Exploration strategies for DQN algorithm

Kamil Łapiński

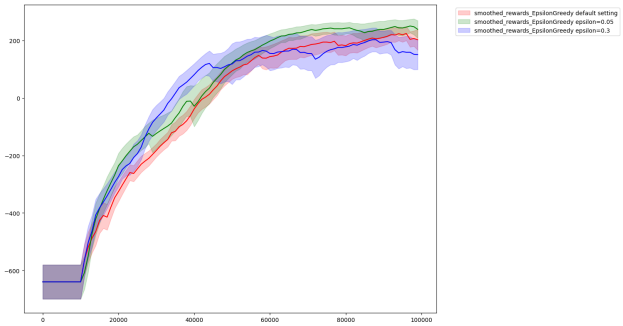
Note 1: Implementation of tasks with some of the plots are available in the attached Google Colab Notebook (here). Note that the code for investigation of hyperparameters and plots generating are commented out in the notebook to avoid long execution times.

Note 2: I also created a separate notebook (here) for more plots. It is based on calculations from the main notebook and it's nothing new there, just more visualizations. Sorry for making it complicated, but for me it was much easier due to long execution times of the main notebook and google colab limitations.

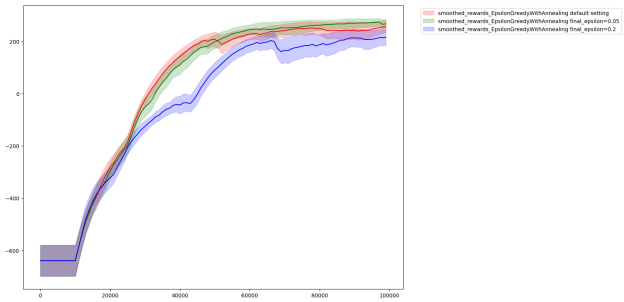
1 Investigation of parameters

In this part, I present the results of hyperparameters tuning for the DQN strategies. There are plots for bonus strategies presented also, you can see section 5, to understand how they work.

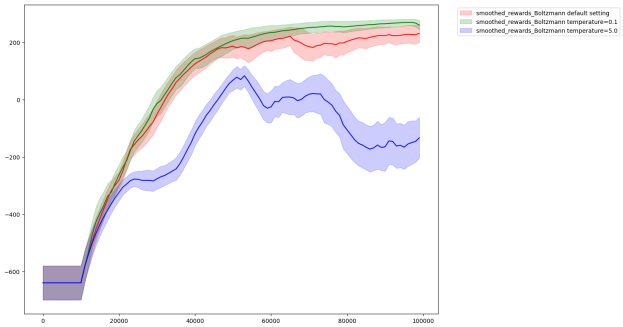
1.1 Plots



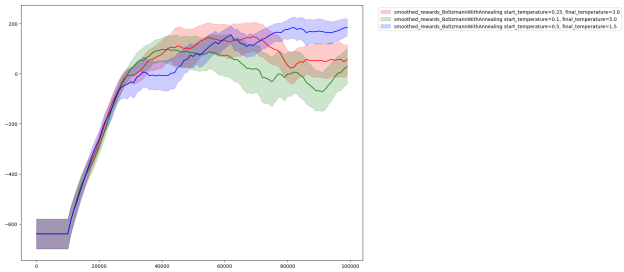
(a) Epsilon-Greedy



(b) Epsilon-Greedy with Annealing

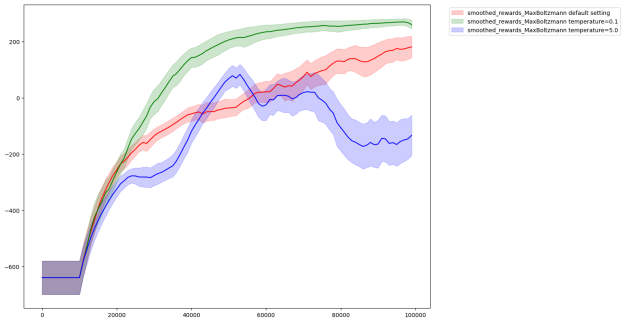


(c) Boltzmann

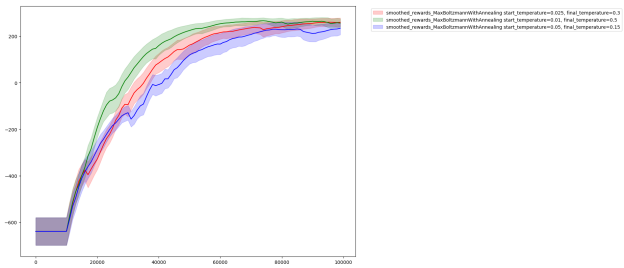


(d) Boltzmann with Temperature Annealing

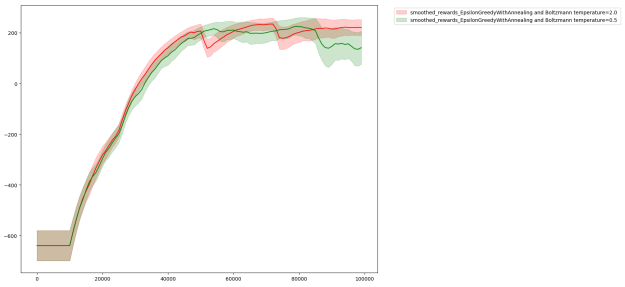
Figure 1: Hyperparameters analysis of DQN exploration strategies in LunarLander-v3.



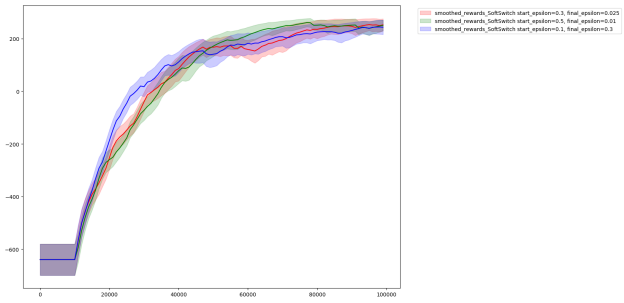
(a) MaxBoltzmann



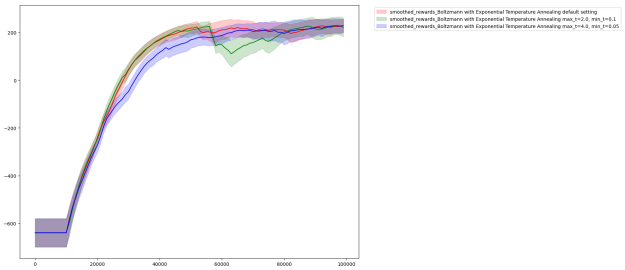
(b) MaxBoltzmann with Temperature Annealing



(c) TwoHalves



(d) SoftSwitch (Bonus 1)5.1



(e) Boltzmann with Exponential Temperature Annealing (Bonus 2)5.2

Figure 2: Hyperparameters analysis of DQN exploration strategies in LunarLander-v3.

We can see, that in most cases, lower values of ϵ for Epsilon-Greedy strategies and lower temperatures for Boltzmann strategies perform better. It shows that in the environment of LunarLander-v3, it's more beneficial to exploit known good actions rather than explore new ones.

1.2 Analysis

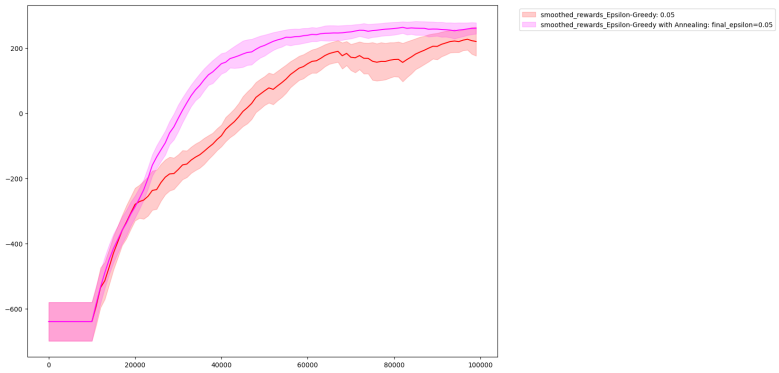
Now, we can choose the best hyperparameters for each strategy based on the plots presented in figures 1 and 2. In some cases, it is not straightforward to determine the best hyperparameters, but I choose the following hyperparameters for each strategy as better than others:

- Epsilon-Greedy: $\epsilon = 0.05$
- Epsilon-Greedy with Annealing: $\epsilon_{final} = 0.05$ ($\epsilon_{start} = 1.0$, $a_{steps} = 30000$)
- Boltzmann: $t = 0.1$
- Boltzmann with Temperature Annealing: $t_{start} = 0.5$, $t_{final} = 1.5$ ($a_{steps} = 30000$)
- MaxBoltzmann: $t = 0.1$
- MaxBoltzmann with Temperature Annealing: $t_{start} = 0.01$, $t_{final} = 0.5$ ($a_{steps} = 30000$)

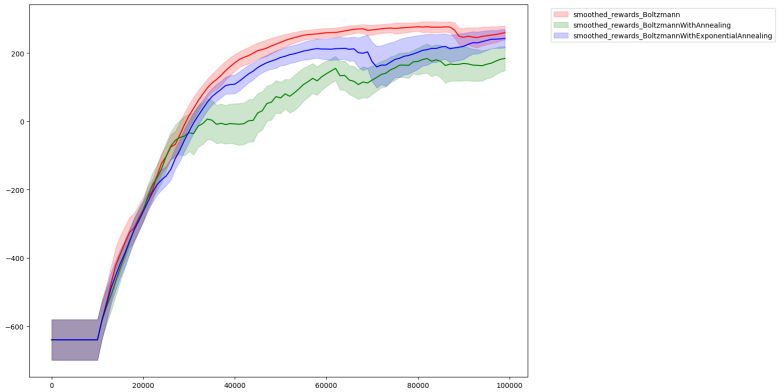
- Two Halves: $\epsilon_{final} = 0.1$, $t = 1.0$ ($\epsilon_{start} = 1.0$, $a_{steps} = 30000$) (In plots below these values are used, do not worry about different values on the plots)
- SoftSwitch: $\epsilon_{start} = 0.5$, $\epsilon_{final} = 0.01$ ($t_{start} = 0.5$, $t_{final} = 0.1$, $a_{steps} = 30000$)
- Boltzmann with Exponential Temperature Annealing: $\tau_{max} = 2.0$, $\tau_{min} = 0.1$ ($\lambda = \frac{5}{T_{total}}$)

2 Impact of annealing

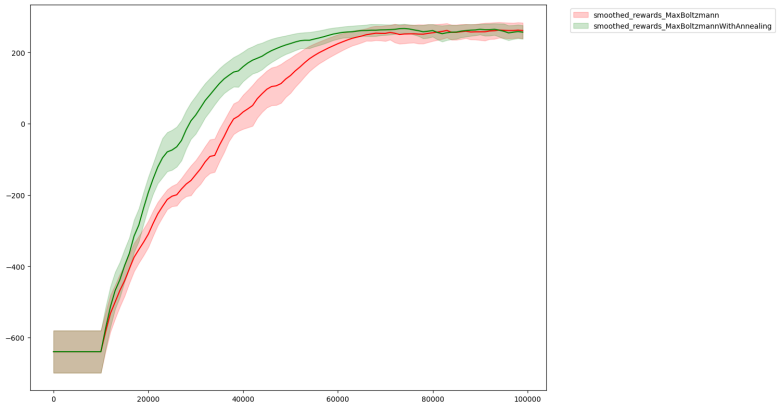
Let's see how strategies with annealing perform compared to their versions without annealing.



(a) Epsilon-Greedy with vs without Annealing



(b) Boltzmann, Boltzmann with Annealing and Boltzmann with Exponential Annealing



(c) MaxBoltzmann and MaxBoltzmann with Annealing

Figure 3: Comparision of strategies with and without annealing.

As we can see in figure 3, it's not that obvious that annealing helps.

In the case of Epsilon-Greedy, the version with annealing performs better, achieving higher average reward and converging faster.

Similarly, MaxBoltzmann with annealing slightly outperforms the version without annealing.

However, in the case of Boltzmann, we can see that Boltzmann with Exponential Annealing performs better than Boltzmann with linear annealing, but both versions with annealing perform worse than the version without annealing.

In conclusion, annealing can be beneficial for some strategies, but its effectiveness depends on the specific implementation and the strategy itself.

3 Boltzmann vs MaxBoltzmann

We can also compare Boltzmann and MaxBoltzmann strategies to see which one performs better. To do this, it's better to choose Boltzmann and MaxBoltzmann with annealing, as they performed better in the previous section.

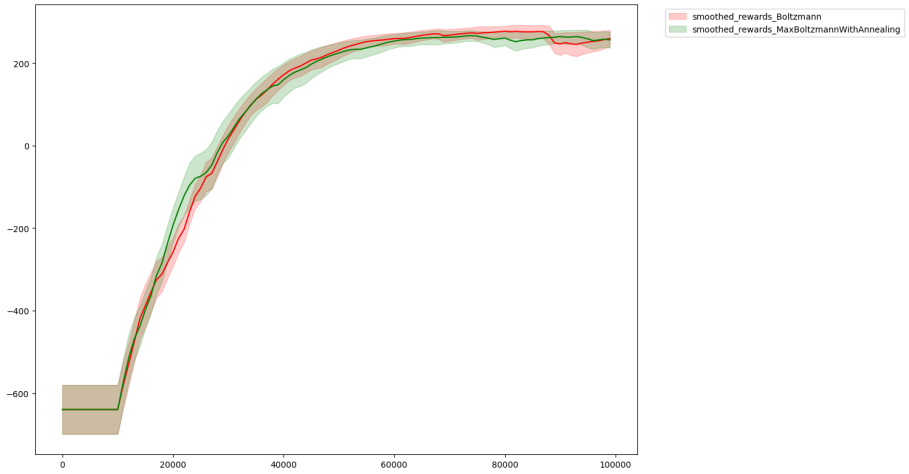


Figure 4: Comparison of Boltzmann and MaxBoltzmann strategies.

It seems, that there's no significant difference in the results of both strategies.

4 How to switch?

In this section, I compare two halves strategy, which switches from Epsilon-Greedy with Annealing to Boltzmann in the middle of training, with the Soft Switch strategy 5.1, which allows for a smoother transition between the two strategies.

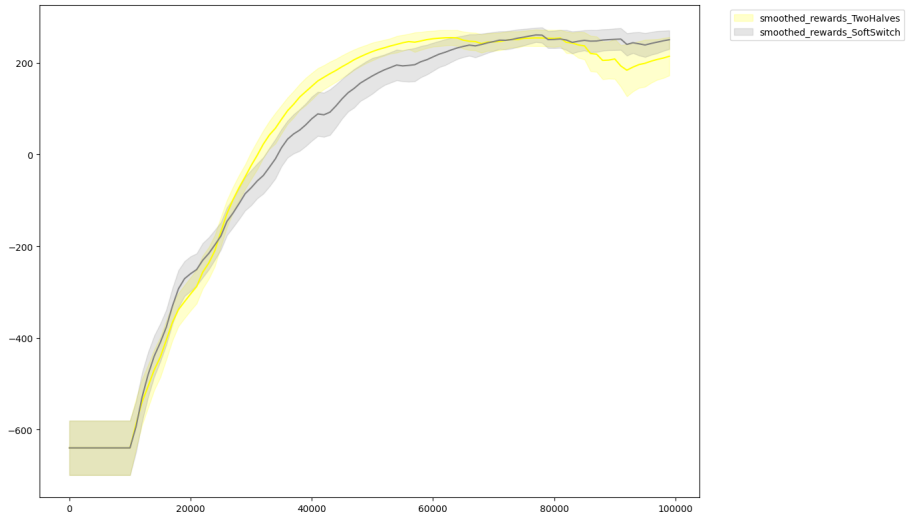


Figure 5: Comparison of Two Halves and Soft Switch strategies.

As we can see in figure 5, the Soft Switch strategy performs worse in the first half of training, but then it is more stable and achieves slightly better results in the second half of training, because two halves strategy has a small drop at the end of training. However, the difference between the two strategies is not significant, and both perform similarly overall.

5 Bonus strategies

In this section, I shortly present the bonus strategies implemented in the assignment.

5.1 Soft Switch

When using the strategy with a switch in the middle of training between Epsilon-Greedy with Annealing and Boltzmann, we can notice that maybe it would be better to have a smooth transition between the two strategies. For this purpose, I implemented the Soft Switch strategy, which uses a probability p to choose between the two strategies at each step. The probability p is calculated as follows:

$$p(t) = \begin{cases} 0 & \text{if } t < t_{start} \\ \frac{t - t_{start}}{t_{end} - t_{start}} & \text{if } t_{start} \leq t \leq t_{end} \\ 1 & \text{if } t > t_{end} \end{cases} \tag{1}$$

where t is the current training step, t_{start} is the step at which the transition starts, and t_{end} is the step at which the transition ends. At each step, we generate a random number r between 0 and 1. If $r \leq p(t)$, we use Boltzmann exploration; otherwise, we use Epsilon-Greedy with Annealing.

I've used in the notebook $t_{start} = 0.4 * T_{total}$ and $t_{end} = 0.6 * T_{total}$, where T_{total} is the total number of training steps.

The implementation is attached in the main notebook.

5.2 Boltzmann with Exponential Temperature Annealing

As you can see in figure 1, the Boltzmann strategy with linear temperature annealing acts pretty weirdly. The curve shows fluctuations. In the figure 3, we can see that this version performs worse than the version without annealing.

Therefore, I decided to implement a variant with exponential temperature annealing. The temperature τ is calculated as follows:

$$\tau(t) = \tau_{min} + (\tau_{max} - \tau_{min}) \cdot e^{-\lambda t} \tag{2}$$

Where the decay constant λ is defined as:

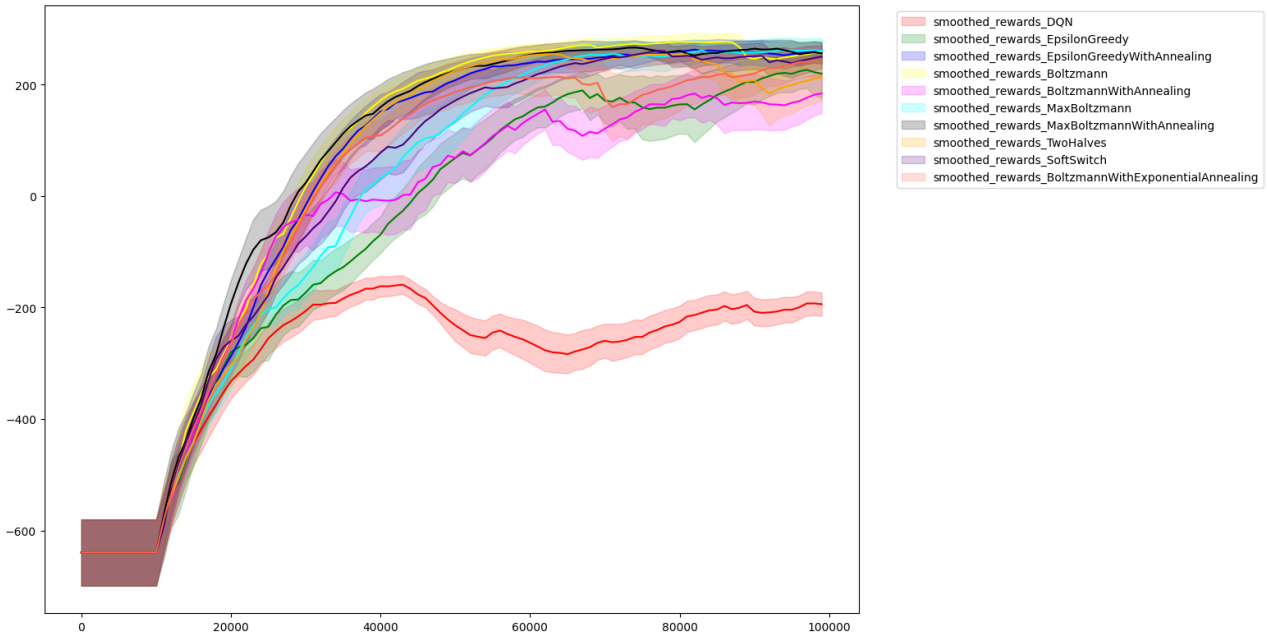
$$\lambda = \frac{5}{T_{total}} \tag{3}$$

Where T_{total} is the total number of training steps. This way, the temperature decreases rapidly at the beginning and then slows down, allowing for more exploration early and more exploitation later.

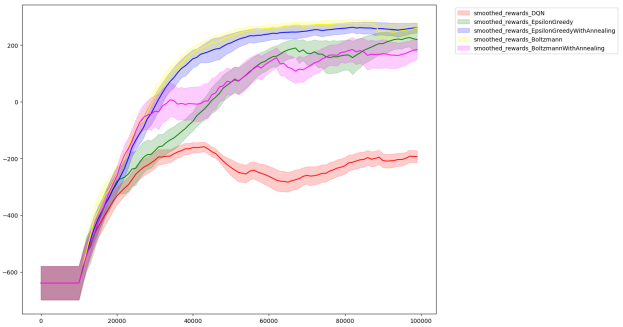
The implementations is presented in the main notebook.

6 Final comparison

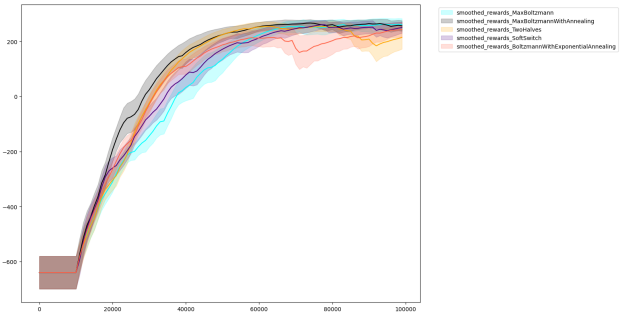
Using tuned hyperparameters from section 1, I performed a final comparison of all strategies. The results are presented in figure 6.



(a) All tuned strategies



(b) First half of all tuned strategies



(c) Second half of all tuned strategies

Figure 6: Final comparison of DQN exploration strategies in LunarLander-v3 with tuned hyperparameters.

Now, we can see that it is hard to choose the best strategy, as most of them perform similarly. However, we can make some observations.

Strategies **Boltzmann**, **Epsilon-Greedy with Annealing**, **MaxBoltzmann**, **MaxBoltzmann with Annealing**, **Two Halves** and **Soft Switch** perform better than **Boltzmann with Exponential Temperature Annealing**, **Epsilon-Greedy**, **Boltzmann with Annealing** and **random DQN**, achieving higher average rewards.

The worst strategy is obviously the **random DQN**, which performs poorly compared to other strategies. What is more, it cannot even reach the average reward of -100, while other strategies reach

at least 150 average reward.

7 Conclusions

In this report, I investigated various exploration strategies for the DQN algorithm in the LunarLander-v3 environment. I tuned hyperparameters for each strategy and compared their performance.

We can see that mostly, parameters that encourage more exploitation (lower ϵ for Epsilon-Greedy, lower temperature for Boltzmann) perform better. It shows that in the environment of LunarLander-v3, it's more beneficial to exploit known good actions rather than explore new ones.

Annealing can be beneficial for some strategies, such as Epsilon-Greedy and MaxBoltzmann, but its effectiveness depends on the specific implementation and the strategy itself. In the case of Boltzmann, both linear and exponential annealing performed worse than the version without annealing, but exponential was better than linear.

Comparing Boltzmann and MaxBoltzmann strategies, I found no significant difference in their performance.

Furthermore, I examined the impact of the way of switching between two strategies. The results indicate that both the abrupt switch (Two Halves) and the smooth transition (Soft Switch) perform similarly, suggesting that the method of switching may not significantly affect overall performance in this case.