DESCRIPTION OF THE ALGORITHM

The algorithm used to solve the environment is a DDQN that uses a Dueling architecture on top of it.
Let's start to explain DQN on which DDQN is based:

The DQN algorithm generalizes the SARSA class of algorithms. Specifically, SARSA max. Instead of using the Q table we use a neural network to approximate the Q table. However, contrary to SARSA max, when we do bootstrap to improve the quality of our Q table (net), we update also all the other Q table values. This is due to the fact that the net weights somehow encodes all the possible values of the Q table. To prevent this destructive error we use a target net which is modified every N steps.

The other parts of the algorithm work in the following way: We play a game with the epsilon greedy approach. The result of our play is stored in a Replay Buffer which is batched once every K steps. This batch of experience is fed to the net in order to apply SGD and improve the quality of the net estimate.
DDQN starts from the observation that DQN forecasts are over-optimistic. To mitigate this factor we do a double check on the target action chosen for the bootstrap. We choose it with one net and re-evaluate this pick with another net. This improves the quality of our estimates.
Finally the Dueling architecture is used to decouple the estimate of the value state $V(s)$ from the estimate of $Q(s, a)$. This allows to have a clear idea of the value of the state instead of relying on all possible $Q(s, a)$.

DESCRIPTION OF THE NET

The net contains several linear layers. The first takes as input the observation space, and gives out an hidden layer with 256 nodes. The second takes these 256 values and gives back 128 values. At this point the net obeys the Dueling architecture and it is splits in two branches: The first is a linear layer with input size 128 and output size 1. The second is a linear layer with input size 128 and output size the dimension of the action space.

DESCRIPTION OF THE LEARNING PARAMETERS

$BUFFER\ SIZE = int(1e5)$
$BATCH\ SIZE = 64$
$GAMMA = 0.99$
$UPDATE\ EVERY = 4$
$LR_RATES = 5e - 4$

Most of the parameters are self explanatory if one know a bit of Deep Learning and if the previous section has been read. Let's just indulge over

$UPDATE\ EVERY$: which decides when to update Net. Which, in our case, is when we collect 4 new pieces of information) $GAMMA$: which is a discount factor used to give less importance to future rewards.

TABLES

The table of the plotting rewards is included inside the Jupyter Notebook. The number of episodes to solve the environment varies between 500 episodes and 700 episodes.

FUTURE IDEAS

Other possibilities to solve this environment include Distributional DQN and Prioritized Experience Replay. One can also combine all these techniques as in the DeepMind paper Rainbow.
Also, one can tweak another bit the hyper-parameters since I doubt they are optimal.