PROJECT REPORT

Submitted by:
Mithil Ashish Parmar
014726023

1. Problem Definition:

Previously, there were a few Reinforcement Learning Agents that were used to tackle the problem of solving the games of Atari Suite 2600. This Atari Suite consists of 57 games which have different problems to tackle. A common agent that works in the same fashion for any given problem doesn't give the best possible solution to all the games. These agents do perform well for most of the games as compared to the average human scores but not for some of them. The first ever agent to tackle these Atari suite games was Deep Q-Network or DQN. Newer agents are built on DQN such as R2D2 and Never Give Up or NGU. These were still not able to perform better than an average human in 4 games in particular: Montezuma's Revenge, Pitfall, Solaris, and Skiing. Agent57 is an agent developed by DeepMind to tackle these 4 games as well as the other 53 games and gives better scores than an average human.

This project had taken references from the following citations:

- Mnih, V., Kavukcuoglu, K., Silver, D. *et al*. Human-level control through deep reinforcement learning. *Nature* **518,** 529–533 (2015). https://doi.org/10.1038/nature14236
- Kapturowski, Steven & Ostrovski, Georg & Quan, John & Munos, Remi & Dabney, Will. (2019). Recurrent Experience Replay in Distributed Reinforcement Learning. https://openreview.net/pdf?id=r1lyTjAqYX
- Never Give Up: Learning Directed Exploration Strategies. https://doi.org/10.48550/arXiv.2002.06038
- Agent57: Outperforming the Atari Human Benchmark. https://doi.org/10.48550/arXiv.2003.13350

This project comprises of concepts learned from the courses of ISE 244 AI Tools and Practice for Systems Engineering, CMPE 257 Machine Learning and CMPE 252 Artificial Intelligence and Data Engineering.

2. Project Objective:

The main objective of this project is to replicate the RL agents used to play games in the Atari Suite like Deep Q Networks, R2D2, Never Give Up and Agent57.

Areas integrated in the project and their usage are:

- OpenAI Gym: An API to implement RL agents in Atari environment.
- Python: Programming language used for coding the project.
- PyTorch: Framework used for distributed training.
- TensorBoard: Visualization for ML experimentation and logging.
- Bandit: ML Framework that allows RL agent to select actions for maximizing cumulative reward in the long term.
- Agent Definition, Training and Evaluation .py files.

3. Analysis:

Deep Q Network comprises of an algorithm used to generate Q values or rewards from taking an observation state *S* and an action *a* and putting them through a convolution neural network. The action which generates the maximum Q value is said to have the best results and is considered to be the best policy π. In Atari Suite, we pass the state of a given environment(game) at any instance to the neural network and give the possible actions for that given state which are pre-defined in the Atari Suite to the Convolution Neural Network to get the Q value. The action which gets the maximum Q value is chosen and then the iteration continues for every subsequent state of the game. This is how the agent gets trained for a particular game. This works out for a lot of the games in the Atari suite but for some games, it doesn't perform well. The reason for not performing well on some of the games is because these games have long time horizons. It means that by performing certain actions, you might eventually get more reward instead of performing some other action to get an immediate reward and then stopping. DQN only takes into consideration the immediate reward you're getting after performing an action.

R2D2 is built upon DQN to solve this long-term credit assignment. It incorporates Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) which uses a recurrent neural network instead of a convolution neural network. This allows us to set a given number of steps to LSTM or GRU for checking the history of inputs and actions performed to get maximized reward. An important criterion for an agent to perform well is not only the ability to perform long credit assignments but also have a good exploration-exploitation tradeoff. This means that the agent should be able to choose whether to perform an action which it knows will yield a good reward (exploit) or to choose an action that might lead to some new reward which is not known before (explore). R2D2 has a poor exploration-exploitation strategy as it takes an epsilon greedy approach. It means that there will be a probability of ε to choose an action which might have new rewards and a probability of *1-*ε to choose an action that yields maximum Q value.

NGU implements a better solution for the exploration-exploitation tradeoff replacing the epsilon greedy approach. The main two terms to remember here are intrinsic motivation and curiosity. These terms together are known as novelty. In NGU, we provide a reward *r'* in addition to the reward *r* from the environment to encourage some behavior in the agent. This additional reward is known as intrinsic reward. NGU tries to maximize this intrinsic reward which leads to more curiosity and motivates the agent by rewarding it to find things it has not seen yet. This is still not the best solution because it is difficult to set parameters for the intrinsic reward for each Atari suite game individually.

Agent57 tackles the problem of setting parameters for intrinsic reward for each game dynamically. For each action a, there are two parameters: β and γ. These parameters are controlled by a meta-controller that observes the environment and rewards and dynamically sets these parameters. The state action value function is split up due to these parameters and it becomes:

$$Q(x, a, j; \theta) = Q(x, a, j; \theta^e) + \beta_j \, Q\left(x, a, j; \theta^i\right)$$

The first term is the extrinsic component, and the second part is the intrinsic component of $Q(x, a, j; \theta)$. The value function for this agent in turn becomes:

$$V_{(s_t)} = \sum_{k=T}^{H} \gamma^{(k-T)} r_k$$

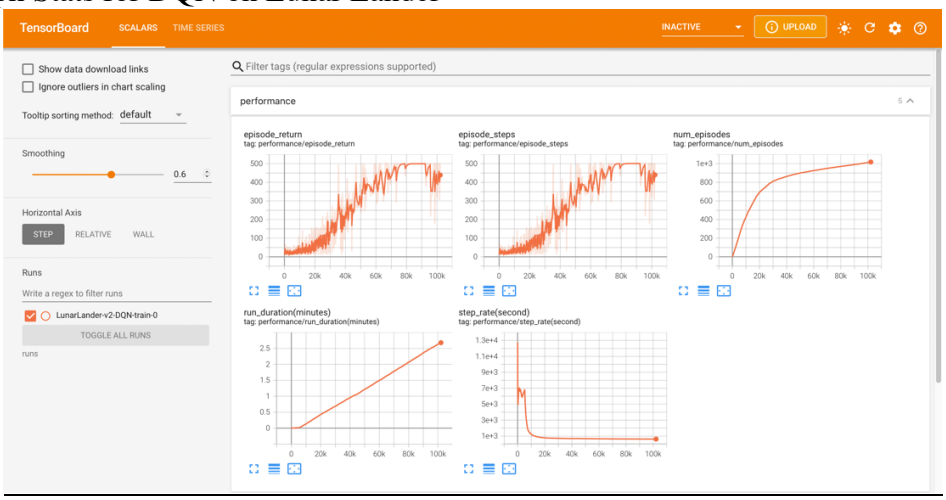
The above equations shows that the value of state s at time t is the summation from time t to a horizon H (which can be considered infinity) of discount factor γ multiplied by reward r.

The β and γ values depend on which type of game it is and where in the learning process you are. If you are at the very beginning of the learning process, you will want to set a very high value for intrinsic reward to encourage exploration and set a low value for discount factor to learn a good immediate value function but as time goes on, you should bring the value of intrinsic reward down and increase the value of discount factor to maximize extrinsic reward and look more into the future which is the end goal. This dynamically adjusting the value of β and γ is known as bandit setting.

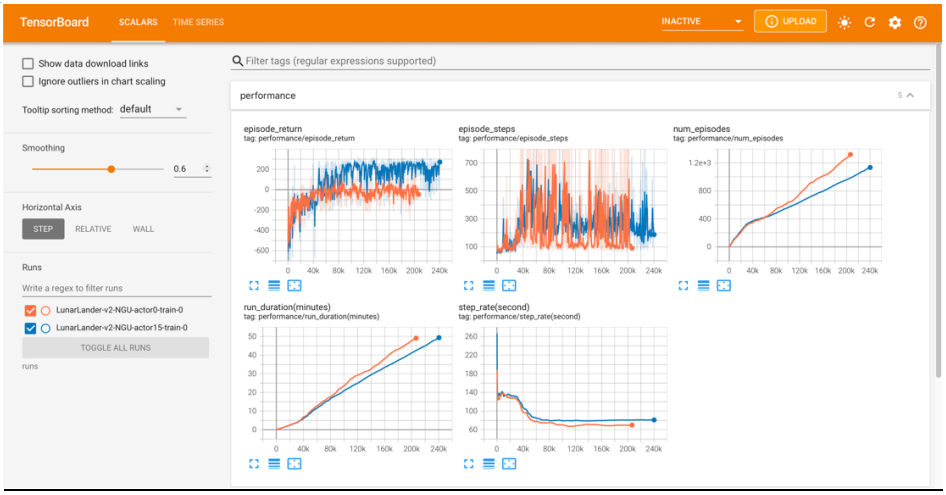
The main assumption made in this project to compare scores is that we are just considering the average human scores for every game. It does not imply that all humans cannot perform better than the given score. Another assumption is that we are training the agents for a limited amount of predefined time.

4. Results:

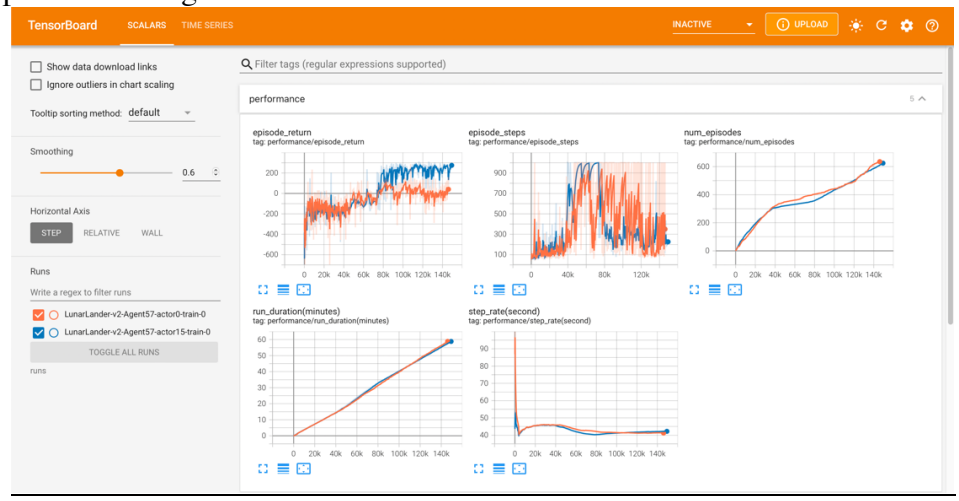
- Graph Stats for DQN on Lunar Lander



- Graph Stats for NGU on Lunar Lander



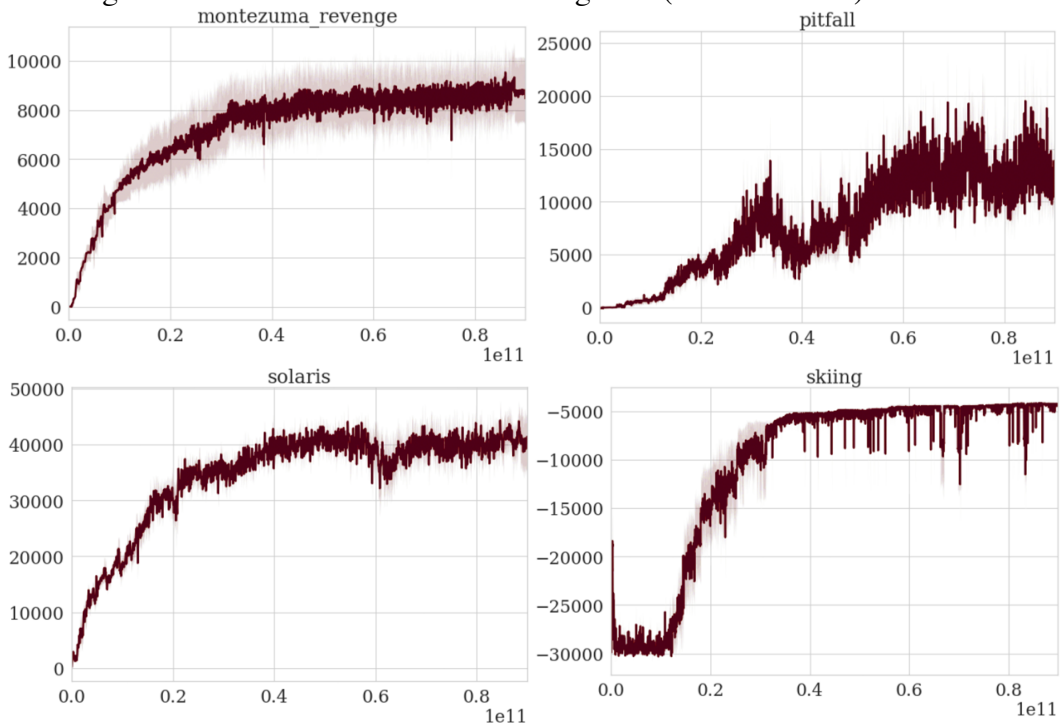
- Graph Stats for Agent57 on Lunar Lander



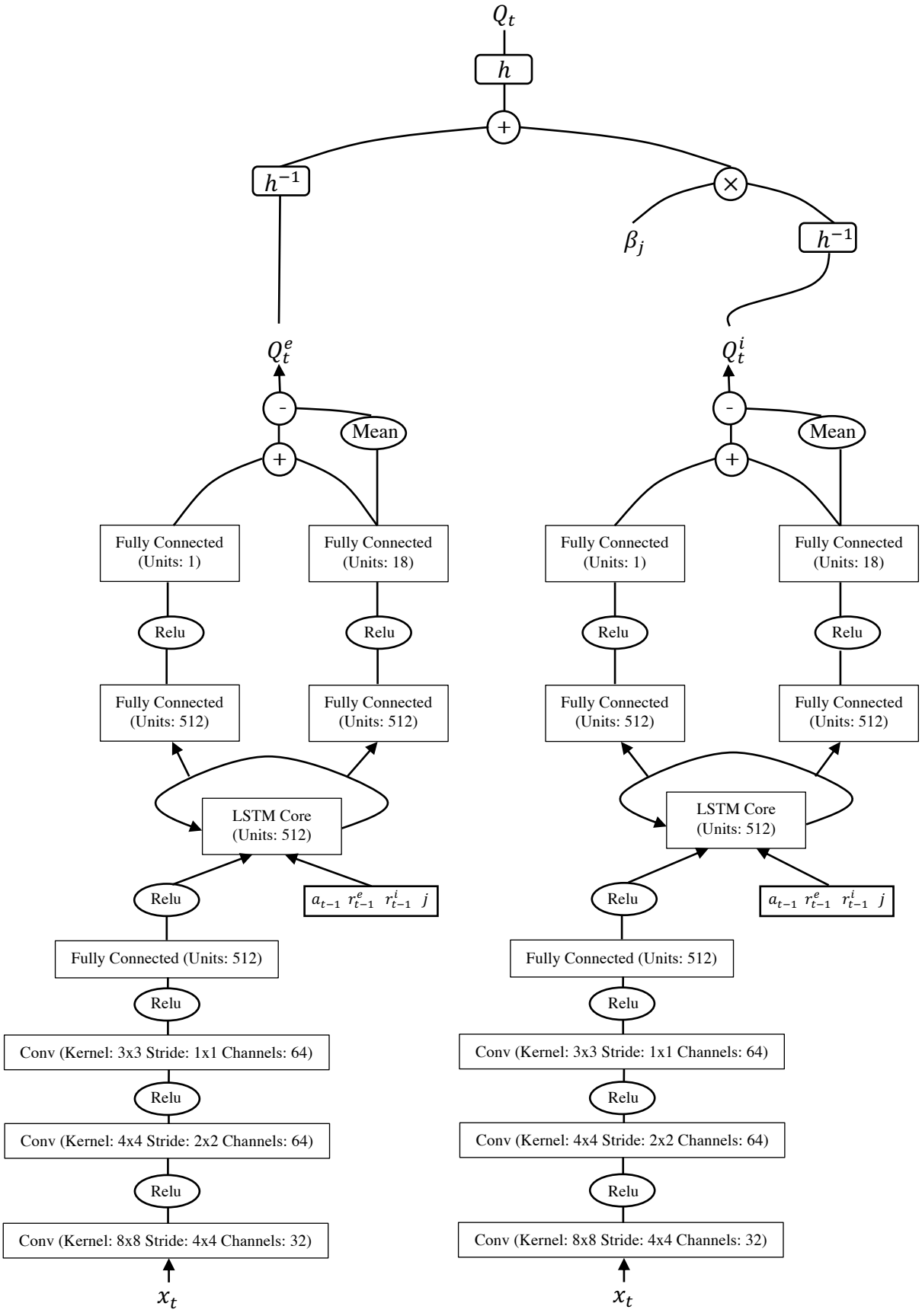
- Scores of Montezuma's Revenge, Pitfall, Solaris and Skiing for different agents:

Games	Average Human	Random	Agent57	NGU	R2D2
Montezuma's Revenge	4753.30	0.00	9352.01 ± 49116.6	**19093.74 ± 12627.66**	2666.67 ± 235.70
Pitfall	6463.70	— 229.4	**18756.01 ± 9783.91**	15334.30 ± 15106.90	0.00 ± 0.00
Solaris	12326.70	1236.30	**44199.93 ± 8055.50**	7254.03 ± 3653.55	11247.88 ± 1999.22
Skiing	— 4336.9	— 17098.1	**— 4202.60 ± 607.85**	— 24271.33 ± 6936.26	— 17797.59 ±866.55

- Learning curves for a few Atari Games on Agent57(scores vs time):



- Architecture of Agent57:



5. Discussion:

Agent57 is the first ever Deep Reinforcement Learning agent to outperform the human benchmark of all 57 Atari Suite games. Agent57 can balance learning of different skills required to outperform diverse set of games in Atari: mainly exploration-exploitation tradeoff and long-term credit assignment. Agent57 is built upon Never Give Up (NGU) agent and the main areas of improvement on NGU were the following three:

- Different parameters of State-Action value function
- Using meta controller to dynamically adapt to novelty preference and discount
- Using longer back propagation through time window to learn the game using retrace algorithm

From the learning graphs of the 4 most challenging games faced by an RL agent, we can see that the longer the agent is allowed to train, the higher score it gets.

6. Evaluation and Reflection:

Although Agent57 performed better than an average human on all 57 Atari games, it doesn't provide the best performance for each and every game as seen in the table of scores for a few games. It shows that for Montezuma's Revenge, Never Give Up agent gives a better score as compared to Agent57. Although these agents perform nicely on all Atari games, it does not imply that humans cannot perform better than them. Anyone can be good at a particular game and can get a high score which could not be beaten by an agent. It is important to note here that if we allow the agent to train for a longer period of time, it may beat this given high score by human as well. Training of these agents to beat the high scores requires a lot of computation power and time. In spite of Agent57 outperforming the human benchmark, it is far from achieving optimal performance for some games.