Evaluation of Exploration Strategies for Deep Q Networks in Atari Pong

CSCE 689 Deep Reinforcement Learning

Team members:

1. Aditya Thagarthi Arun (731009189)
2. Vishruth Raghuraj Gollahalli (730007888)

Github Code link: https://github.com/vishruth-tamu/RL-Final-Project
Video link: https://youtu.be/DFiscBT9_HY

1 Introduction

Exploration strategy for action selection is a very crucial component in any reinforcement learning algorithm. A good strategy should be able to find the right balance between exploitation and exploration for a given environment. In this project we experiment with three different exploration techniques, namely, epsilon greedy exploration, Boltzmann exploration (softmax) and noise based exploration for a Deep Q network that is learning how to play pong in the OpenAI Atari Pong (specifically Pong Deterministic-v4) environment[5]. We will discuss the performance of these strategies along with their shortcomings.

2 Motivation

There exists a very strong need to understand the performance of various exploration strategies in a reinforcement learning setting. A reinforcement learning algorithm, in essence, is one that works on rewarding “good” actions and punishing “bad” ones. To find the aforementioned “good” actions, the agent must search the action space. This search for actions is called exploration. In Atari Pong, the right paddle is trained either to move left, move right, or do nothing (no-op).

At any stage of the game, the right paddle, henceforth known as the agent, must pick an action from the action space of left, right, no-op. To find an optimal action for a particular state, the agent needs to explore the action space and pick an action. The agent is then either rewarded or punished based on the outcome of the action. If the agent picks a good action once, it might pick the same action again and again because it was rewarded at one point in time. What might be a good action for one state may not be so for another. Therefore, to break this exploitation of an action, it is again necessary to have an exploration strategy. Similarly, if an algorithm over-explores, it will never learn what the good actions are for that given state. We aim to study the effects of three exploration strategies on the performance of the Deep-Q network[2] model.

3 Related works

Previously, reinforcement learning has been applied to a set of varied domains, the nature of these domains has been low-dimensional state spaces. However, the Nature paper[2] delves into the ability of agents to analyze and act in high-dimensional states using reinforcement learning, by introducing an artificial agent termed ‘a deep Q Network’(DQN). The DQN is able to combine methods of reinforcement learning with a class of neural networks, specifically deep convolutional neural networks. This paper is crucial for our project as we follow a very similar methodology in the implementation of the standard DQN model. This paper has been very useful for us to implement the DQN model on a multi-dimensional image-based state space in the case of Atari Pong.

Adding noise to weights and biases in a neural network has been expressed as an effective way to promote exploration in the reinforcement learning setting. The NoisyNet[1] is simply a neural network with one or more noisy layers. Noisy layers refers to the layers which alter the weights and biases of those layers with a gaussian noise with a specified standard deviation. We have made use of this architecture in our analysis of the noise-based exploration model.

In this project we use the Boltzman distribution to generate the softmax action probabilities, Cesa-Bianchi and Gentile[4] detail the limitations and the strengths of different Botlzman exploration strategies. This paper provides insight into this strategy and shows us that the standard Boltzmann strategy explores well when beginning rewards take typical values. This knowledge is useful because we do not expect to see non-typical rewards in Pong, so we can expect this strategy to perform well.

4 Methodology

All the frames in the observation space are cropped and processed from an RGB image of dimension 210 X 160 X 3 (figure 1) to a gray scale of dimension 84 X 84 (figure 2) to make the algorithm more computationally efficient. For each state observed, a sequence of three prior frames (replayed from memory) observed along with the currently observed state (a total of four frames) becomes the input to our deep neural network. This treatment ensures that the sequential nature and Markovian[10] property are preserved.

In our implementation of the Deep Q Network (DQN) we opted to use the architecture proposed in the DQN Nature paper[2]. The vanilla implementation of the network includes an input layer of dimension 4 X 84 X 84, 3 convolutional layers, a fully connected layer and output layer of dimension equal to the number of actions (3 in our case). The pong environment supports 6 actions for the right paddle but we only choose 3 (no-op,left and right) of those 6 actions as they are the most meaningful actions for pong and our experiment.

Training is done in batches of 32. During training, a state is randomly sampled from memory. For each state, an action is picked using an exploration strategy, this gives us the reward observed and the next state. We get the prediction of Q values for the current state from our network. The Q value of the state and the selected action is then updated as per the Bellman equation shown in equation 1. The model is finally fitted to these updated Q values for the 32 states in the batch.

$$Q(s,a) = r(s,a) + \gamma max_a Q(s',a)$$

Equation 1: Bellman Q value update

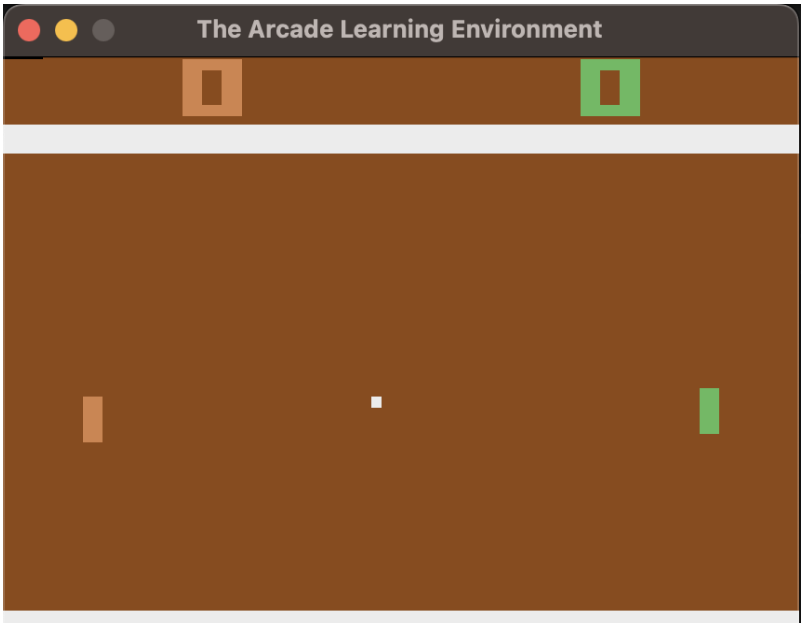

Figure 1: Image of state before preprocessing

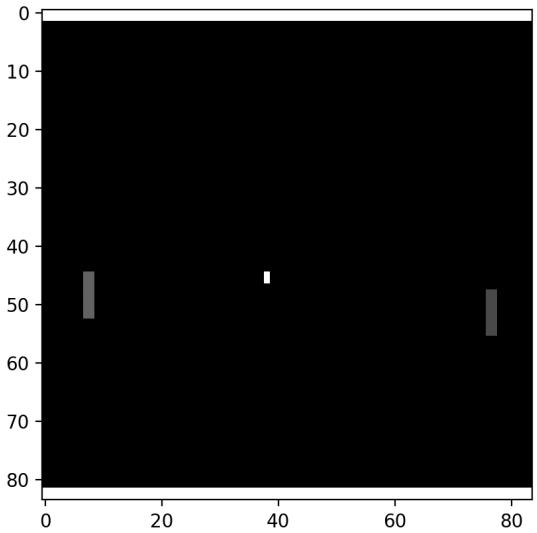

Figure 2: Grayscale image of the state

4.1 Epsilon greedy action selection

In the epsilon-greedy policy the agent chooses a strategy where it picks the action with the best state-action value with a probability of 1 - ε (ε is a hyperparameter) and with probability ε, the agent picks an action from the action space randomly in search of a suitable action. Once this action is picked, the episode continues and a reward is given out in the end. If this reward is indeed favorable, the state-action value associated with the selected action will increase. Epsilon-greedy strategy is a simple exploration strategy that does not require a lot of complicated math or logic to implement. Our basic treatment of Atari Pong is one using this exploration strategy. In general, the epsilon greedy policy looks like:

```
initialize ε
rn = np.random.random()
if rn < ε:
    sample an action from action space
else:
    select greedy action
```

Figure 3: Epsilon-greedy algorithm

As the Epsilon greedy strategy was used in the original Nature paper[2] we use it as a benchmark to compare implementations of other exploration strategies which we present below.

4.2 Softmax action selection

Even though the epsilon greedy approach is a robust and simple way to have a balance between exploitation and exploration, it suffers from a few drawbacks. When the epsilon greedy approach chooses to explore, it chooses the action in a random manner where each action has an equal probability of being picked. In cases where the worst action is very detrimental to the agent this approach may not be the best choice. The softmax action selection[4] approach overcomes this shortcoming by generating action probability as a result of a graded function.

Under the softmax action selection strategy the best action is still given the highest probability but the other action probability are weighted according to the values they present. The Boltzman distribution (equation 2) is commonly used to generate these action selection probabilities. To implement this in our project, we make use of the softmax function found under the tensorflow.nn module, this takes the vector of Q values outputted by our model and passes it through a separate dense layer that uses the softmax activation function, this gives us the softmax action probabilities. The action is then selected based on these action probabilities.

Additionally, a temperature parameter (τ) is used to adjust the greediness of this strategy, high temperatures make the strategy select actions in an almost equi-probable manner and a temperature equal to zero makes the approach equivalent to a greedy policy. The probability of an action *i* being selected is calculated as[8]:

$$p(i) = \frac{e^{Q(i)/\tau}}{\sum_{j=1}^{n} e^{Q(j)/\tau}}$$

Equation 2: Boltzmann distribution

4.3 Noisy deep q learning (Gaussian noise layer)

Adding noise to a Deep-Q Network is a fairly new idea[1]. Neural network noise may be either parametric or action noise. Action noise is where the noise is added to the final output of the neural network but parameter noise is when noise is added to the parameters (weights and biases) of the neural networks itself. The former is the one we deal with in our experiment. We have chosen parameter noise as it is more effective in promoting exploration than action noise[3]. Figure 4 shows visually what each of the noise types looks like[3]

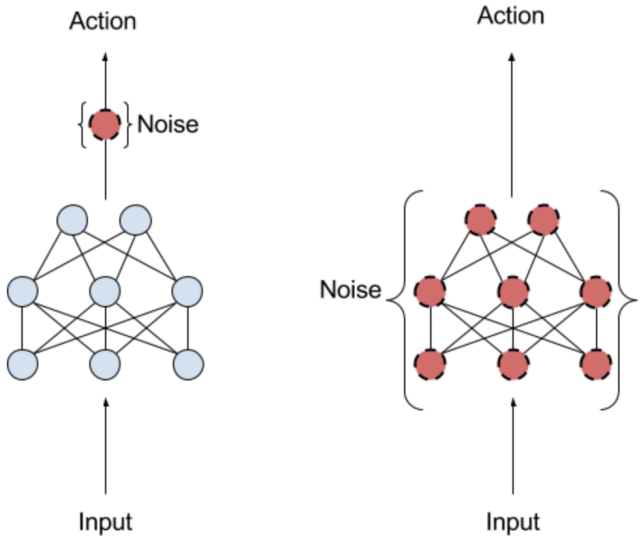

Figure 4: Noise Types in Noisy Deep Networks

We implemented a noisy layer with a Gaussian noise as part of our neural network using a Gaussian Noise function. The Gaussian noise function is defined in equation 3.

$$\frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-(z-\mu_n)^2/2\sigma_n^2}$$

Equation 3: Gaussian Noise Equation

Equation 3 uses standard notations where μ is the mean and σ is the standard deviation.

As parameter noise is applied to the weights and biases during training this would also aid in avoiding overfitting of the network apart from allowing better exploration. This parameter noise was added using keras' GaussianNoise layer which is a regularization layer that is active only during runtime. It is simply a regular dense layer with noise added to the parameters within that layer and is connected to the final output layer in our network. The action was selected in a greedy manner as opposed to epsilon greedy in the noisy network setting.

The data collected in our study of these exploration strategies are visualized in the following section.

5 Experiment

To evaluate the exploration strategies explained above, we ran the python scripts for each of the exploration strategies on the TAMU High Performance Research Computing (HPRC) nodes. Total training time was 36 hours for all of the models on a single node single core system on the HPRC cluster. The table below shows the training data:

Model	Training Time	Hardware Used	Episodes Completed	Notes
Epsilon-greedy	36 hours	1 node 1 core	256	Starting ε = 1 Decay rate = 5e-6 Minimum ε = 0.05
Softmax-02	36 hours	1 node 1 core	262	Temperature = 0.2
Softmax-05	36 hours	1 node 1 core	346	Temperature = 0.5
Softmax-decay	36 hours	1 node 1 core	176	Temperature = 0.2 Decay rate = 1e-2 Minimum temperature = 0.02
Noisy-001	36 hours	1 node 1 core	261	Noise standard deviation = 0.01
Noisy-002	36 hours	1 node 1 core	351	Noise standard deviation = 0.02

Table 1: Training Data

The models, although trained for the same amount of time, trained for different numbers of episodes. This gave us more insight as to the performance of the different models. Since the models were trained in an image-based environment, it takes a very long time to train the agent

to completion. We therefore present our findings on partially trained models. Even though the agent is not completely trained, it still gives us significant insights about the performance and efficiency of the different exploration strategies. We would like to emphasize that the training was capped not by the number of trained episodes but by the training time. Additionally, a discount factor of 0.99 was chosen for all the experiments conducted. We now present the analysis on the performance of the models one at a time and finally provide a holistic comparison of the best models in each category.

5.1 Epsilon greedy action selection

We first present the implementation of the epsilon-greedy policy. We have used an initial ε value of 1. From the above definition of epsilon-greedy algorithm, an ε value of 1 would mean that an action would be sampled from the action space 100% of the time, i.e, it promotes exploration fully. For the first few episodes, this would be preferred since the agent is only just getting to know the environment. It makes no sense to exploit any action that early in the training process.

Once a few episodes have passed, we gradually reduce ε[11] till it reaches a minimum value. This minimum ε we have defined to be 0.05. At the minimum value, the agent would explore the action space 5% of the time and exploit the best action the remaining 95% of the time. This makes sense because near the end of training the agent has a good idea what the state-action values would be. We have set the decay rate of ε to be 5e-6 per episode.

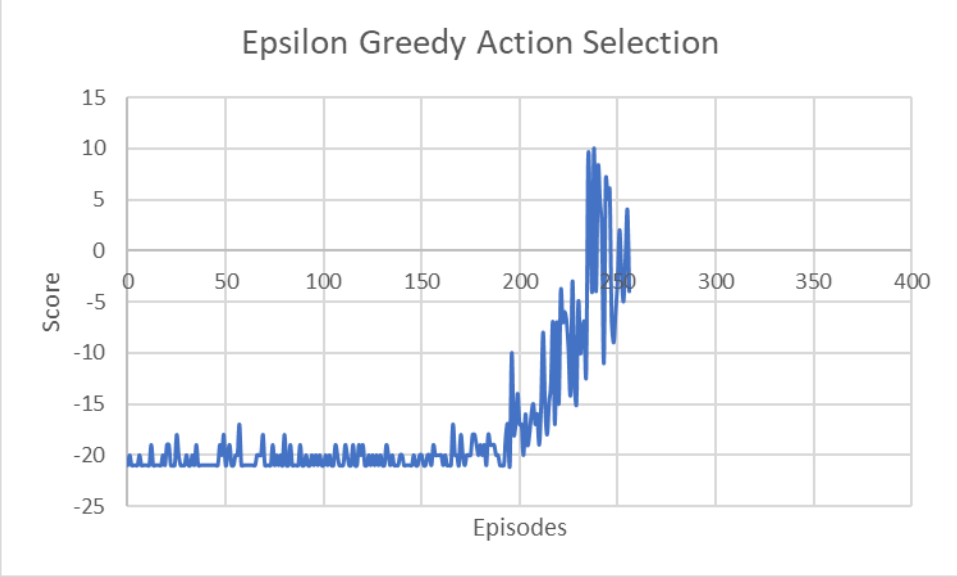

Figure 5: Epsilon-greedy exploration performance

The Deep-Q Network epsilon-greedy exploration strategy completed 256 episodes in 36 hours of training time. The maximum score it reached was 10.

5.2 Softmax action selection

For these experiments we initially ran with temperature 0.2 and 0.5 (intelligent guesses for trial and error), the performances we observed were not satisfactory for either of the runs using the fixed temperature. We then tried the technique where we dynamically decay the temperature by a factor of 1e-2 for each episode that the model trains, this in fact gave us the best results. This is because it is a good idea for an algorithm to become more greedy as it learns, and it is better for the algorithm to stay explorative in the initial part of the training. Decaying the temperature accomplishes exactly that. The pong scores of the experiments can be seen below.

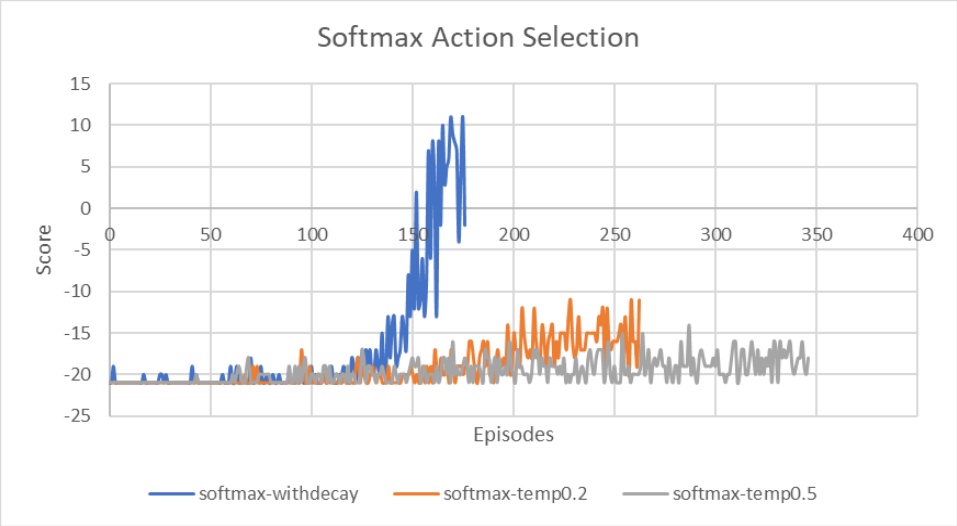

Figure 6: Softmax Model Performance

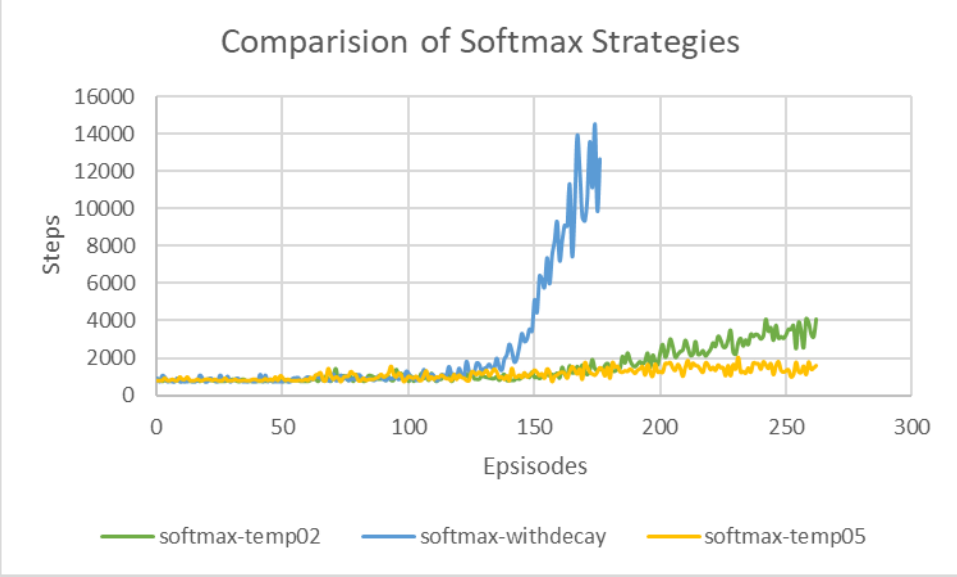

Figure 7: Steps vs Episodes for Softmax Experiments

From Figure 6 we can see that the strategy employing the temperature decay technique trains for a fewer number of episodes but achieves much better results than the strategies using a fixed temperature for generating the softmax action probabilities. Softmax action selection with temperature decay does significantly more steps per episode as it trains. Figure 7 shows that the agent is doing a better job at learning how to play pong. The number of steps in the episode can be seen to be rising much faster, suggesting that the agent is playing each game for longer periods of time.

5.3 Noisy Deep-Q Network

We have used a Gaussian Noise layer as an exploration method. The GaussianNoise layer in Keras was added to the deep-Q network with mean 0 and standard deviation 0.02 and repeated the training process once again with standard deviation of 0.01 to identify what the effect of the noise standard deviation would have on the training quality and rewards. These standard deviation values were picked to be small values as too much noise would negatively impact the trainable parameters of the neural network. Thus, we used 0.01 and 0.02 as sufficiently small standard deviation values.

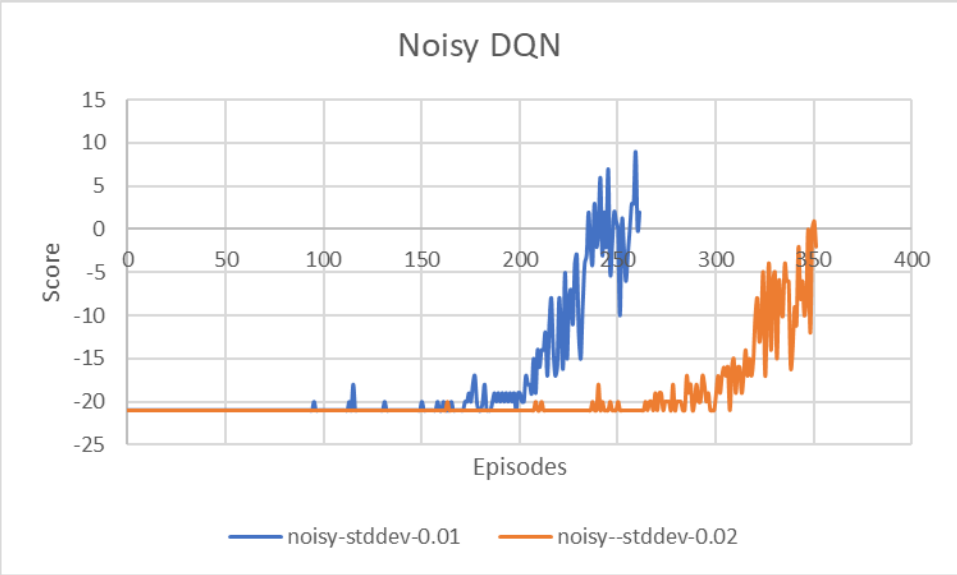

Figure 8: Noisy DQN performance

Figure 8 shows the comparison of the Noisy DQN model with the 0.01 and 0.02 standard deviation values for the noise parameter. It is clear that the DQN model with 0.01 standard deviation (Noisy-001) outperformed the DQN model with 0.02 standard deviation (Noisy-002). Noisy-001 ran for much fewer episodes than Noisy-002 but still managed to achieve a better score at a much faster rate. Noisy-001 achieved its best score nearly 100 episodes before Noisy-002.

5.4 Comparison of the exploration Strategies

Exploration Strategy	Maximum Score Achieved
Epsilon-greedy	10
Softmax-withdecay	11
Noisy-001	9

Table 2: Comparison of Maximum Scores achieved by various models

After performing the experiment on the different exploration strategies, we identified the Noisy-001 and Softmax-decay models as the best performers in their respective categories. We used the epsilon-greedy based model as the benchmark for comparison of the exploration strategies.

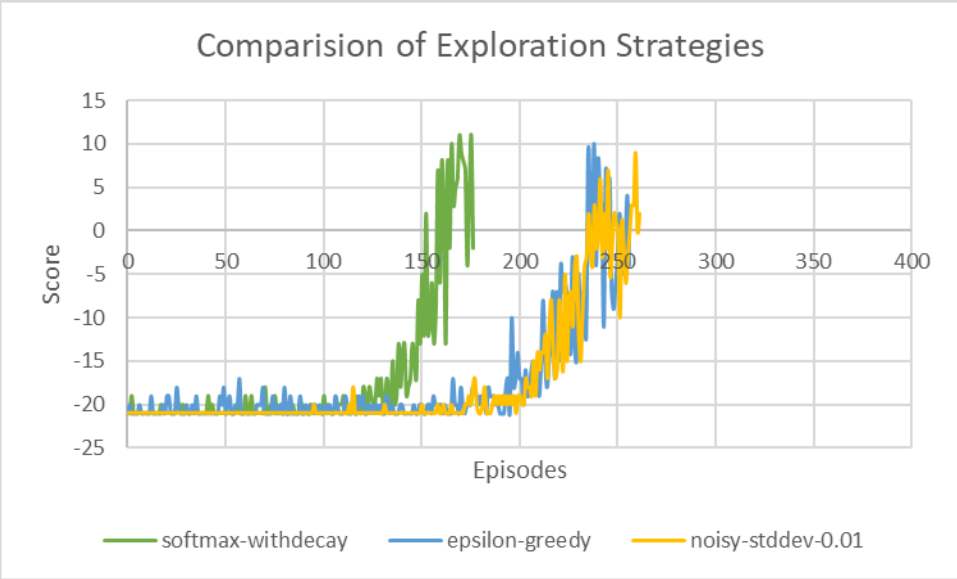

Figure 9: Comparison of Exploration Strategies

In comparison to the epsilon-greedy model, the Noisy-001 model performed very similarly, with both reaching the same average scores at nearly the same episode number. Further, the number of episodes run during training for these two models were also very similar at 256 and 261 respectively. In contrast, the Softmax-decay model performed better than the other two, reaching the highest maximum score in the shortest number of episodes. It achieved this at nearly a hundred episodes fewer than the epsilon-greedy and the Noisy-001 models.

6 Conclusions

Upon the analysis on the performance of the three exploration strategies in the Deep-Q Networks all the strategies are able to beat Pong comfortably. We found that among the noise-based exploration, a standard deviation of 0.01 did considerably better than the model

with standard deviation 0.02. This gives us an insight into the effect of standard deviation of the noise function on training of Deep-Q networks and that it plays an important role in determining the efficiency of the Deep-Q network.

Among Softmax strategies, the softmax model with temperature decay performed the best in comparison to fixed temperature softmax models.

Among the best performing models, it was evident that the softmax model with temperature decay outperformed the epsilon-greedy benchmark model as well as the Noisy-001 model by a significant margin. This is due to the difference in sampling of actions between the softmax-decay and the epsilon-greedy strategies. In epsilon-greedy strategy, the actions are sampled with equal probability but in the case of softmax-decay strategy, the actions' probabilities are determined by the softmax function and therefore the sampling of actions happen with varied probabilities. Specifically, in the case of softmax with temperature decay, the worst action(s) is(are) assigned very low probabilities, and so, when actions are sampled, these inferior actions are hardly ever chosen. We believe the softmax exploration strategy performs better because the penalty for performing the worst action is very high in Pong.

In conclusion, the impact of sampling of actions cannot be understated in a reinforcement learning algorithm and the softmax exploration with temperature decay offers the best exploration strategy among the ones we have studied for this project for the Atari Pong environment.

7 Future work

One of the shortcomings of our project is that we have not performed sufficient hyperparameter tuning for the experiments. We selected the parameters for standard deviation of the noise function and the temperature for the softmax function without carrying out much tuning to identify the best values.

Another shortcoming is the runtime for training. We capped the training time at 36 hours due to lack of time to train until completion. The vanilla form of DQN we have implemented is known to be sample inefficient, hence it can use a more efficient way of using the replay memory such as prioritized experience replay to further improve upon sample efficiency and performance.[9]

As part of future work on this project, we wish to improve upon these shortcomings. Carrying out hyperparameter tuning would help in understanding the impact of exploration strategies in a more pronounced manner. Further, through longer training time, we can analyze the performance of the exploration strategies fully and draw insight from a better perspective than what we had for this project. Through this project, we realized that the analysis of exploration strategies in reinforcement learning is still a fairly new study and that there is a lot of scope in this domain of reinforcement learning.

References

[1] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., & Legg, S. (2017, June 30). Noisy Networks for Exploration. arXiv.org. Retrieved December 2, 2022, from https://arxiv.org/abs/1706.10295v1

[2] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). Retrieved November 5, 2022, from https://doi.org/10.1038/nature14236

[3] Plappert, M. (2020, June 29). *Better exploration with parameter noise*. OpenAI. Retrieved December 3, 2022, from https://openai.com/blog/better-exploration-with-parameter-noise/

[4] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. 2017. Boltzmann exploration done right. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6287–6296.

[5] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016, June 5). Openai Gym. arXiv.org. Retrieved December 2, 2022, from https://arxiv.org/abs/1606.01540

[6] Tokic, Michel. “Adaptive epsilon-Greedy Exploration in Reinforcement Learning Based on Value Difference.” KI (2010).

[7] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016, February 25). Prioritized experience replay. arXiv.org. Retrieved November 5, 2022, from https://arxiv.org/abs/1511.05952

[8] Softmax Action Selection. Incomplete Ideas. (n.d.). Retrieved December 3, 2022, from http://incompleteideas.net/book/ebook/node17.html

[9] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016, February 25). Prioritized experience replay. arXiv.org. Retrieved November 5, 2022, from https://arxiv.org/abs/1511.05952

[10] L. Rabiner and B. Juang, "An introduction to hidden Markov models," in IEEE ASSP Magazine, vol. 3, no. 1, pp. 4-16, Jan 1986, doi: 10.1109/MASSP.1986.1165342.

[11] Maroti, A. (2019, October 30). RBED: Reward based Epsilon Decay. arXiv.org. Retrieved December 3, 2022, from https://arxiv.org/abs/1910.13701

[12] A. D. Tijsma, M. M. Drugan and M. A. Wiering, "Comparing exploration strategies for Q-learning in random stochastic mazes," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1-8, doi: 10.1109/SSCI.2016.7849366.