TicTacToe

Maximilian Gangloff, Trung-Dung Hoang
CS-433 Machine Learning, EPFL, Switzerland

I. Q-LEARNING

A. Learning from experts

Question 1: The agent did successfully learn to play TicTacToe using a low exploration rate when training against an optimal player with an exploration rate of 0.5. The rewards were the highest with an exploration rate of $\epsilon = 0$.

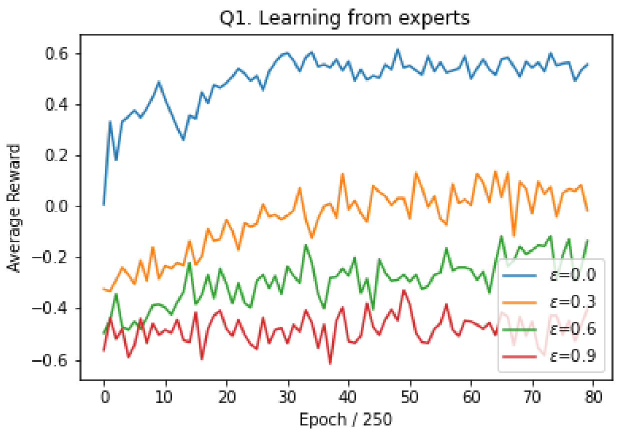

Fig. 1. Average rewards with different values of the exploration rate ϵ.

1) Decreasing exploration:

Question 2: Using a decreasing exploration with a minimum of 0.1 does not help the agent to learn to play against an optimal player with $\epsilon_{opt} = 0.5$. This is because the agent has the highest rewards when $\epsilon = 0$ which is never reached with our decreasing exploration function.
When looking at the rewards of $n^* = 10'000$, we can clearly see, how the decreasing exploration function affects the agent. At the start, the exploration rate is high meaning that the agent has a higher probability to take a random action which helps him to discover new states. The exploration rate then decreases over time which makes the rewards go up. At around 10'000 games, the exploration function then reaches its minimum of 0.1. Comparing now the rewards of $n^* = 10'000$ to the rewards of $n^* = 1$ and $n^* = 100$, we can see that the rewards are slightly higher after that half of the games are played. However, when n^* is too high e.g. $n^* = 40'000$, the exploration rate never reaches its minimum and the performance is worse. (after 20'000 games, $\epsilon = 0.4$)

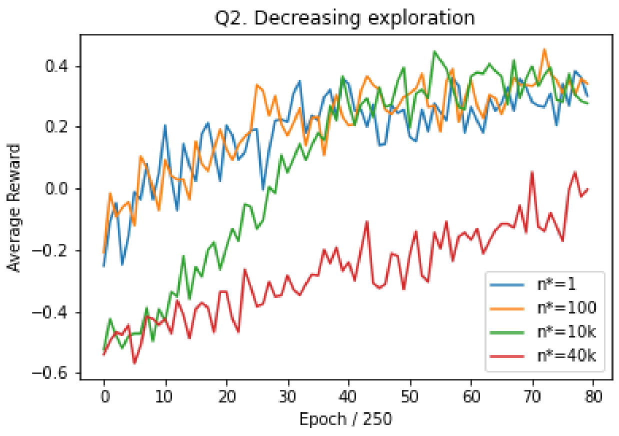

Fig. 2. Average rewards with different values of the exploratory games, n*.

Question 3: When testing the agents against M_{opt} and M_{rand}, we observe, that if n^* is not too high, i.e. the exploration rate reaches its minimum before the 20'000 games, then the agents perform comparably well against the random player. The agent with $n^* = 40'000$ perform a bit worse against the random player compared to the other agents with lower n^*. When looking at the performance against the optimal player, the agents with low n^* performs better than the player with $n^* = 40'000$. However, they perform a bit worse compared to the agent with $n^* = 10'000$ which was able to discover more states.

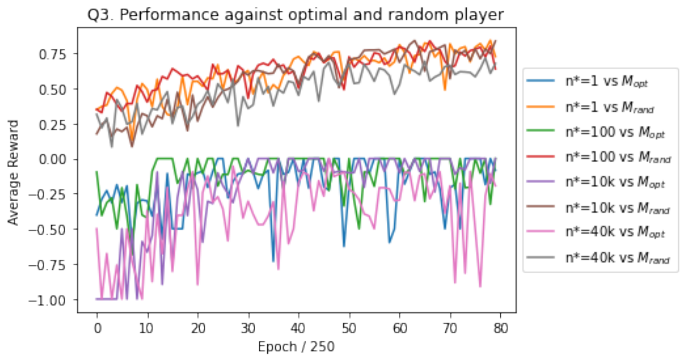

Fig. 3. Average rewards with different n* against an M_{opt} and M_{rand} player

2) Good experts and bad experts:

Question 4: We now train against optimal players with different exploration rates $\epsilon_{opt} \in [0, 1)$. When looking at the 'test' performance against the random player, we observe that the higher ϵ_{opt} gets, the higher the rewards are. This is because,

as ϵ_{opt} increases, the optimal player becomes more and more a random player and makes mistakes. These mistakes can result in a positive reward for the agent which helps him win games against a random player. However, the higher ϵ_{opt} gets, the worse the performance against the optimal player gets because when the agent makes a mistake against a player with high ϵ_{opt}, then the mistake might not get punished and the agent will have a higher probability to repeat these mistakes which will result in a loss against an optimal player.
When learning against $\epsilon_{opt} = 0$, the optimal player always plays the best move which results in the optimal player always 'attacking' and the agent always defending 'defending'. In this case, the agent only learns how to not lose the game instead of how to win it. When then using a higher ϵ_{opt}, e.g. $\epsilon_{opt} = 0.5$, the agent learns how to draw games against the optimal player but also how to win most of the games against a random player.

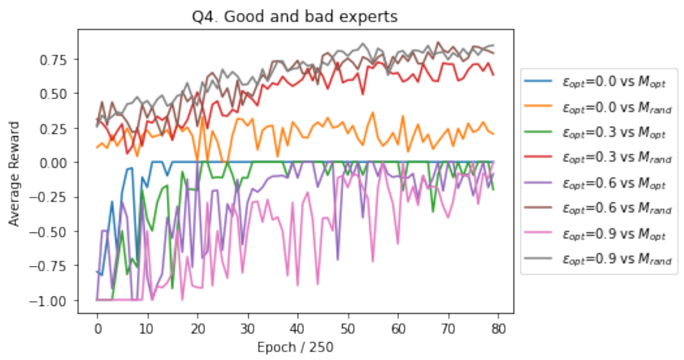

Fig. 4. Learning with $n^* = 10'000$ against different ϵ_{opt}

Question 5: After 20'000 games, the highest M_{opt} is equal to 0. This is normal and to be awaited since if correctly played, a player never losses. Thus the optimal player will never lose. This means that the rewards for the agent will always be less than or equal to 0. If now the agent has played and learned enough, then in the case of Tic Tac Toe, he should never lose. This will result in all games ending in a draw.
The highest M_{rand} is equal to 0.874. This value is pretty high but is not out of the ordinary since if correctly played, a player never losses. If now the agent has played and learned enough, then all games will end in either a win for the agent or a draw.

Question 6: If agent 1 finds an optimal Q1(s,a) by playing against Opt(0) and Agent 2 finds an optimal Q2(s,a) by playing against Opt(1), then Q1(s,a) and Q2(s,a) do not necessarily have to have the same value. Since Agent 1 plays against an optimal player who always chooses the best action, the game always ends in either a draw or a lose for Agent 1. This means that all the Q-values of Agent 1 will be smaller or equal to 0 since the only possibility to obtain a positive reward is by winning a game. On the other hand, Agent 2 will win most of its games and never lose once he has learned the optimal Q-values. Thus he will have Q-values that are greater or equal than 0.

B. Learning by self-practice

Question 7: Using now self-practice with different values of ϵ, we test the agents after every 250 games against a random and optimal player. We see that for $\epsilon = 0$ the agents are not able to learn to play Tic Tac Toe because not enough new states are discovered. The optimal ϵ seems to be between 0.2 and 0.3. It is high enough to explore enough new states but also low enough to most of the time draw against the optimal player. With higher exploration rates the agents also perform well against the random player but perform poorly against the optimal player since the moves are more random.

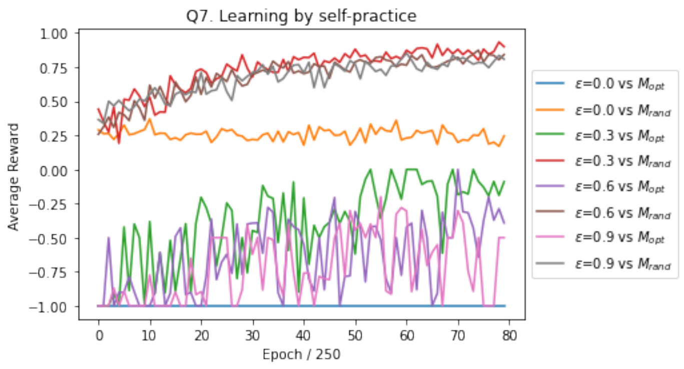

Fig. 5. 'test' M_{opt} and M_{rand} for different values of $\epsilon \in [0, 1)$.

Question 8: A low number of n* corresponds to a fixed ϵ so there is not much difference between the fixed and the decreasing exploration rate. However, for higher values of n*, we see an improvement against the optimal player at the end of the games. The optimal n* seems to be 10'000. The agents have long enough time where the exploration rate is higher to discover new states but reaches its minimum before the end of the 20'000 games. When n* is too high, however, the exploration rate is still too high at the end (e.g. $n^* = 40'000$ corresponds to an exploration rate of 0.4 after 20'000 games).

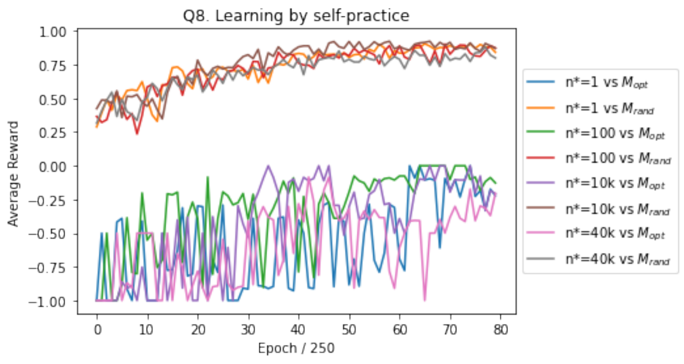

Fig. 6. 'test' M_{opt} and M_{rand} for different values of n*.

Question 9: The highest rewards obtained after playing 20'000 games were 0.95 against the random player and 0 against the optimal player. The value against the optimal player is the same as in 5. However, when looking at the value against the random player, they are quite higher when learning by self-practice. This could be because when learning by self-practice, the two agents use the same Q-table and so after 20'000 games, more states are discovered. This mainly affects the results against the random player since when testing against the optimal player, the same states are repeated much more often.

Question 10: The agent learned well to play against a random player but still struggles to constantly draw against an optimal player.
When looking at the three states in figure 7, we observe in the first state, that there is a winning opportunity for the first player 'X' which corresponds to the highest Q-value.
The second heat map shows the block of a fork where the player 'O' has only two options to play, either the bottom left or the top right corner, to avoid the fork. The Q-value of the top right corner is 0.48 while the one of the bottom left corner is only -0.65 even though both plays result in a draw if played perfectly.
However, if player 'O' then plays somewhere else e.g. in position (1, 0) which is the second-highest Q-value in the second heat map, then the player 'X' has a fork opportunity which results in a forced win shown by the highest Q-value in the 3rd heat map.
In all of these cases, the highest Q-value corresponds to one of the optimal plays to make by the agent. Thus, we can conclude that the agent did learn to play Tic Tac Toe however is not yet an optimal player yet.

II. Deep Q-learning

A. Learning from experts

Question 11: The agent did successfully learn to play TicTacToe using a low exploration rate when training against the 0.5-greedy optimal player. In the case $\epsilon = 0.0$, the loss first increases then decreases and the reward increases over time, which confirms the conclusion. Figure 9.

Question 12: We repeat the training but without the replay buffer and with a batch size of 1. The trends of the loss and reward are quite similar to these in the previous part Figure 10. However, they observe more fluctuations, bigger losses and smaller rewards.

Question 13: Using a decreasing exploration with a minimum of 0.1 does not help the agent to learn to play against the optimal player with ϵ_{opt} = 0.5. While the values of M_{rand} are approximately equal, the value of M_{opt} when ϵ is fixed is way more stable and higher than these of decreasing ϵ.The agent has the best performance when $\epsilon = 0$ which is never reached with our decreasing exploration function.

Higher n^* helps the agent spends more steps to discover new states. When $n^* = 10000$, after exploring a certain number of states during about 700 games and the ϵ approaches the minimum value ($\epsilon_{min} = 0.1$), both M_{rand} and M_{opt} become higher than these figure of $n^* = 1$ and $n^* = 100$ Figure 11. However, if n^* is too large, the fact that many discovered states are not useful and the agent does not exploit well makes the performance worse.

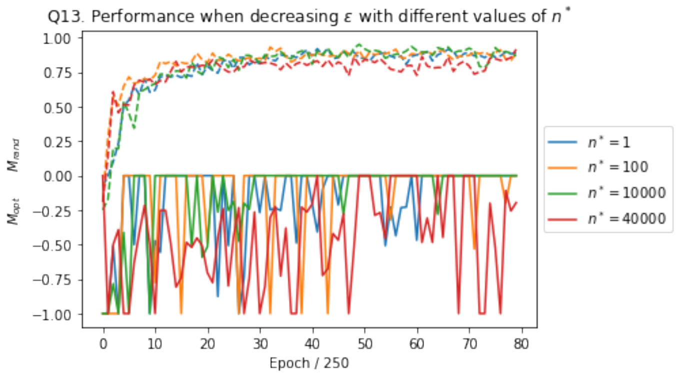

Fig. 11. M_{rand} and M_{opt} for different values of n^* compared to the case that ϵ is fixed to 0.0. Solid lines and dashed lines stand for M_{rand} and M_{opt}, respectively.

Question 14: We observe that when ϵ_{opt} gets higher, the M_{rand} increases faster at the beginning, while the M_{opt} becomes smaller and more fluctuated Figure 12. The reason is that the expert behaves approximately similar to a random player. On the one hand, it helps the agent win the random player. On the other hand, the agent does not receive good feedback from the environment, which leads to poor performance against the actual optimal player.

When $\epsilon_{opt} = 0$, the optimal player always plays the truly best action. The agent is trained and then tested against exactly the same optimal player, which helps M_{opt} reach 0 quickly. However, when playing against a random player, it observes multiple states that it has never seen before and has bad performance. A value between the two above extreme cases performs best. It helps the agent learns how to draw games against the optimal player but also how to win most of the games against a random player.

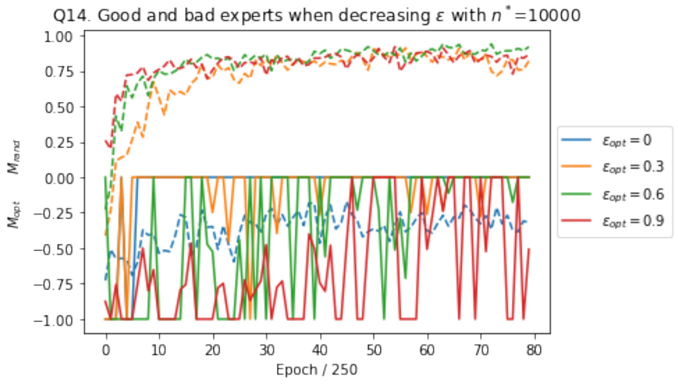

Fig. 12. M_{rand} and M_{opt} with different values of ϵ_{opt}

Question 15: The highest values of M_{opt} and M_{rand} achieved after playing 20000 games are 0.000 and 0.952, obtained with $\epsilon_{opt} = 0.5, n^* = 10000$.

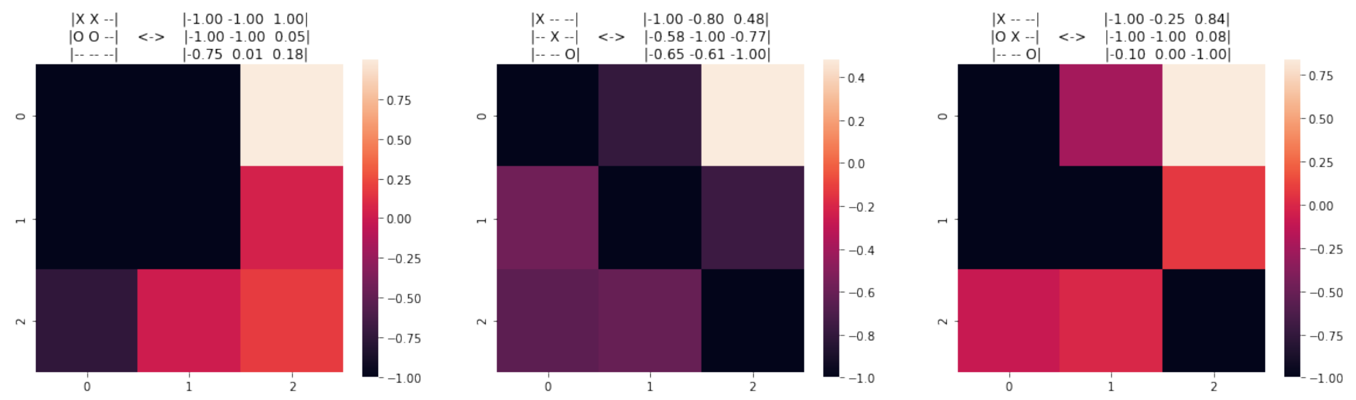

Fig. 7. Q-Learning heat maps of three different states showing their Q-values. A value of -1 means that the cell has already been taken and is invalid.

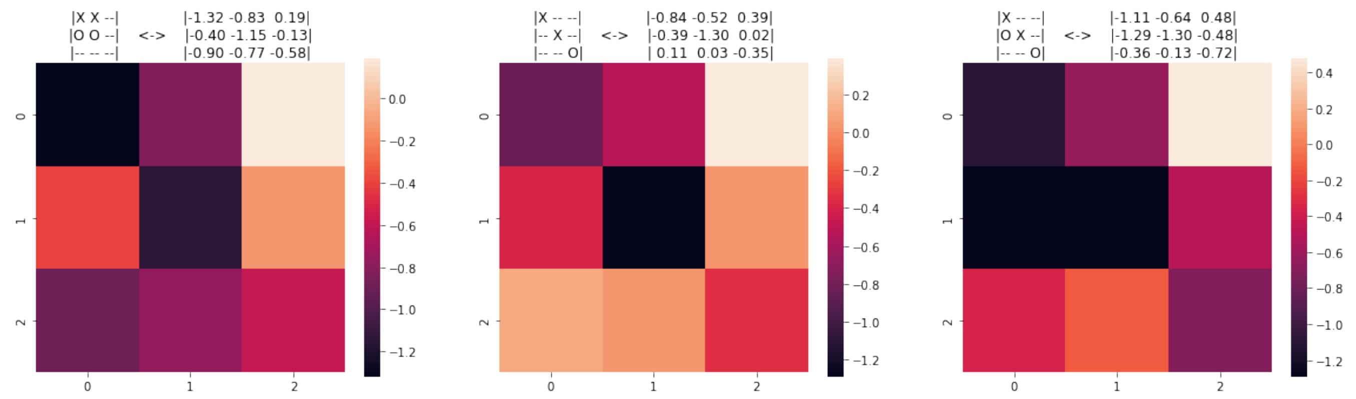

Fig. 8. DQN heat maps of three different states showing their Q-values.

B. Learning by self-practice

Question 16: The agent can learn to play Tic Tac Toe depending on the ϵ Figure 13. If ϵ is too small, the agent hardly discovers new states. On the other hand, if ϵ is large, self-practice behaves like two nearly random players play against each other and the agent does not receive proper feedback to learn. That leads to worse performance compared to the smaller ϵ cases. While the agent is still able to learn if we set ϵ to 0.6 or 0.9, the better performance is achieved with lower one, e.g. $\epsilon = 0.1, 0.3$.

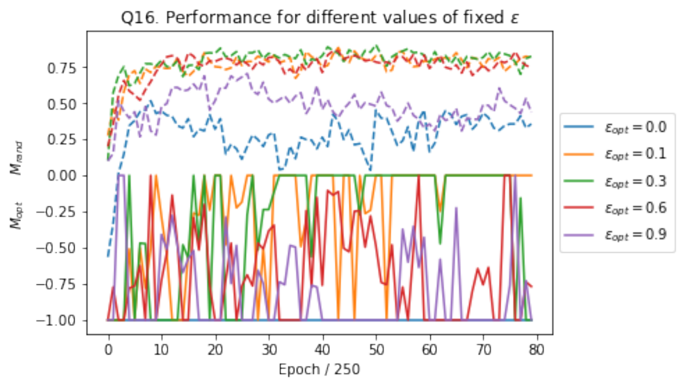

Fig. 13. Self-practice M_{rand} and M_{opt} for different values of ϵ_{opt}.

Question 17: Decreasing ϵ can help training compared to having a fixed ϵ, depending on the value n^* we set Figure 14. If n^* is small, the performances of decreasing ϵ and fixed ϵ are approximately the same since ϵ approaches the lower bound very fast in this case. In addition, when we set n^* is large, like 40000, the agent spends a major number of steps for exploration without much exploitation. The optimal choices of n^* are some values in the middle, i.e. 10000.

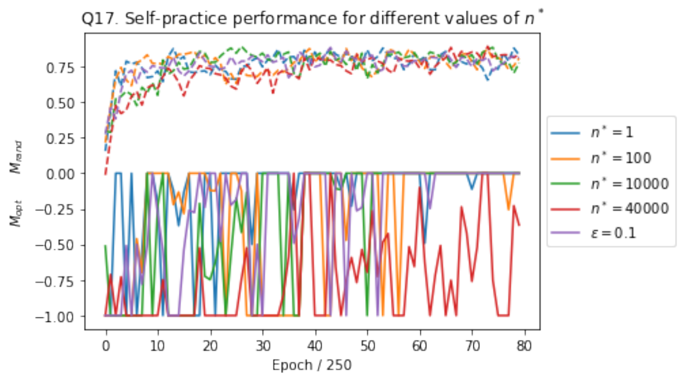

Fig. 14. M_{rand} and M_{opt}.

Question 18: The highest values of M_{opt} and M_{rand} achieved after playing 20000 games are 0.0 and 0.904, obtained with $n^* = 1000$.

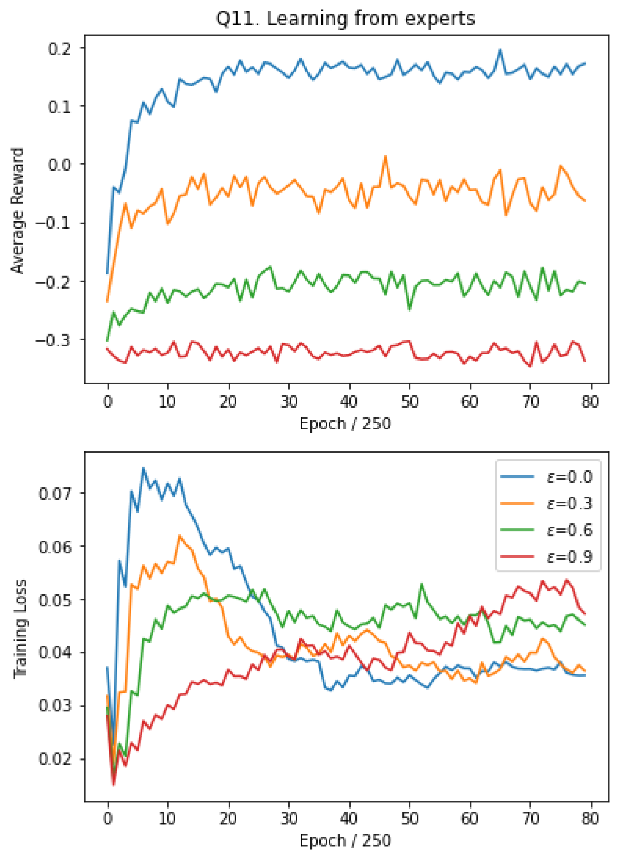

Fig. 9. Average rewards and training losses for different values of the exploration rate ϵ.

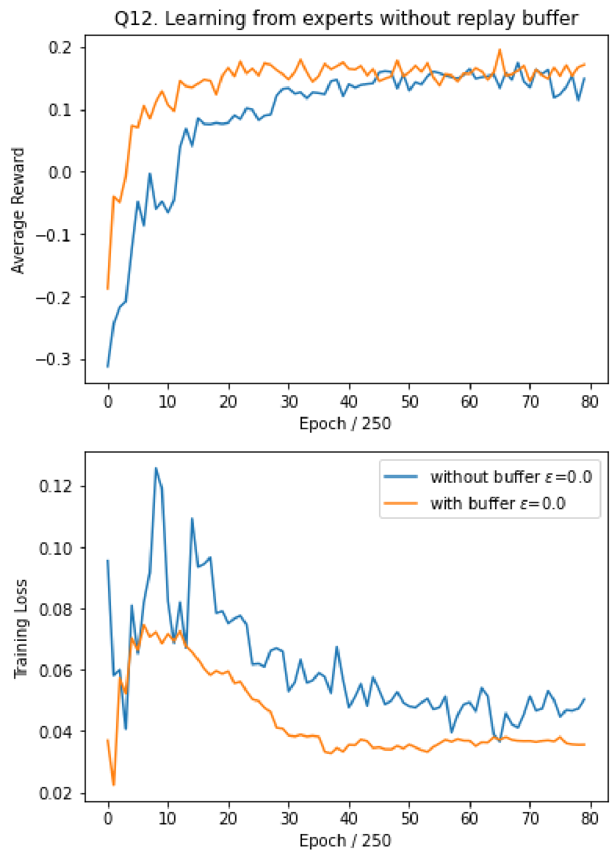

Fig. 10. Average rewards and training losses for $\epsilon = 0.0$ with and without replay buffer.

Method	Learn from experts		Self-practice	
	(M_{opt}, M_{rand})	Time	(M_{opt}, M_{rand})	Time
Q-Learning	(0.0, 0.87)	11'000	(0.0, 0.95)	13'000
DQN	(0.0, 0.952)	5500	(0.0, 0.904)	1500

TABLE I
COMPARING Q-LEARNING WITH DEEP Q-LEARNING

Question 19: We have the same setting as the previous part in Q-Learning. The agent can learn to win, block an opponent's fork and fork Figure 8. Thus, we can conclude that the agent can learn to play Tic Tac Toe however is not an optimal player yet.

III. Q-LEARNING VS. DEEP Q-LEARNING

Question 20: The best performance (the highest M_{opt} and M_{rand}) of Q-Learning and DQN (both for learning from experts and for learning by self-practice) and their corresponding training time Table I.

Question 21: When looking at the Table I above, the biggest difference we see comes from the convergence time. We can see, that Deep Q-learning converges much faster than classical Q-learning. This probably comes from the fact, that when a state has never been seen before in Q-learning, then all the Q-values are 0 and so we choose randomly an action and the information takes more time to propagate. This could be enhanced by implementing eligibility traces. However in Deep Q-learning, when encountering unseen states, the neural network is still able to predict suitable actions for them based on the information from observed states. The actions might be not the optimal ones, but still more reasonable than a random action. Moreover, all Q-values are predicted by the same network, so once a state is used to update the network, the information can be shared directly with other states.

To conclude, we can say, that both methods worked to correctly learn to play Tic Tac Toe. Tic Tac Toe is an easy enough game where the possible states are pretty limited so the Q-learning can also learn quite well the game. However, when training for 20'000 games, there can be the case that the Q-learning algorithm discovers a state that has never been seen before and thus choose a random action whereas the deep Q-learning algorithm will perform pretty well despite never having seen the state.