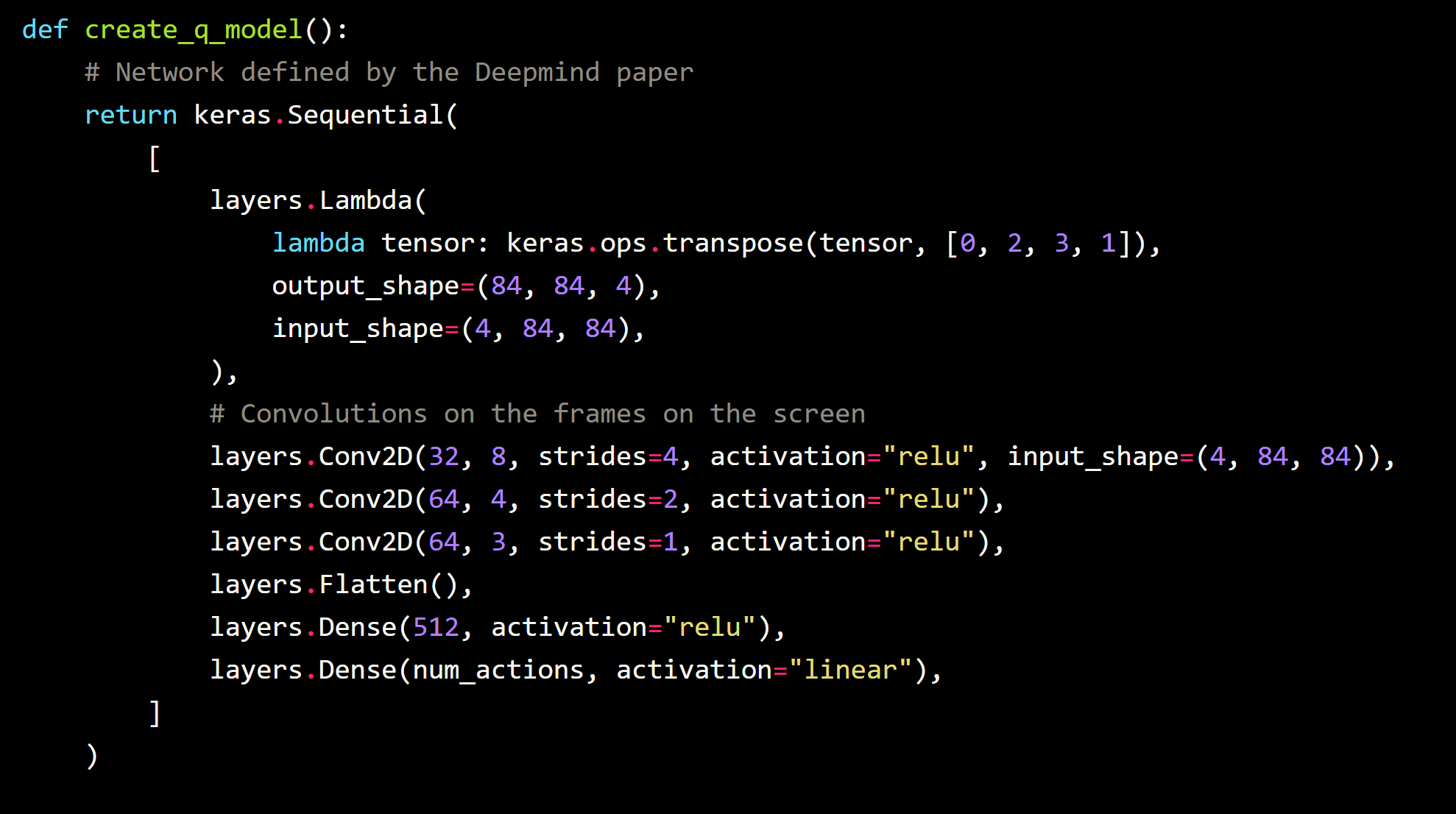
For this project, we are using assignment 3 as our foundation. We haven’t yet started to implement anything for our project, but because of our specific option chosen, we have a good starting point for what a successful model training process can look like. We can use some of the same tactics as A3 such as parameter tuning to ensure that our training is the most efficient and fast it can be.

We have done some preliminary research into the Pac-Man implementation, sources are listed below:

* <https://medium.com/analytics-vidhya/how-to-train-ms-pacman-with-reinforcement-learning-dea714a2365e>
* <https://cs229.stanford.edu/proj2017/final-reports/5241109.pdf>
* <http://ai.berkeley.edu/project_overview.html>
* <https://towardsdatascience.com/deep-split-q-learning-and-ms-pacman-5749791d55c8>

Atari Breakout Resources:

* <https://colab.research.google.com/github/GiannisMitr/DQN-Atari-Breakout/blob/master/dqn_atari_breakout.ipynb#scrollTo=8C5cmoBmvziM>
* <https://keras.io/examples/rl/deep_q_network_breakout/>
* [**https://daanklijn0.medium.com/beating-atari-using-neuroevolution-tensorflow-and-rays-rllib-1619b7330389**](https://daanklijn0.medium.com/beating-atari-using-neuroevolution-tensorflow-and-rays-rllib-1619b7330389) (has some Ray implementation information, but is not specific to breakout and also uses genetic algorithms/neuroevolution)
  + <https://github.com/daanklijn/deep-neuroevolution>
* <https://docs.ray.io/en/latest/rllib/rllib-algorithms.html> ← Ray’s RLib library has a tuned breakout model? (not sure what that entails)
* <https://docs.ray.io/en/latest/rllib/rllib-algorithms.html#dqn>
* <https://medium.com/aureliantactics/using-ray-for-reinforcement-learning-dff2f38a07e3> ← resource on hyperparameter tuning with Ray
* <https://github.com/GiannisMitr/DQN-Atari-Breakout/blob/master/dqn_atari_breakout.ipynb>



Notes on the coding:

TO CHANGE:

* Change QNetwork to use Conv2D (above)
  + How to modify the weights from keras? <https://stackoverflow.com/questions/51354186/how-to-update-weights-manually-with-keras>
* Adapt training method to work with new QNetwork
  + <https://keras.io/examples/rl/deep_q_network_breakout/>
  + <https://docs.ray.io/en/latest/train/distributed-tensorflow-keras.html> ← Ray’s introduction to working with tf/keras models
* Alternatively: use an algorithm/network from RLlib? <https://docs.ray.io/en/latest/rllib/rllib-examples.html>
* Use breakout env. from gym

Actions in breakout= left, right, fire, nothing

Ours uses pixels, other uses flat vectors

Change state size and action size as parameters to the agents

| Observation Space lunar lander | (8,) |
| --- | --- |

| Observation Space [atari](https://becominghuman.ai/playing-atari-using-reinforcement-learning-9fe52fd4f262) | (210, 160, 3) |
| --- | --- |

For using tensorflow:

pip install tensorflow

pip install "ray[rllib]" tensorflow torch

Installation for breakout:

pip3 install 'gym[atari]==0.21.0'

pip install gym[accept-rom-license]

LINK TO JUPYTER NOTEBOOK:

<https://drive.google.com/file/d/1pyPzt6yaeXcuo__YoXF0aY04ywxXGMea/view?usp=drive_link>