|  |  |  |
| --- | --- | --- |
| Epsilon | Time | Episodes |
| 0 | 57.38670372962952 | 820 |
| 0.1 | 52.873379945755005 | 745 |
| 0.2 | 58.15610671043396 | 736 |
| 0.25 | 57.99597883224487 | 655 |
| 0.3 | Not solved | Not solved |
| 0.5 | Not solved | Not solved |

You might find it hard to get deep Q-Learning working properly and find that training it is quite unstable. This comes from the fact that now a single update for a specific action-value-pair (St,At) can in influence the approximate q values for many other state-vale-pairs, since all of the networks weights may be affected by an update. As subsequent states and actions are highly correlated to each other, the network is under risk to “forget” about values in the regions of the state-action space that were visited earlier.

However, training has become unstable, due to the fact that now one update for a particular action-value pair can affect the approximate values of q for many other state-value pairs, since all the weights of the networks can depend on update. We use Experience Replay to solve this problem. During Experience replay, the algorithm remembers all the state transitions that took place during the operation and trains on them all. In this case we can still perform an update in every time step, but we could also decide to update the network's weights only every few time steps. For our task, we update every 100 steps. Table \ref{table:4} shows the results for Deep Q-Learning.\\