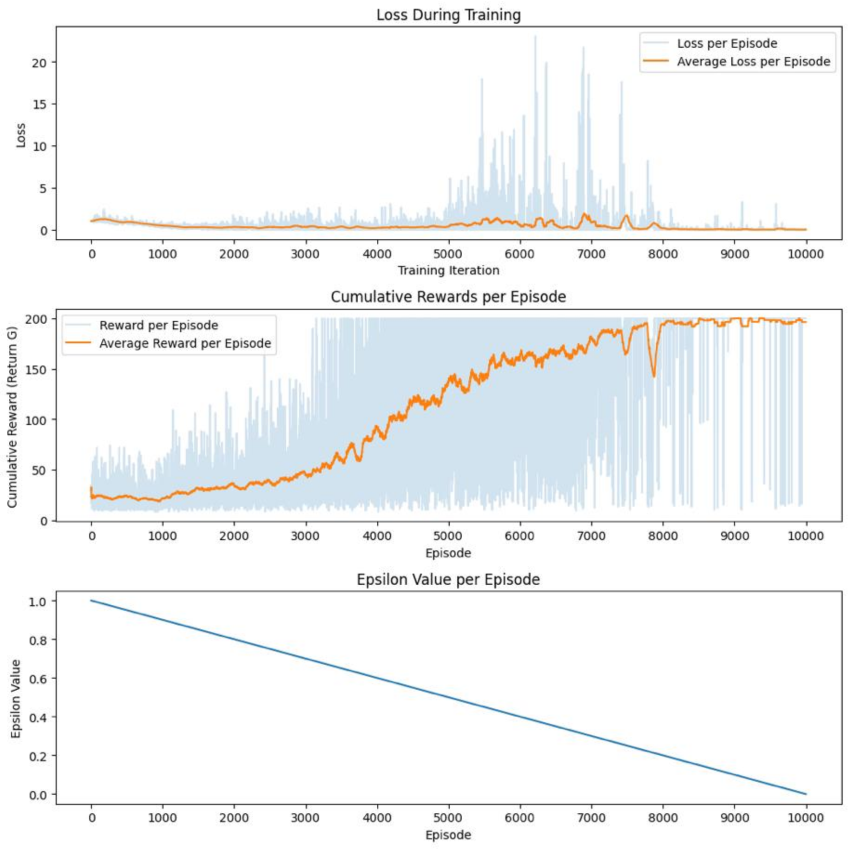
Deep Reinforcement Learning

Sheet 05

Valentin Adam, Moritz Grunert, Sven Ullmann

Exercise 1

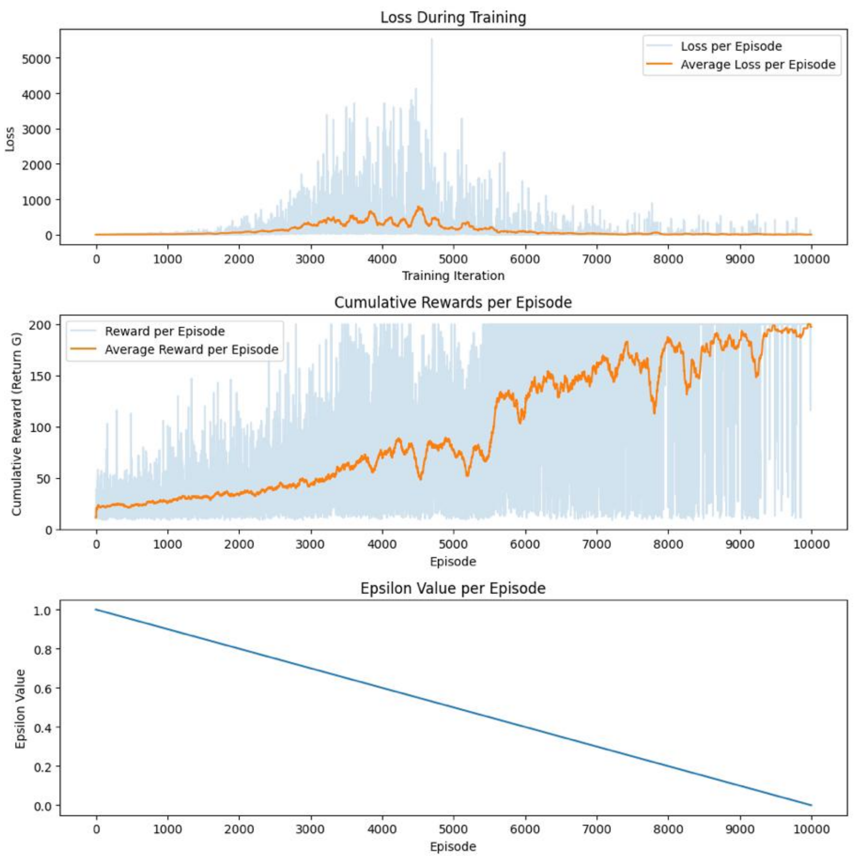
a) Visualization:

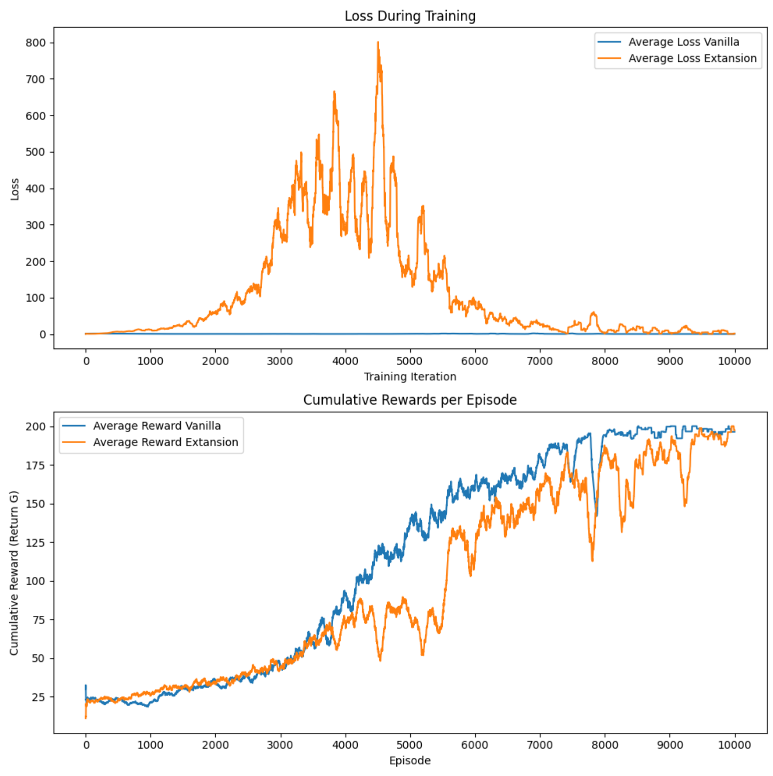


a) Reflection

Function approximation and large state spaces make visualization of the resulting value functions unintuitive. One effective idea to visualize the value function is to use heatmaps. A heatmap can represent the Q-values for different actions in a subset of the state space (e.g., fixing some state dimensions while varying others). Alternatively, in environments like CartPole, a 2D projection can be generated by fixing specific parameters (e.g., the pole angle and angular velocity) and visualizing the Q-values for the remaining state variables.

b) Visualization:





b) Reflection

Replay buffer: Replay buffers store the transitions and use them for training. This is because N random transitions are selected, which are then used for training. This allows correlations between successive transitions to be avoided, making the algorithm more stable, as overfitting can be avoided.

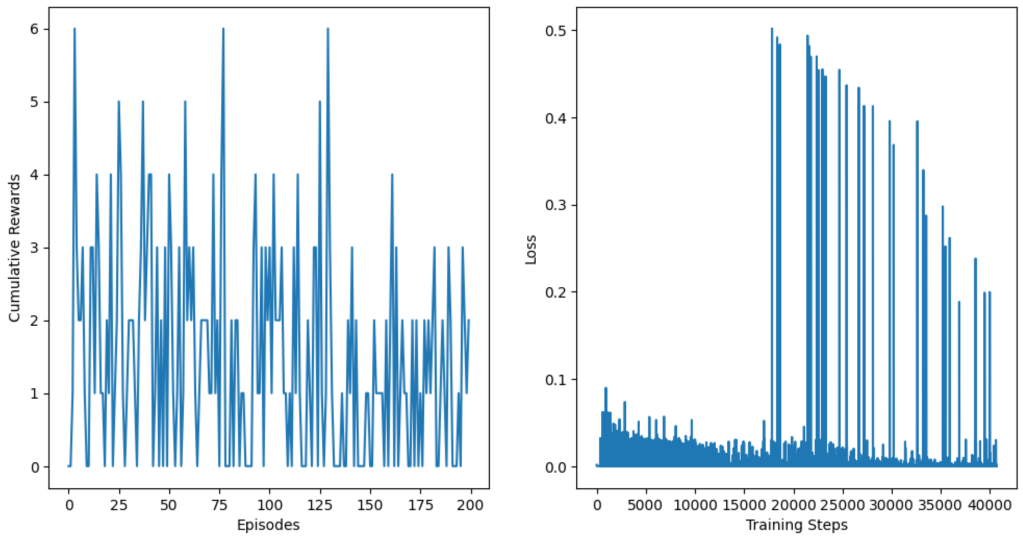
Target network: The target network copies the Q network every C step. This results in a more stable Q-value estimation, as the target network is updated less frequently and therefore does not fluctuate as much, and the gradient calculation can also be more stable as a result.

The algorithm with a buffer and with a target network should actually be better, unfortunately this is not the case in our example. There could be several reasons for this:

1. the most plausible reason would be that something is wrong in the algorithm, i.e. that we have an implementation error.
2. the other reason could be that the buffer size and C, i.e. after many episodes the Q network should be copied, are set incorrectly and this leads to poorer model performance.

Exercise 2

Visualization:



This is obviously not the reward/loss plots we were hoping for but we did not find the bug in our code....

Reflection: Using CNNs for feature engineering in the Breakout environment transforms the high-dimensional pixel data into a lower-dimensional representation.

1. Instead of using input data of size $210 \times 160 \times 3$, we perform preprocessing of the images to obtain inputs of size $84 \times 84 \times 1$. The 1 dimension indicates that we use greyscale instead of RGB. As our experiments / visualization shows, for a game like breakout, the greyscale pictures are enough for the agent to learn the policy.

2. Extracting Relevant Features: The convolutional layers in the CNN automatically learns features which are critical for understanding game elements like the paddle, ball, and brick instead of relying on raw pixel data, which has a lot of irrelevant information. Using preprocessed images reduces the curse of dimensionality and enables the agent to operate effectively in complex environments like Breakout.