Reinforcement Learning -
Assignment -Snakes and Ladders

ID: m10902121 name: Yu-Cheng, Li 李育誠

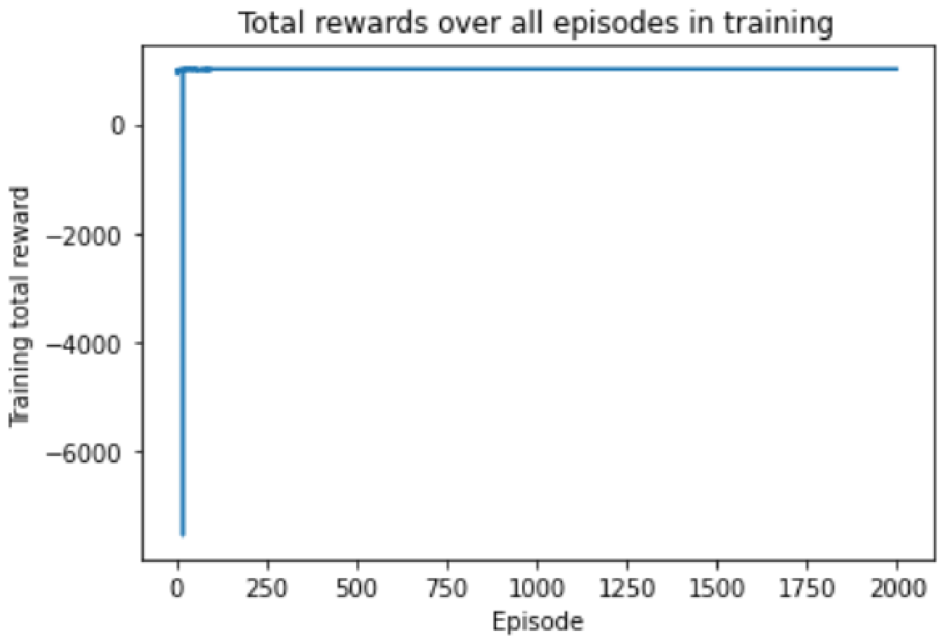
Assignment Requirements:

This environment is based on a variation of the snakes and ladders game. The player rolls a fair dice and then moves this many spots forward. Create a game with 200 spots by combining two 100 spot boards and one with 300 spots by combining three boards with 100 spots.

Programming Function:

I have implemented version 0, and version 1 of this assignment. However, I have implemented Q-Learning and TD(lambda) and it’s variant of the descending epsilon-greedy algorithm.

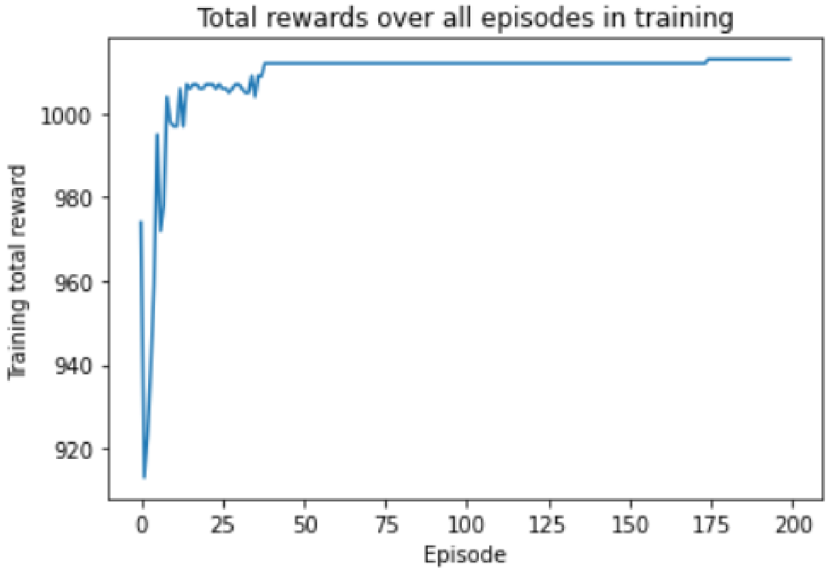
Results and Discussion:
1.1 TD (0.1) numbers of spots:100
Hyper parameters:
alpha = 0.01
gamma = 0.8
train_episodes = 2000

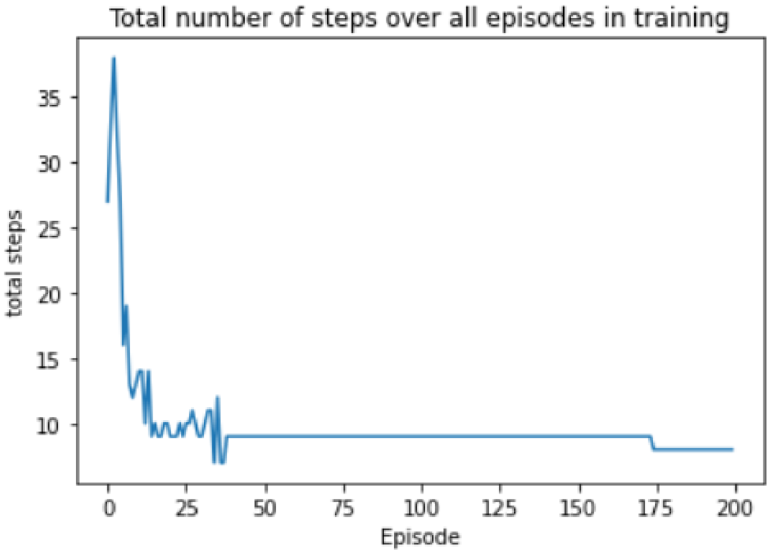


I observe this graph and figure out it is not diverse at the end. Even at the beginning it is just merely diverse. Because of this reason I tried to usd descending epsilon-greedy algorithm to the implementation of this TD-algorithm.

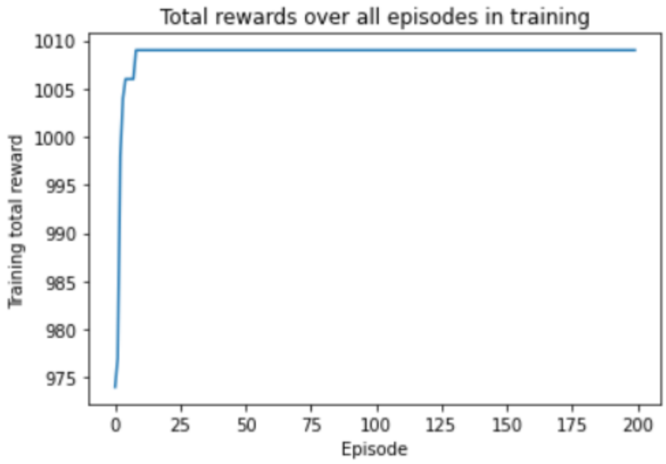
1.2 TD (.) numbers of spots:100
spot_num = 100
alpha = 0.001
gamma = 0.85
epsilon = 0.5
max_epsilon = 1
min_epsilon = 0.01
decay = 0.1
train_episodes = 200

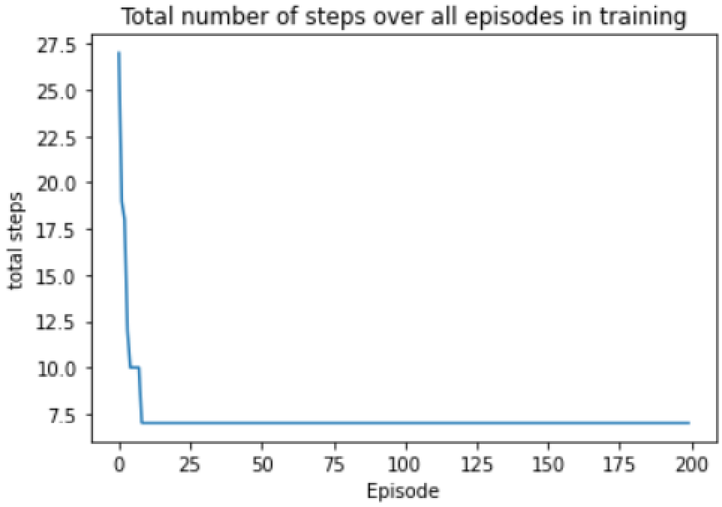
1.2.1 TD (0.1) numbers of spots:100 - variant



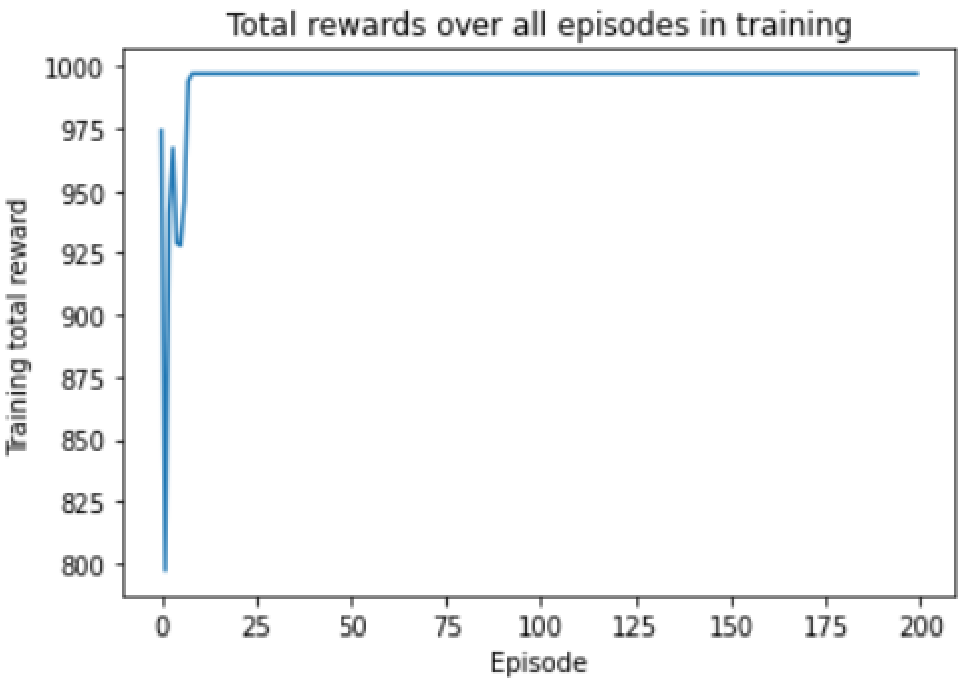


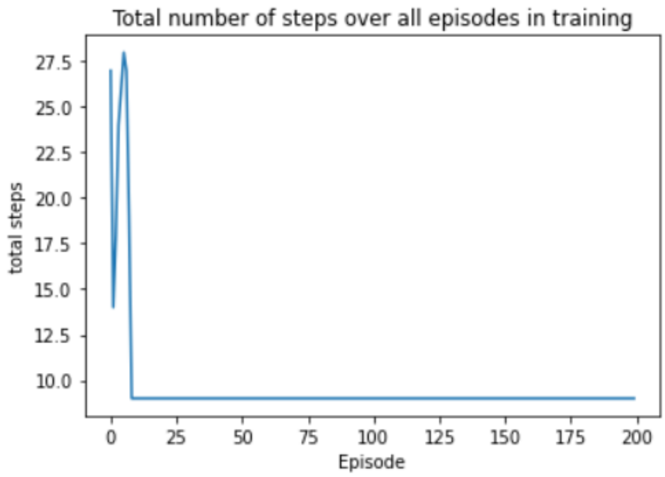
1.2.2 TD (0.5) numbers of spots:100 - variant





1.2.3 TD (0.9) numbers of spots:100 - variant

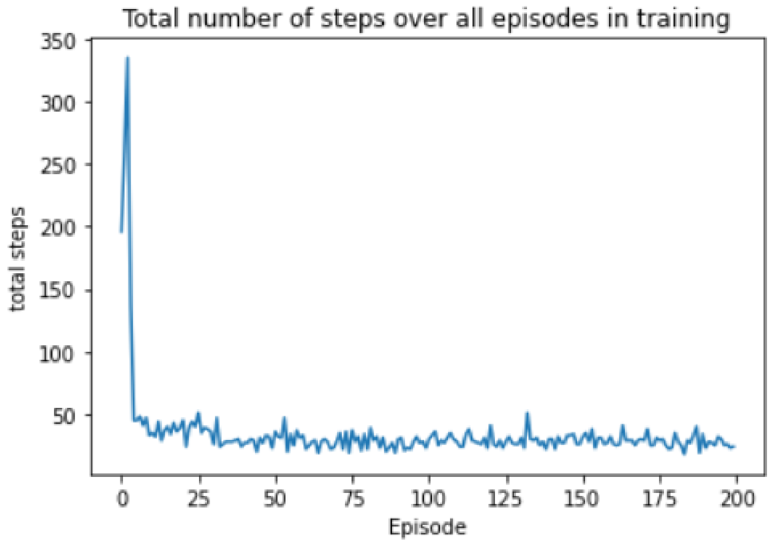




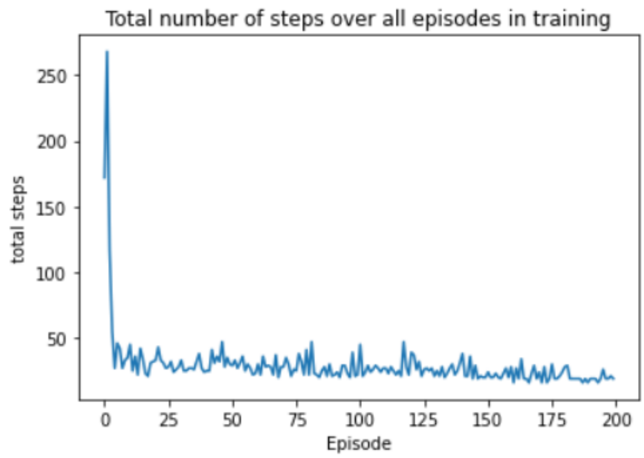
1.3 TD (.) numbers of spots:200

spot_num = 200
alpha = 0.001
gamma = 0.85
epsilon = 0.5
max_epsilon = 1
min_epsilon = 0.01
decay = 0.1
train_episodes = 200

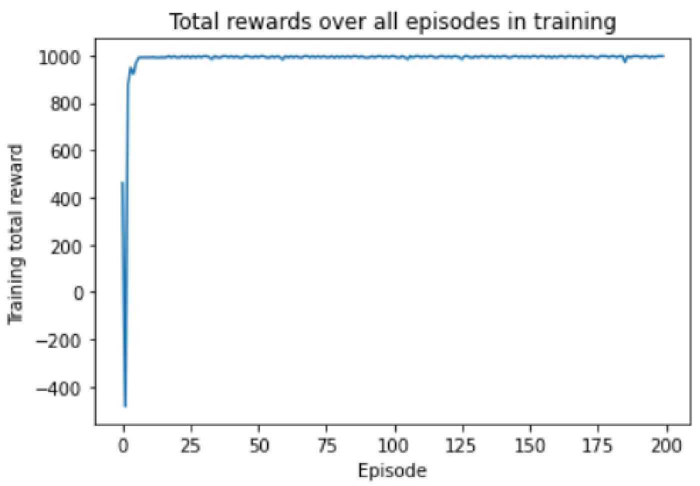
1.3.1 TD (0.1) numbers of spots:200 - variant

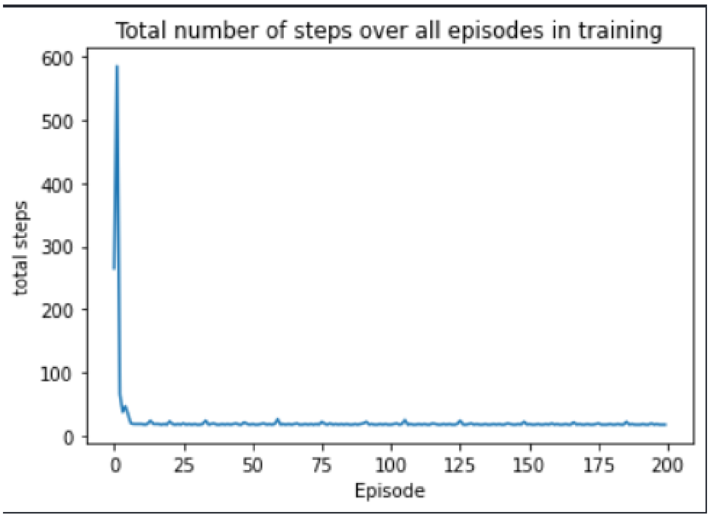


1.3.2 TD (0.5) numbers of spots:200 - variant



1.3.3 TD (0.9) numbers of spots:200 - variant

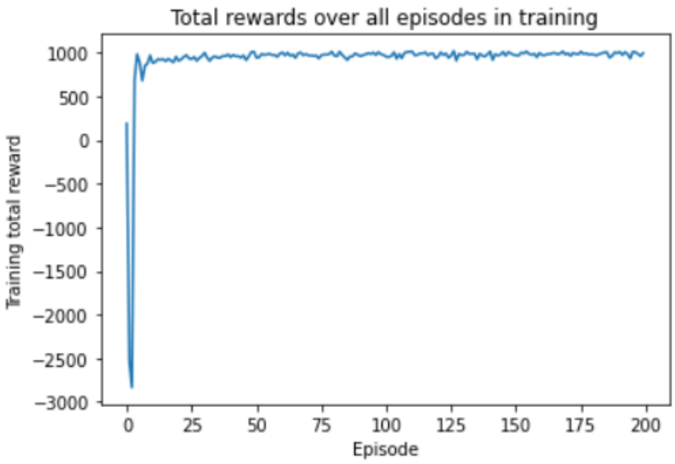


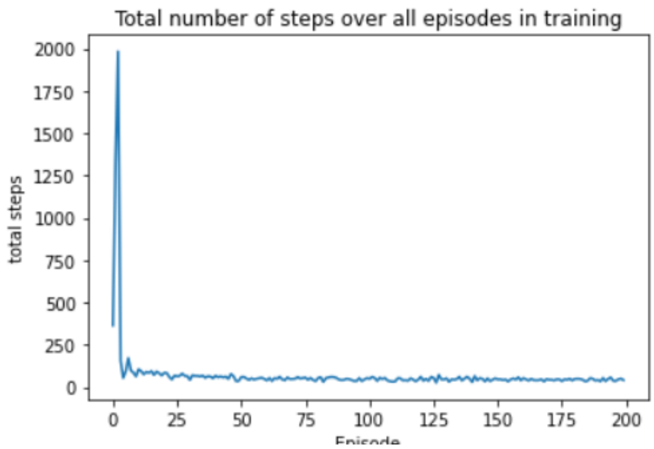


1.4 TD (.) numbers of spots:300

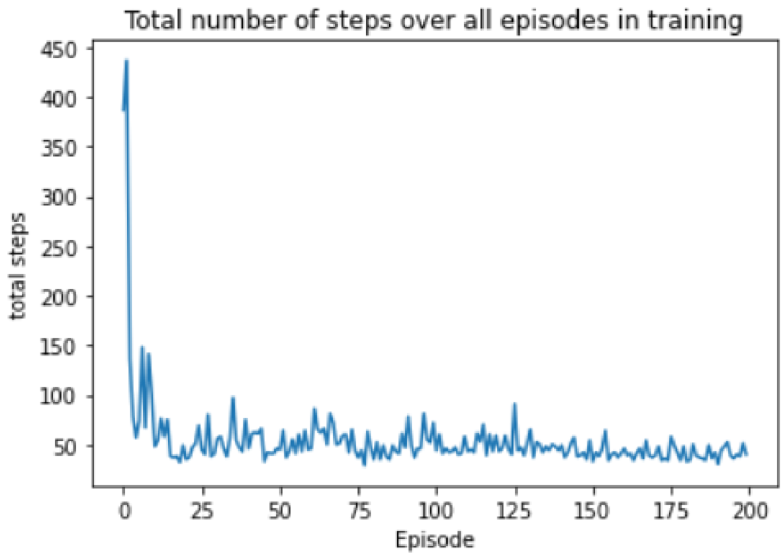
spot_num = 300
alpha = 0.001 #learning rate
gamma = 0.85
epsilon = 0.5
max_epsilon = 1
min_epsilon = 0.01
decay = 0.1
train_episodes = 200

1.4.1 TD (0.1) numbers of spots:300 - variant

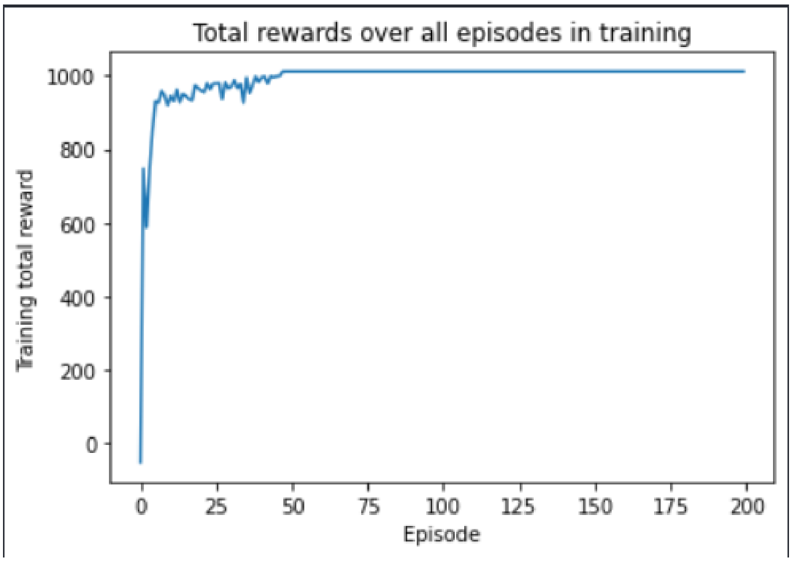


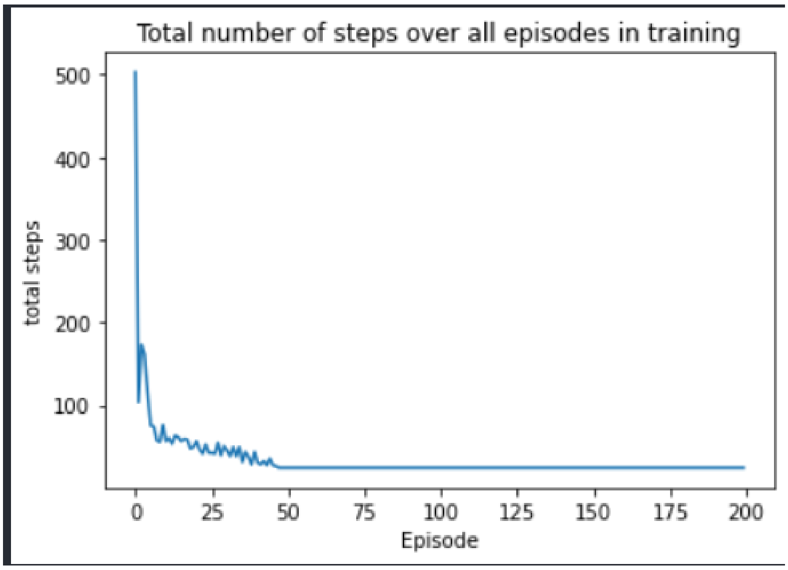


1.4.2 TD (0.5) numbers of spots:300 - variant

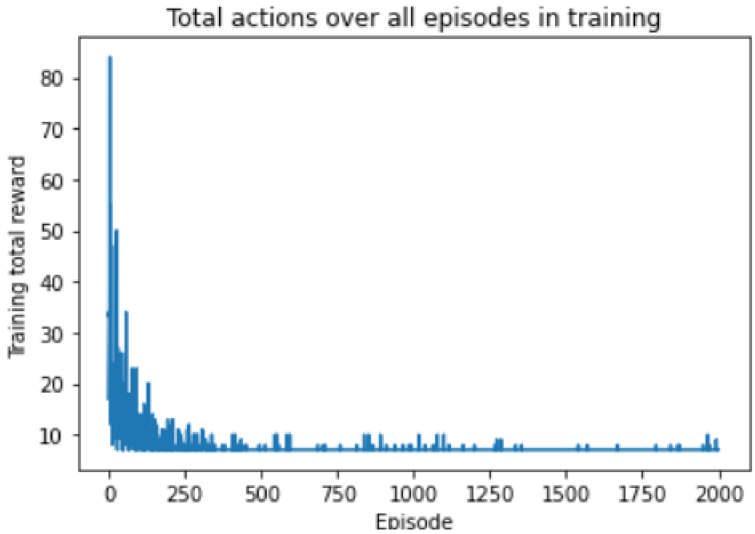


1.4.3 TD (0.9) numbers of spots:300 - variant



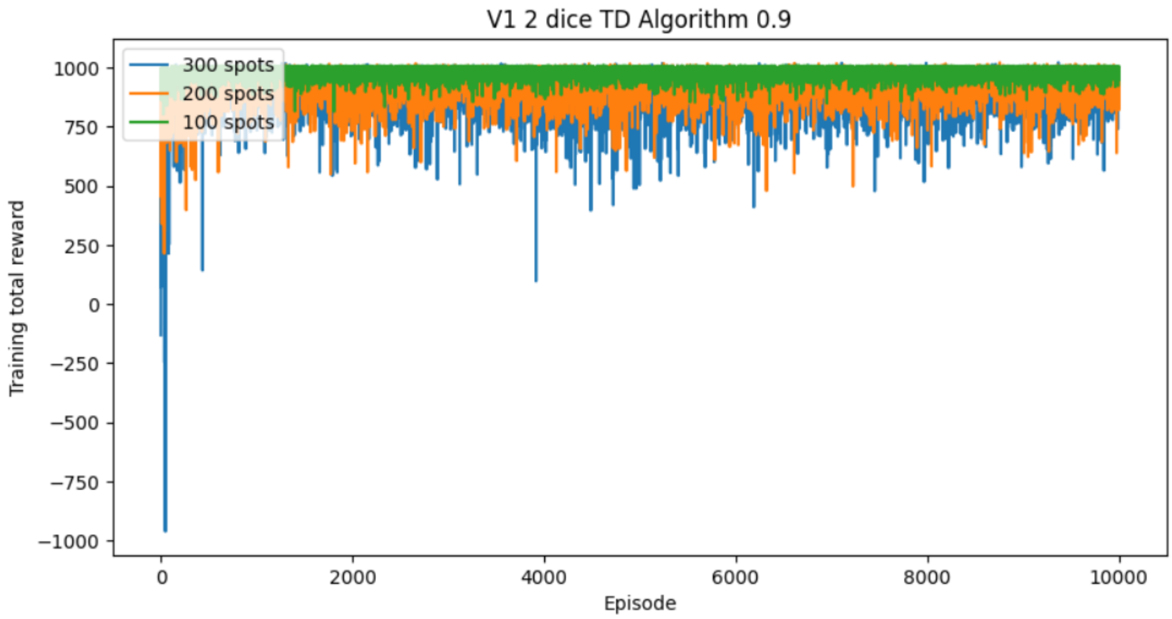


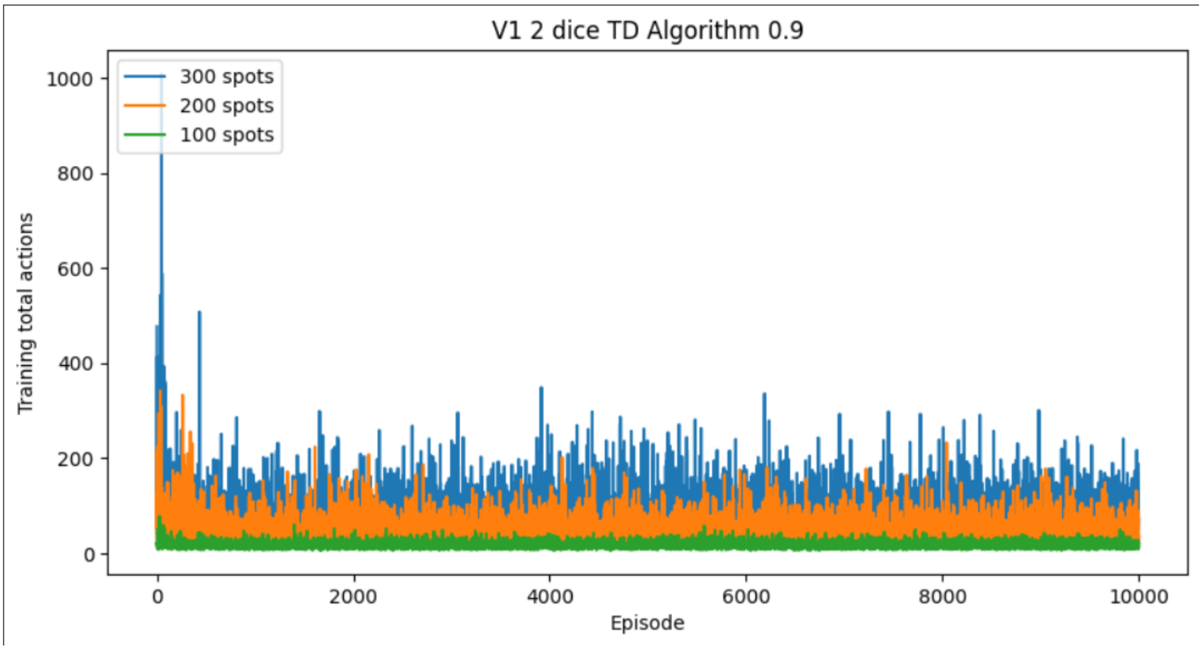
1.5.1 Q-Learning number of spots: 100
Hyper parameters:
alpha = 0.6
discount_factor = 0.8
epsilon = 1
max_epsilon = 1
min_epsilon = 0.01
decay = 0.01
train_episodes = 2000



2.1.1 V1 2 dice TD(0.9)

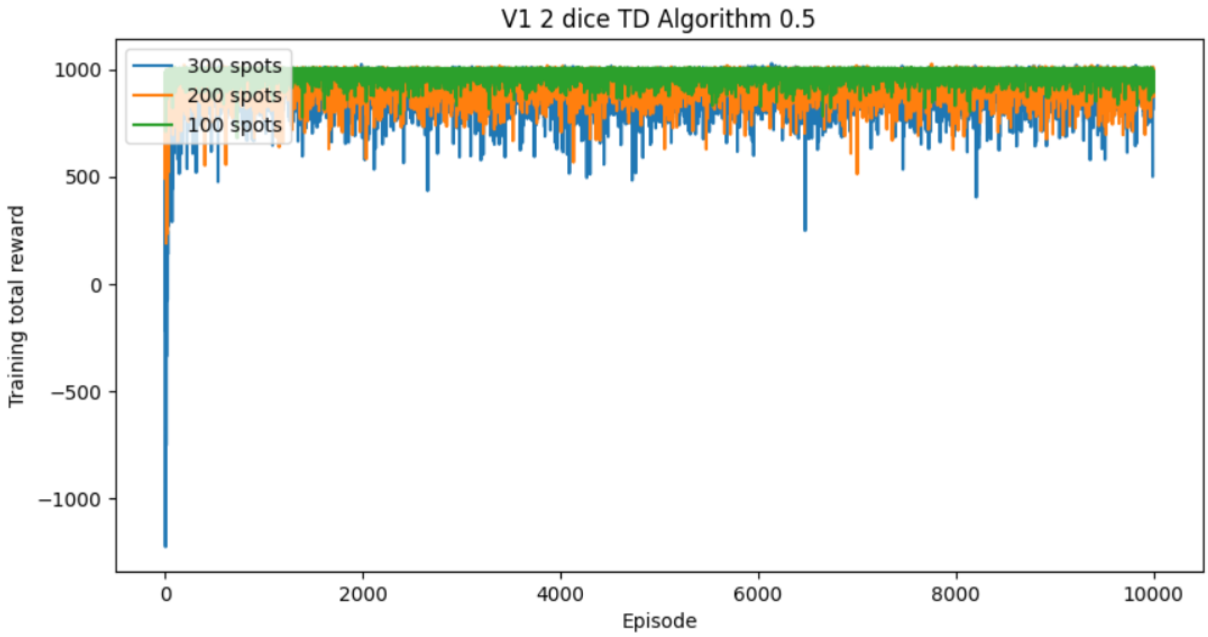
	100 spots	200 spots	300 spots
training score average	979.6655	939.2738	911.8864
training actions average	17.696	47.9912	75.7191

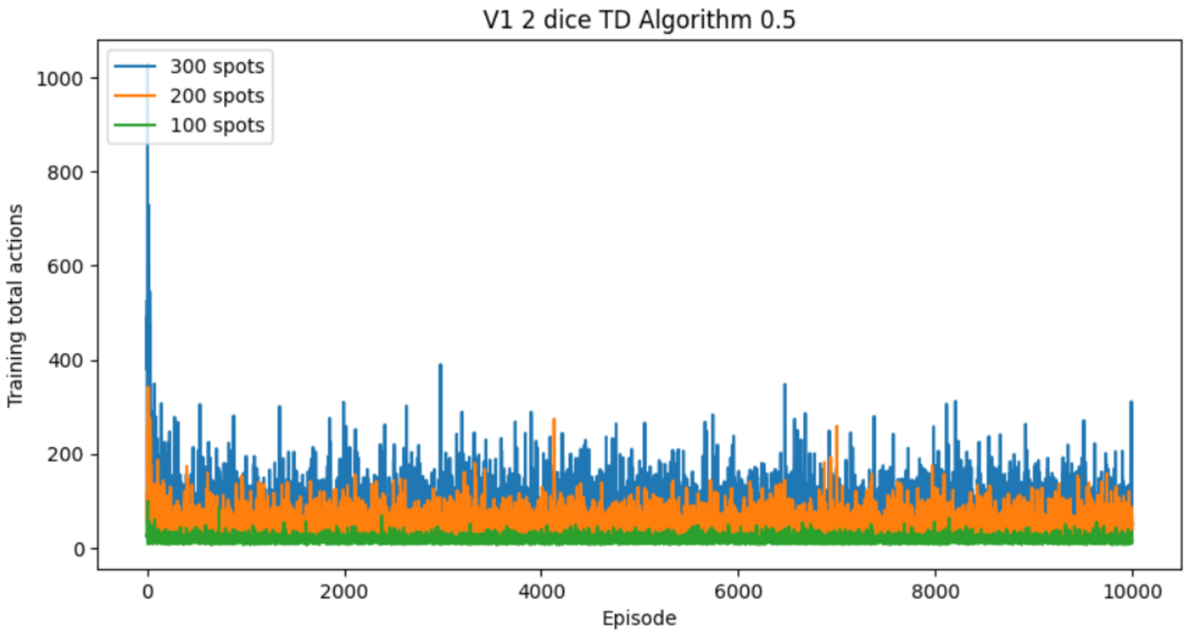




2.1.2 V1 2 dice TD(0.5)

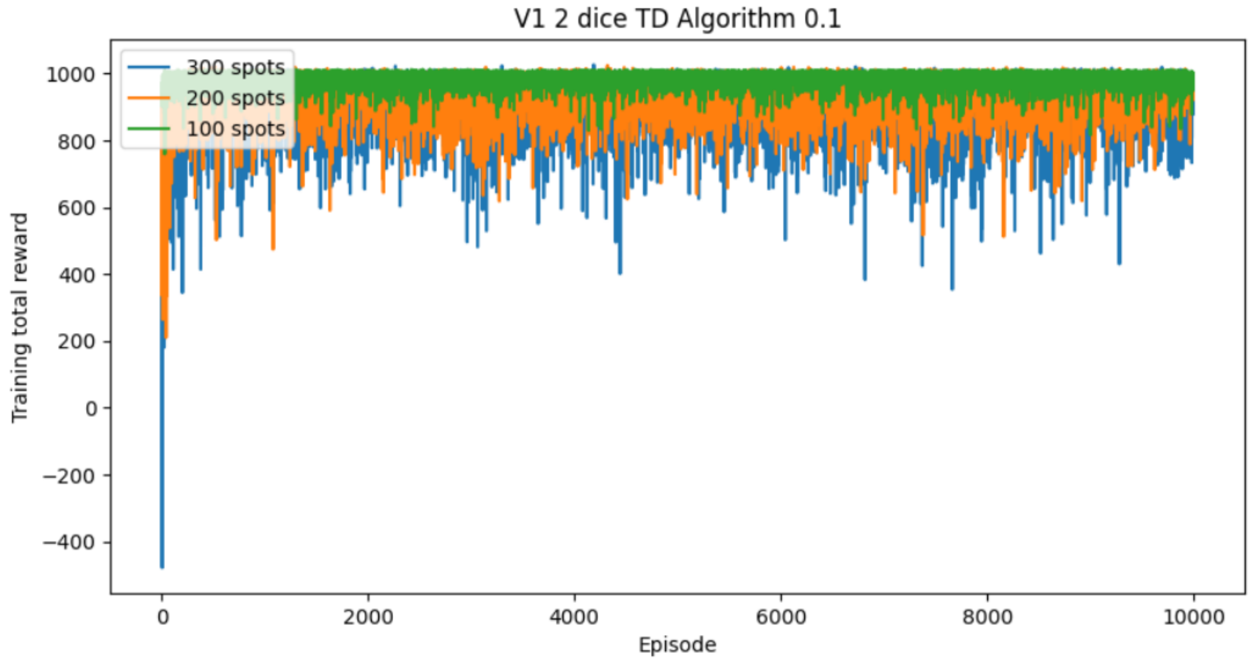
	100 spots	200 spots	300 spots
training score average	980.1065	943.4562	916.2117
training actions average	17.3615	45.6948	72.7063

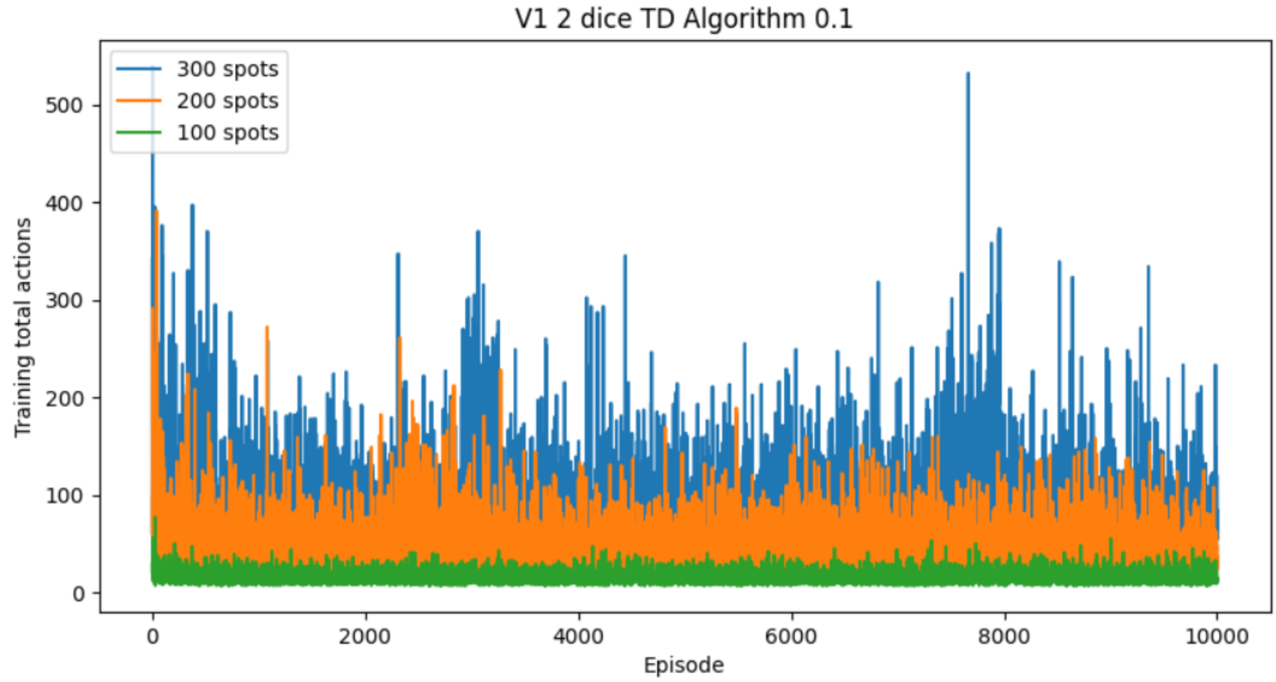




2.1.3 V1 2 dice TD(0.1)

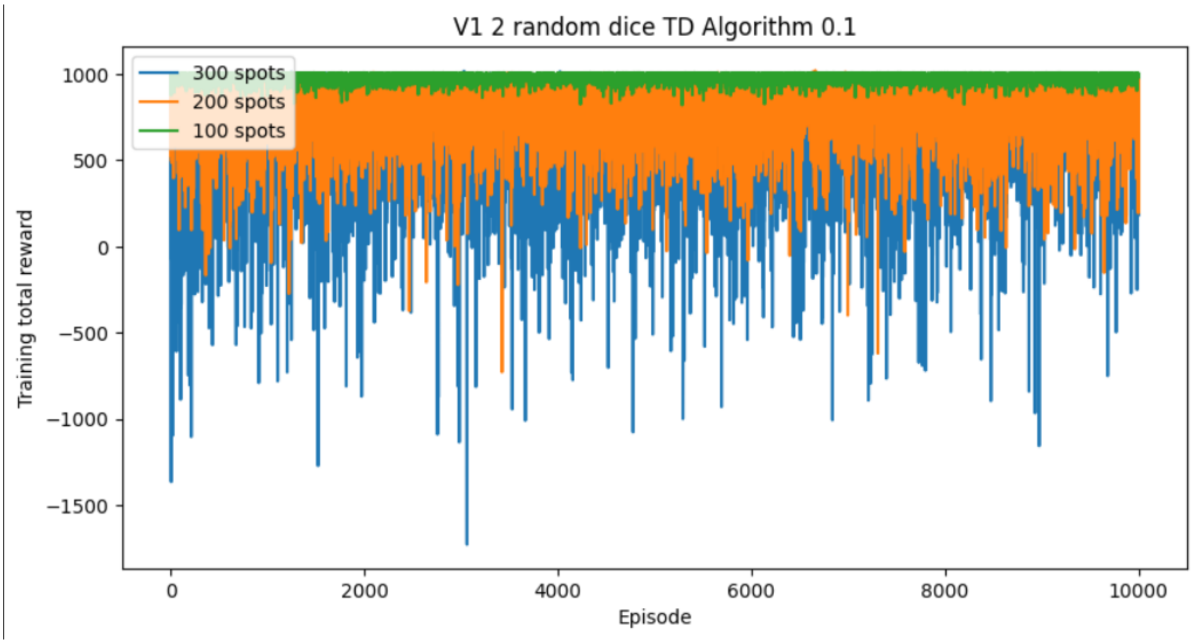
	100 spots	200 spots	300 spots
training score average	984.7376	943.8392	915.1968
training actions average	15.8789	45.9978	74.2267

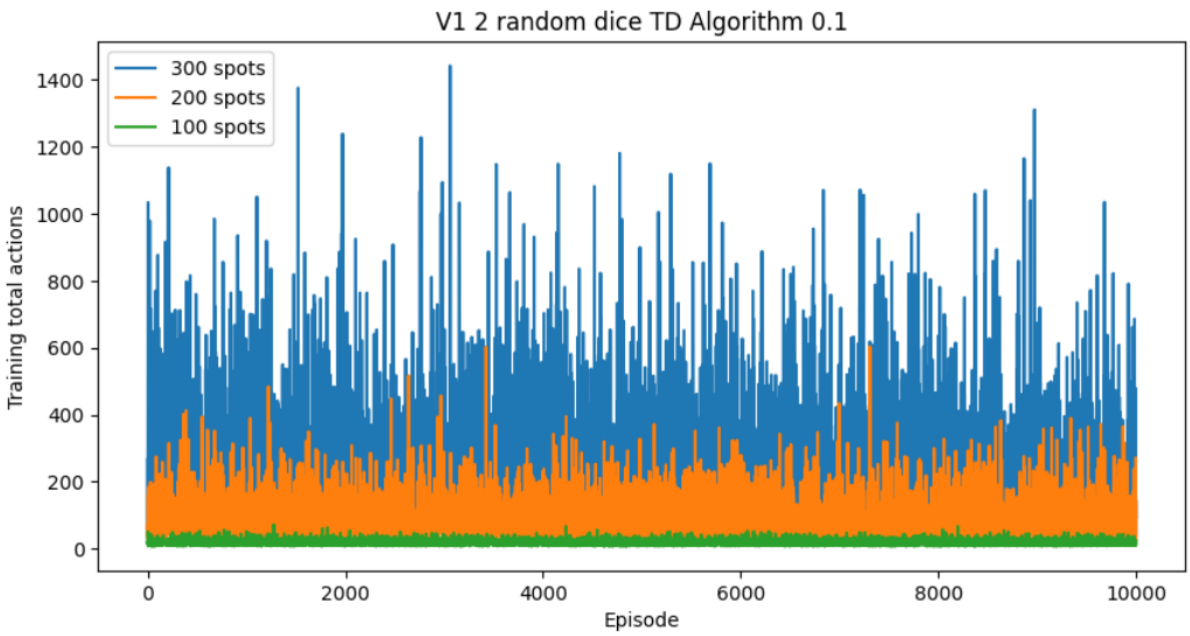




3.1.1 V1 2 random dice TD(0.1)

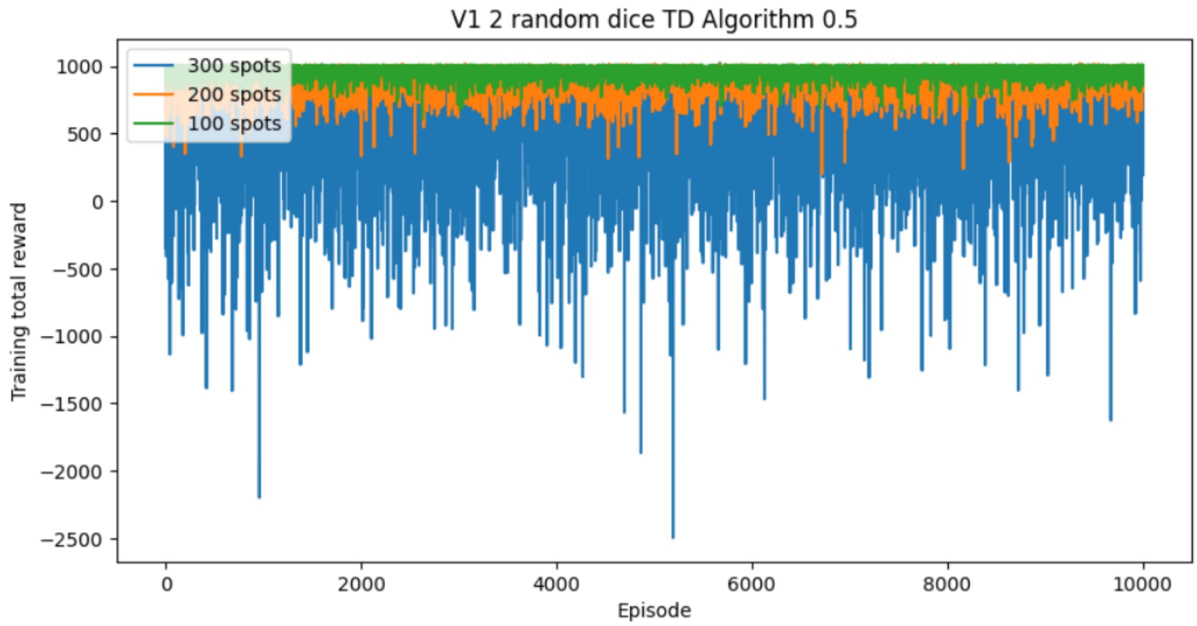
	100 spots	200 spots	300 spots
dice	[3, 3, 2, 4, 3, 2] [6, 1, 1, 5, 6, 1]	[5, 3, 2, 4, 2, 4] [3, 2, 2, 6, 4, 6]	[4, 5, 3, 5, 3, 3] [3, 4, 6, 4, 3, 2]
training score average	981.2218	805.4742	676.8139
training actions average	18.8597	87.4578	198.6021

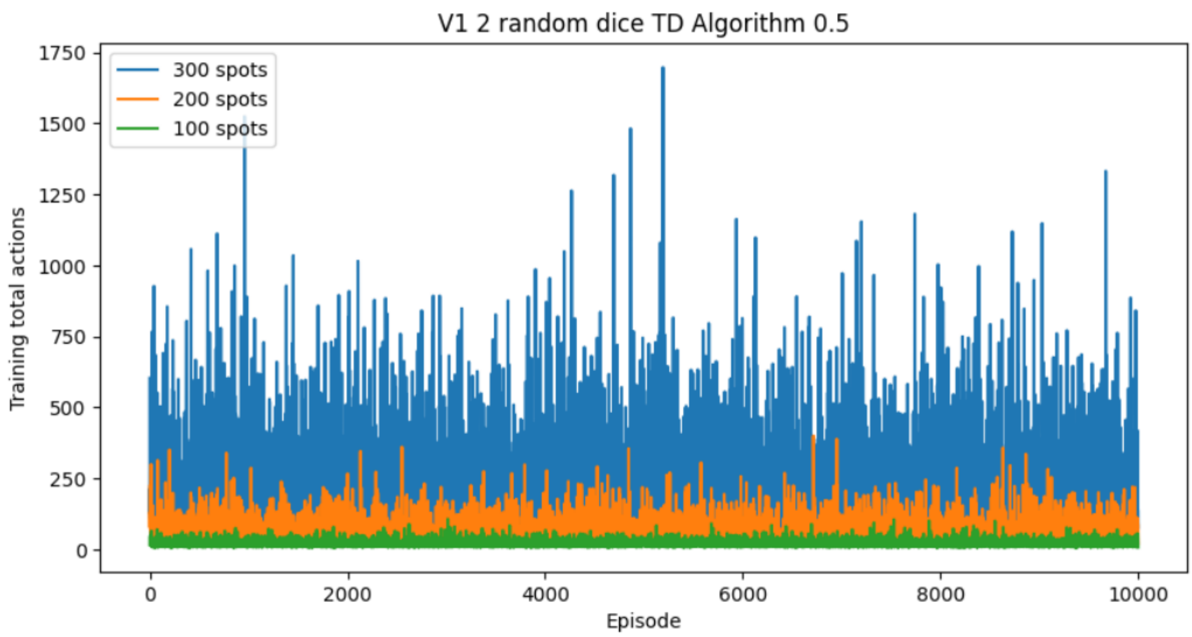




3.1.2 V1 2 random dice TD(0.5)

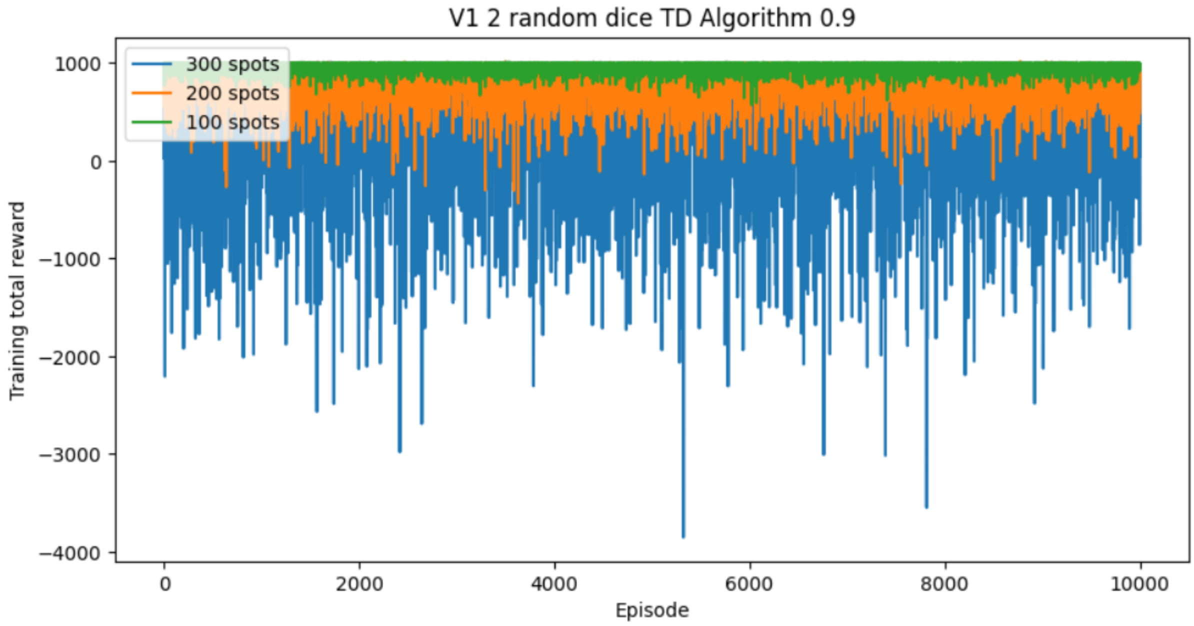
	100 spots	200 spots	300 spots
dice	[3, 1, 1, 6, 5, 2] [4, 6, 4, 5, 6, 1]	[5, 5, 6, 5, 5, 6] [4, 3, 4, 3, 1, 6]	[3, 2, 1, 4, 4, 5] [5, 3, 4, 2, 5, 5]
training score average	961.4152	919.4236	616.1816
training actions average	22.0053	58.4404	198.4449

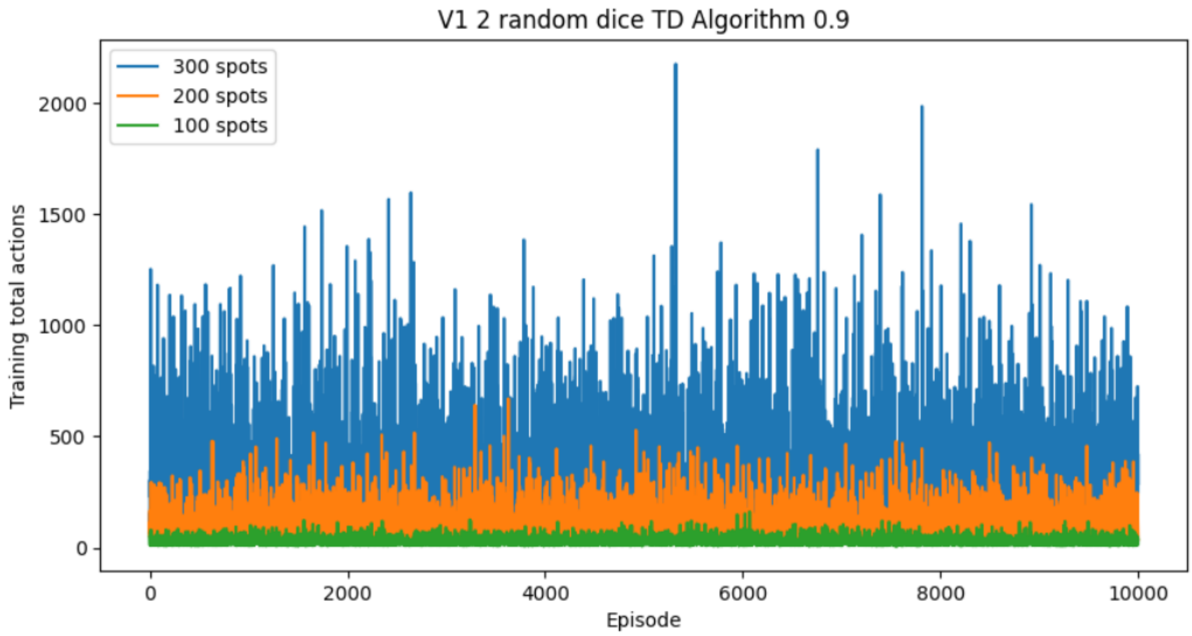




3.1.2 V1 2 random dice TD(0.9)

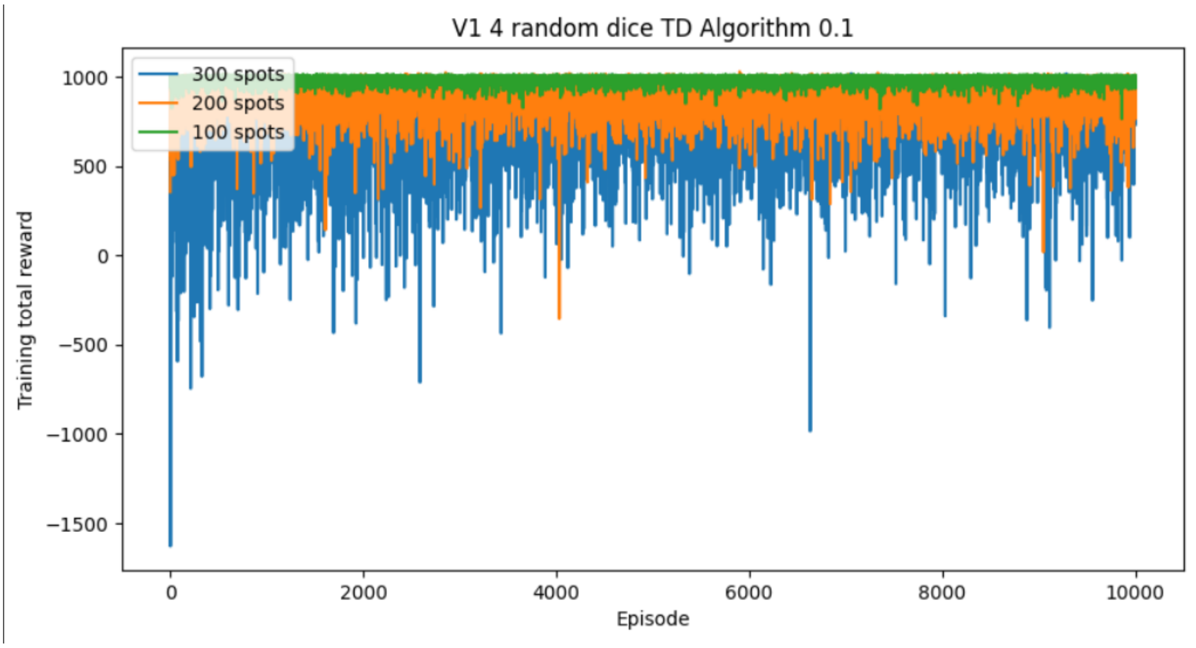
	100 spots	200 spots	300 spots
dice	[2, 2, 5, 2, 1, 1] [4, 1, 1, 6, 5, 1]	[3, 6, 1, 1, 3, 2] [5, 4, 4, 5, 2, 5]	[5, 4, 1, 6, 6, 6] [6, 1, 2, 6, 4, 6]
training score average	953.3478	818.5506	428.0858
training actions average	28.9292	97.1349	258.4662

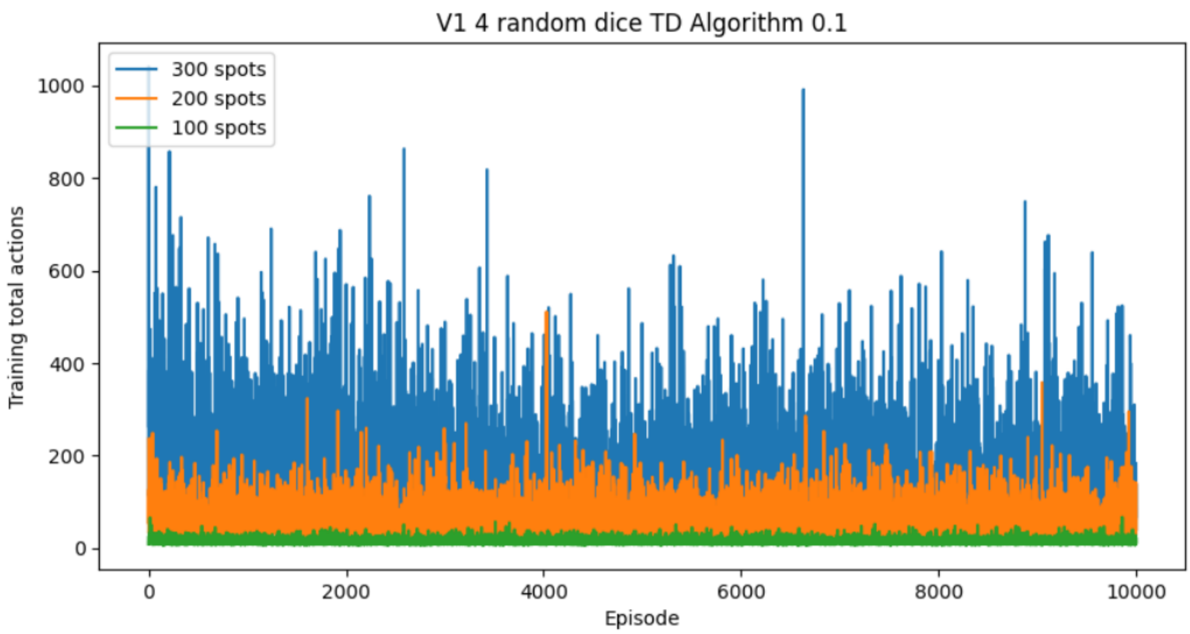




3.2.1 V1 4 random dice TD(0.1)

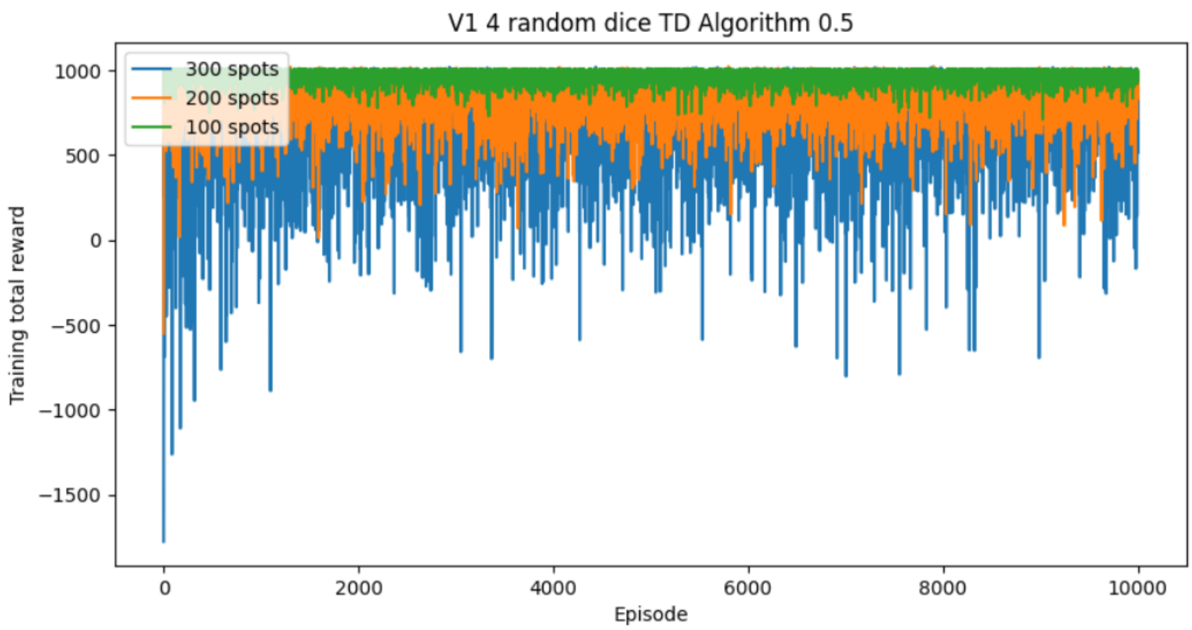
	100 spots	200 spots	300 spots
dice	[2, 5, 5, 3, 5, 6] [6, 6, 4, 5, 4, 4] [1, 6, 1, 3, 2, 4] [3, 2, 4, 6, 1, 3]	[5, 4, 6, 3, 1, 6] [1, 2, 6, 3, 6, 2] [2, 3, 2, 5, 4, 4] [2, 4, 1, 4, 3, 3]	[3, 3, 5, 2, 1, 4] [2, 2, 6, 6, 6, 1] [1, 5, 2, 4, 1, 6] [1, 3, 2, 2, 4, 6]
training score average	983.9816	900.024	768.018
training actions average	15.7169	60.3515	139.146

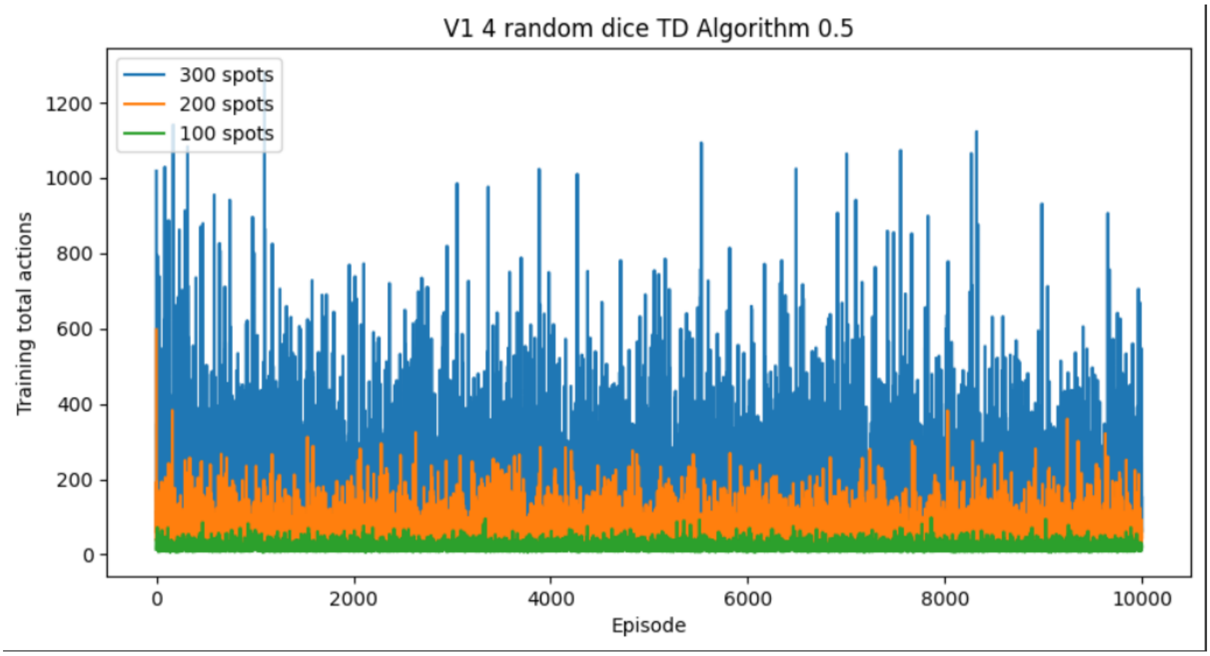




3.2.2 V1 4 random dice TD(0.5)

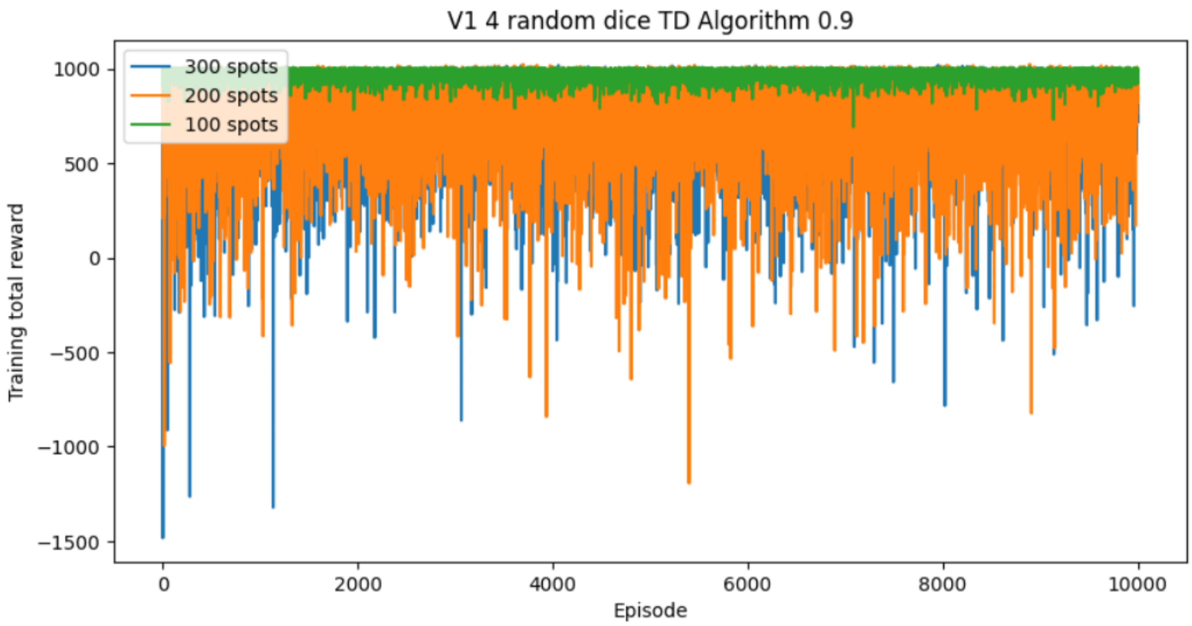
	100 spots	200 spots	300 spots
dice	[5, 6, 5, 5, 1, 5] [3, 6, 6, 3, 5, 4] [1, 3, 4, 6, 1, 6] [1, 2, 3, 4, 3, 1]	[4, 3, 6, 2, 5, 6] [3, 4, 3, 3, 2, 5] [4, 6, 2, 5, 5, 4] [6, 3, 6, 2, 5, 6]	[4, 5, 3, 6, 1, 1] [2, 6, 4, 4, 5, 5] [2, 5, 6, 2, 1, 6] [1, 3, 1, 5, 6, 1]
training score average	971.9163	879.3213	755.1231
training actions average	20.2437	64.4192	166.7634

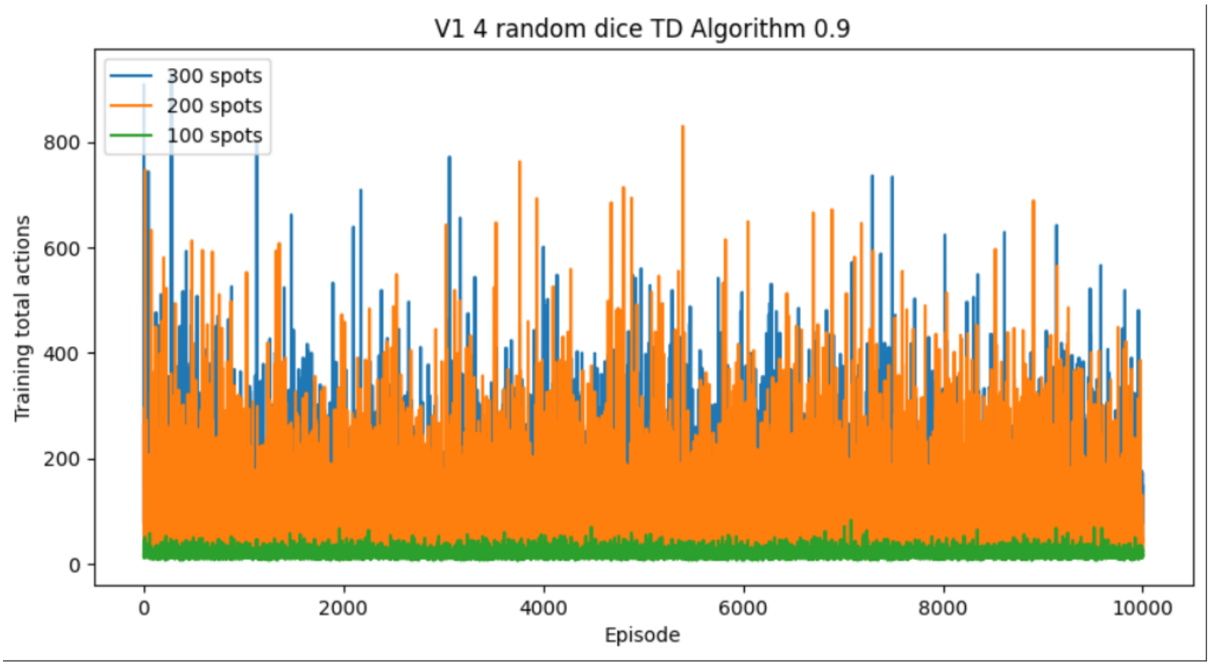




3.2.2 V1 4 random dice TD(0.9)

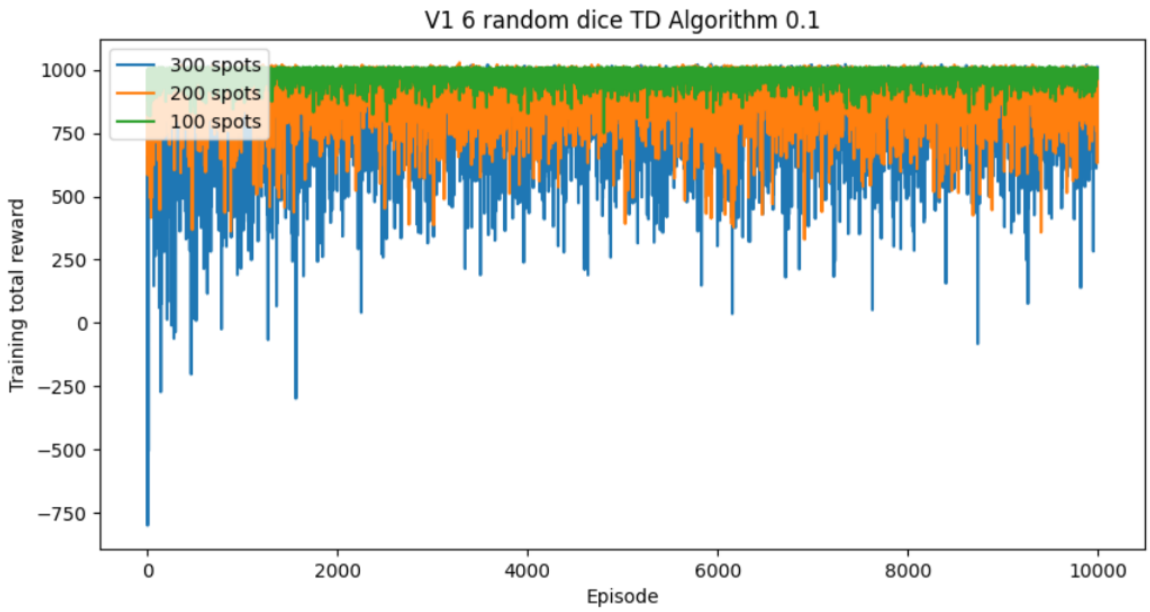
	100 spots	200 spots	300 spots
dice	[2, 5, 5, 3, 5, 6] [6, 6, 4, 5, 4, 4] [1, 6, 1, 3, 2, 4] [3, 2, 4, 6, 1, 3]	[5, 4, 6, 3, 1, 6] [1, 2, 6, 3, 6, 2] [2, 3, 2, 5, 4, 4] [2, 4, 1, 4, 3, 3]	[3, 3, 5, 2, 1, 4] [2, 2, 6, 6, 6, 1] [1, 5, 2, 4, 1, 6] [1, 3, 2, 2, 4, 6]
training score average	983.9816	900.024	768.018
training actions average	15.7169	60.3515	139.146

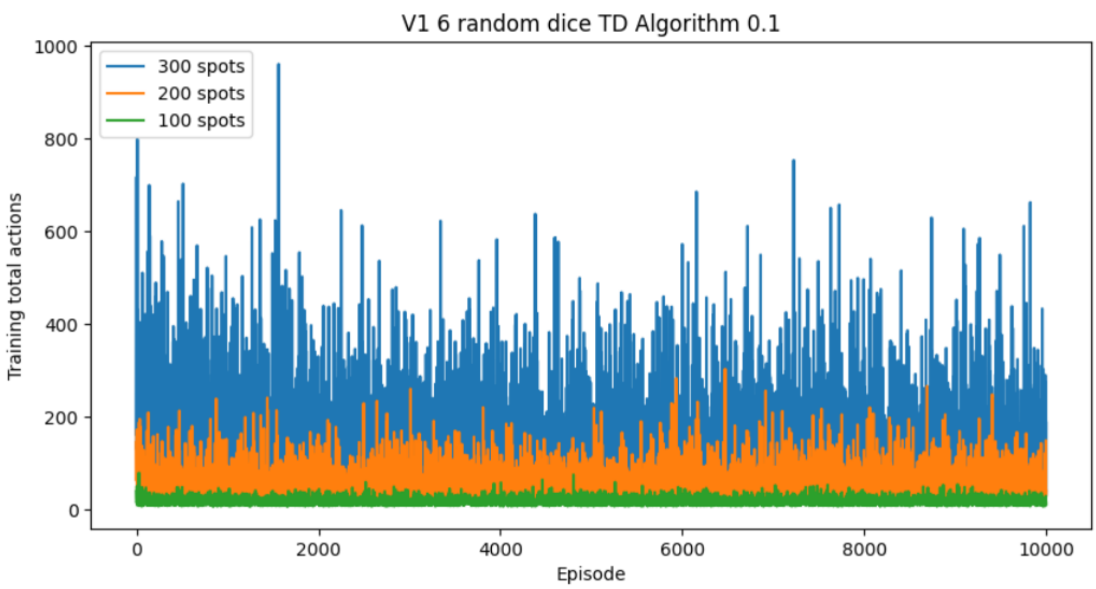




3.3.1 V1 6 random dice TD(0.1)

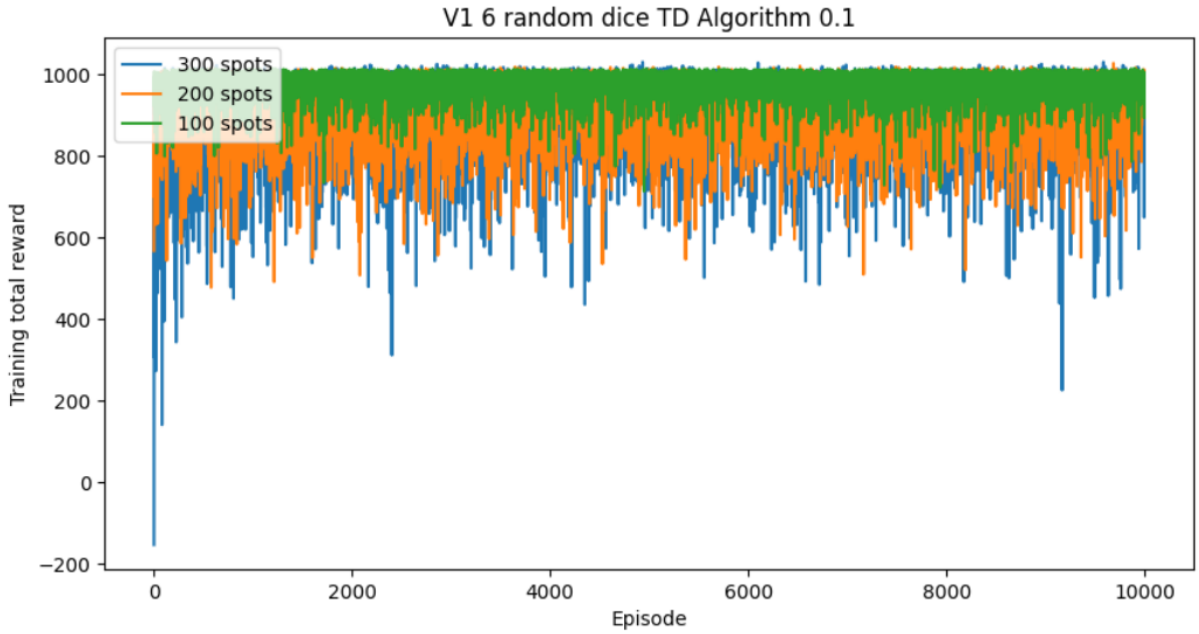
	100 spots	200 spots	300 spots
dice	[2, 3, 5, 2, 3, 2] [5, 4, 6, 5, 6, 1] [5, 1, 5, 6, 3, 1] [1, 6, 2, 1, 1, 2] [3, 4, 2, 3, 1, 6] [4, 4, 2, 4, 1, 3]	[1, 3, 2, 1, 2, 6] [6, 1, 6, 6, 3, 4] [6, 4, 6, 6, 4, 3] [6, 3, 1, 6, 1, 4] [5, 3, 6, 3, 6, 3] [2, 6, 1, 1, 2, 6]	[1, 3, 6, 5, 1, 5] [3, 4, 3, 5, 2, 2] [1, 6, 5, 5, 2, 5] [2, 4, 6, 2, 5, 3] [4, 2, 5, 1, 3, 1] [3, 6, 6, 2, 3, 1]
training score average	981.1896	920.0358	845.9697
training actions average	17.3819	52.2617	126.7093

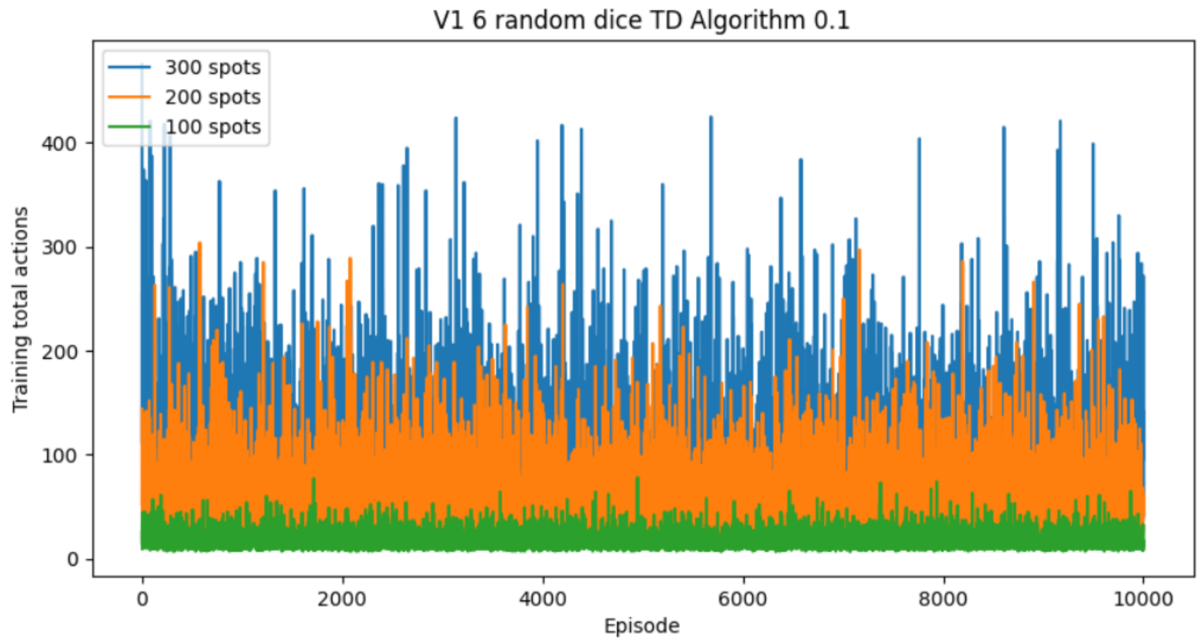




3.3.2 V1 6 random dice TD(0.5)

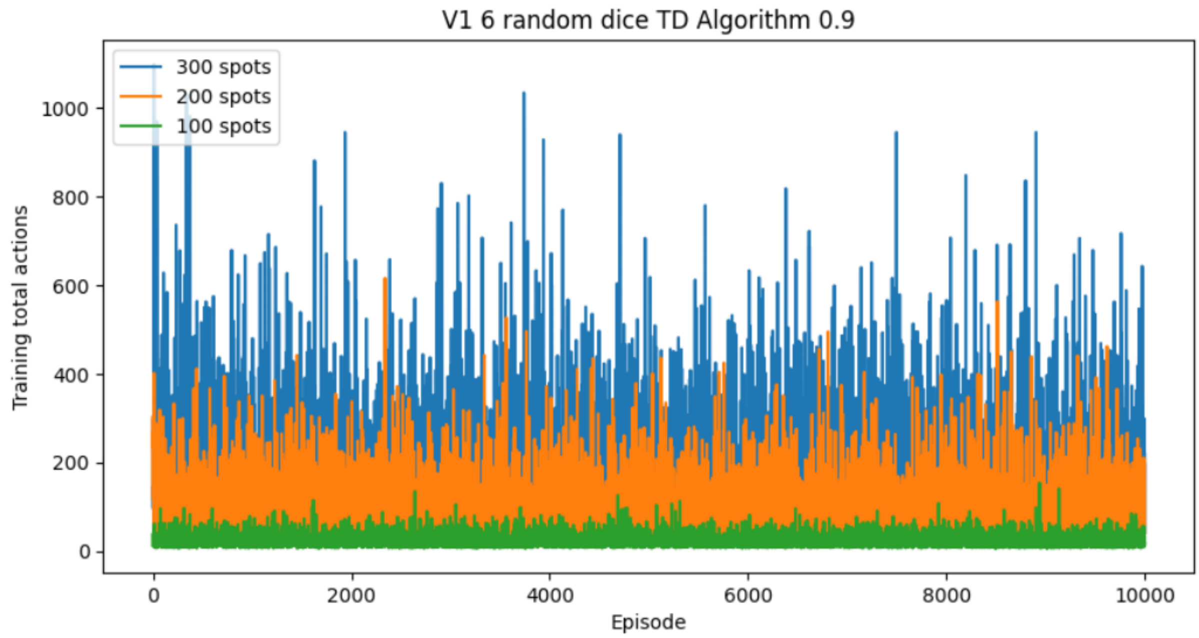
	100 spots	200 spots	300 spots
dice	[2, 4, 4, 2, 3, 5] [4, 6, 6, 2, 2, 2] [6, 1, 5, 2, 4, 4] [4, 5, 6, 3, 3, 4] [5, 3, 5, 2, 1, 2] [3, 3, 3, 2, 5, 3]	[3, 3, 6, 3, 3, 1] [5, 2, 4, 5, 6, 6] [6, 3, 4, 2, 2, 6] [5, 5, 6, 1, 3, 3] [1, 6, 1, 3, 2, 6] [3, 5, 6, 2, 4, 5]	[5, 4, 1, 1, 5, 1] [6, 2, 5, 2, 6, 4] [3, 1, 2, 2, 2, 5] [6, 5, 3, 3, 2, 5] [5, 2, 1, 3, 1, 1] [4, 4, 2, 1, 2, 1]
training score average	977.2403	930.4627	914.4905
training actions average	17.9477	54.9808	85.59

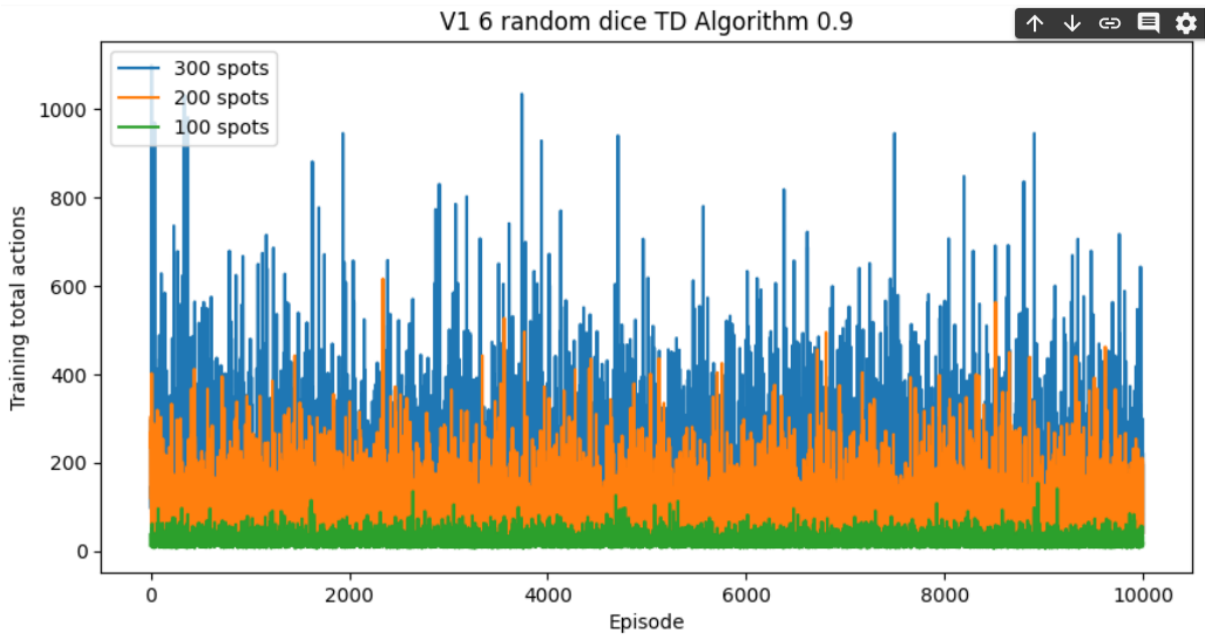




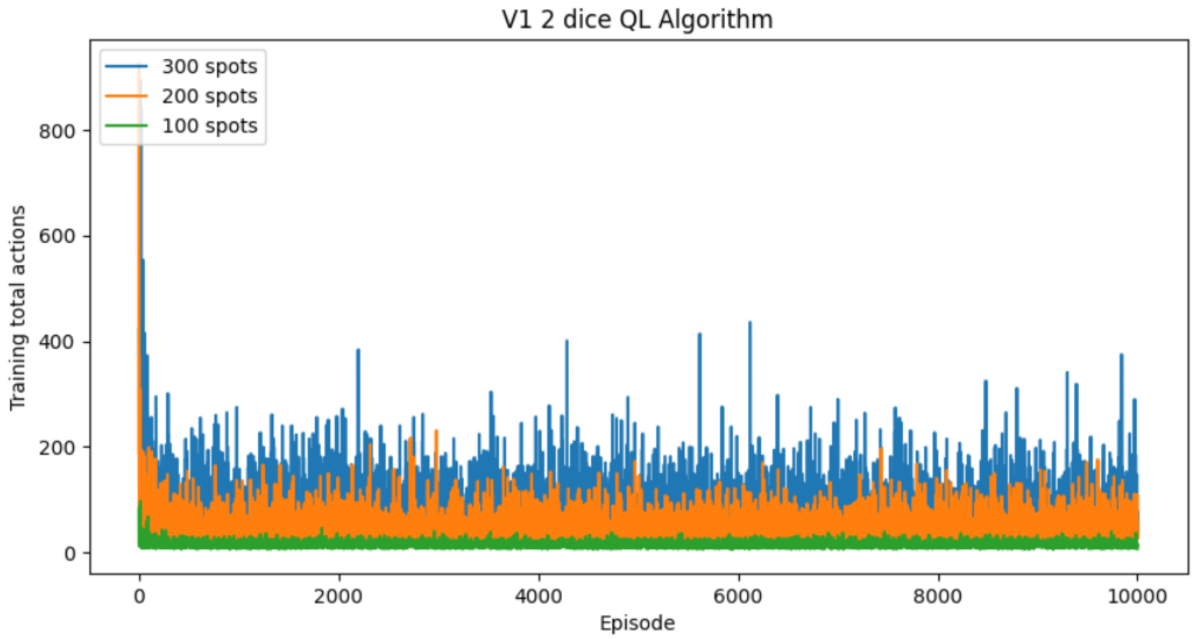
3.3.3 V1 6 random dice TD(0.9)

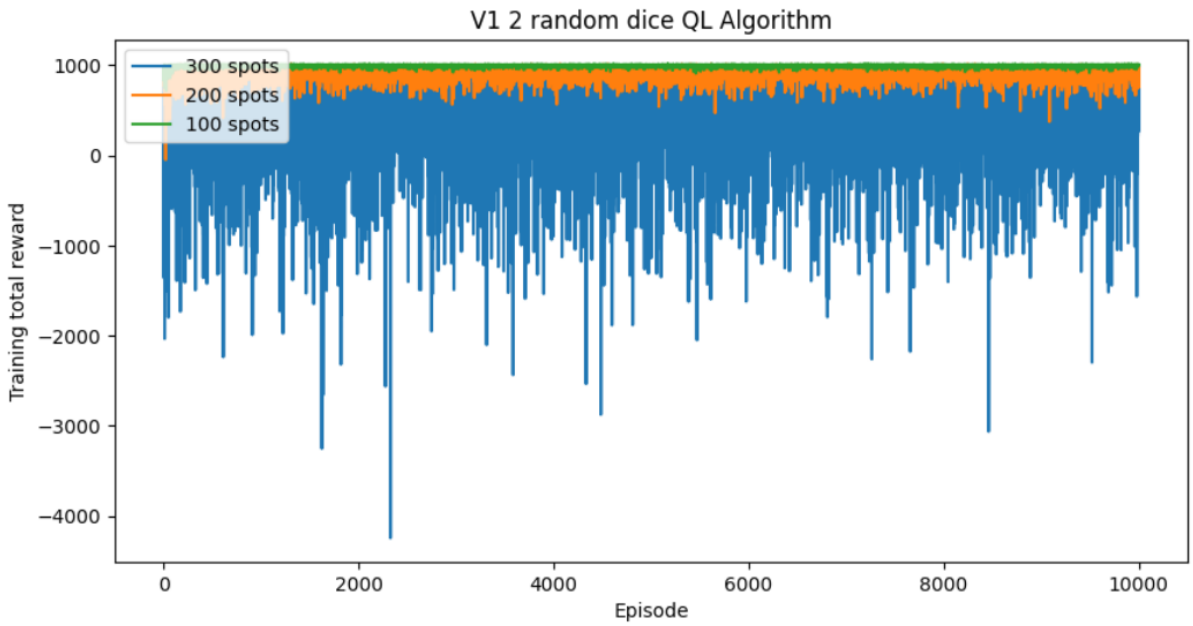
	100 spots	200 spots	300 spots
dice	[6, 5, 6, 2, 4, 6] [6, 3, 2, 1, 3, 5] [1, 1, 4, 6, 4, 6] [1, 6, 6, 4, 4, 4] [5, 3, 2, 6, 4, 1] [2, 4, 3, 6, 4, 1]	[6, 6, 4, 5, 6, 1] [5, 6, 4, 6, 3, 1] [6, 4, 2, 5, 5, 5] [6, 6, 4, 1, 1, 4] [4, 1, 6, 1, 1, 4] [1, 2, 6, 5, 2, 6]	[3, 3, 3, 1, 5, 6] [4, 4, 6, 4, 2, 3] [4, 4, 4, 5, 3, 3] [3, 3, 3, 3, 6, 5] [5, 4, 2, 1, 2, 4] [5, 3, 2, 6, 4, 3]
training score average	947.6267	930.4627	778.5963
training actions average	23.5613	86.7063	147.6412

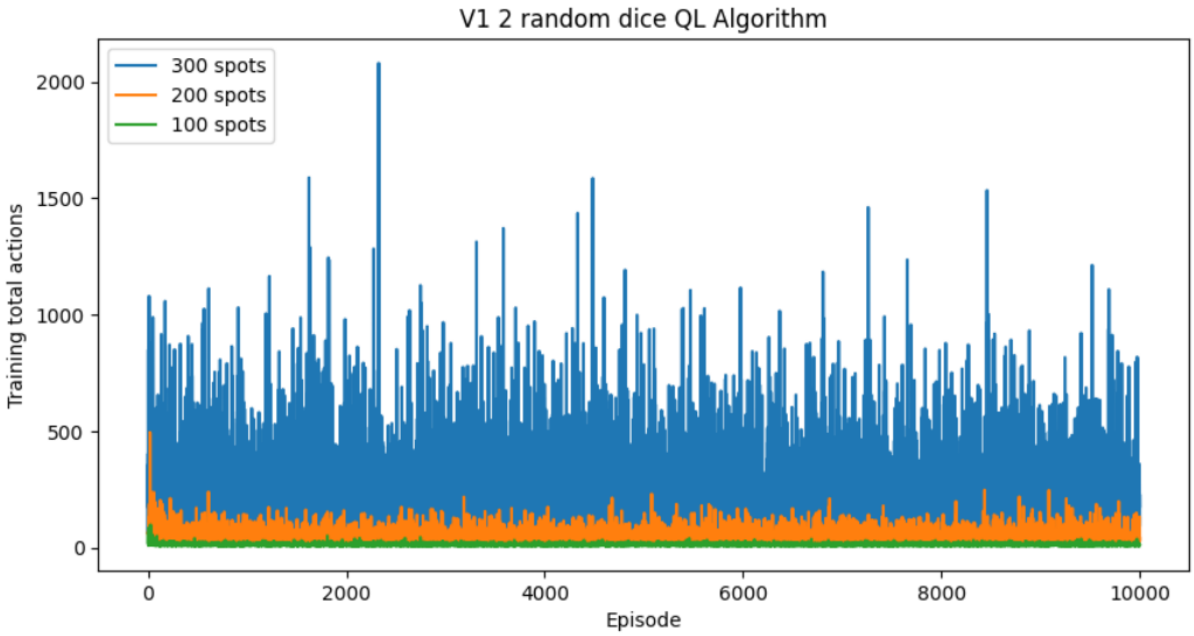


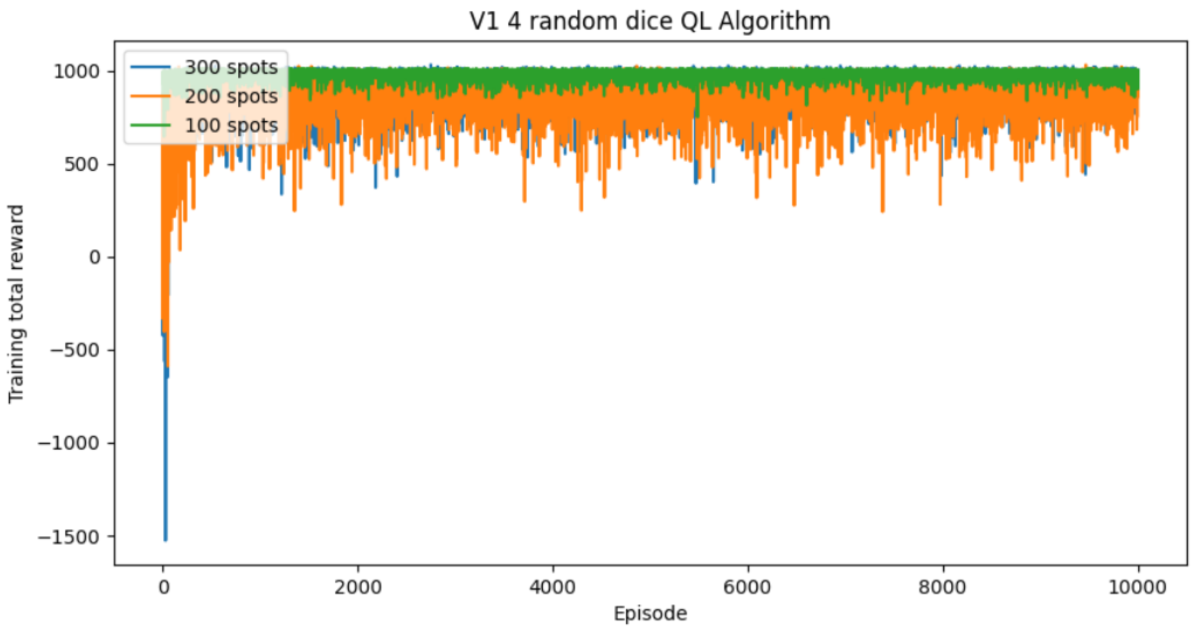


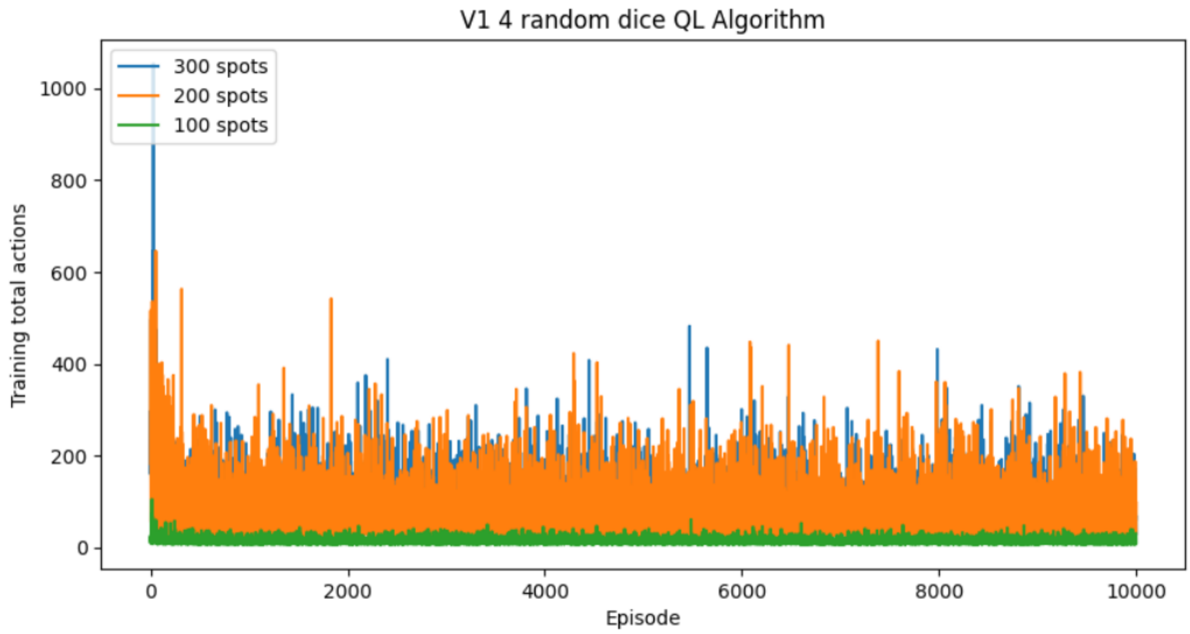
3.4 V1 Q-Learning



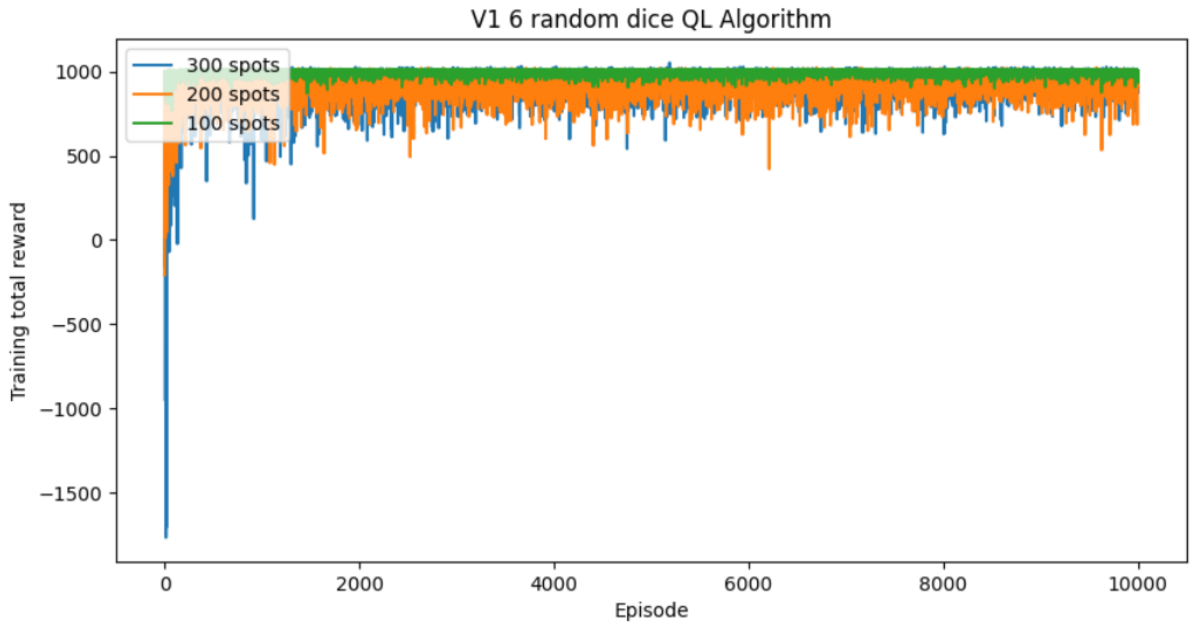



V1 2 random dice QL Algorithm
300 spots
200 spots
100 spots
Training total actions
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
Episode

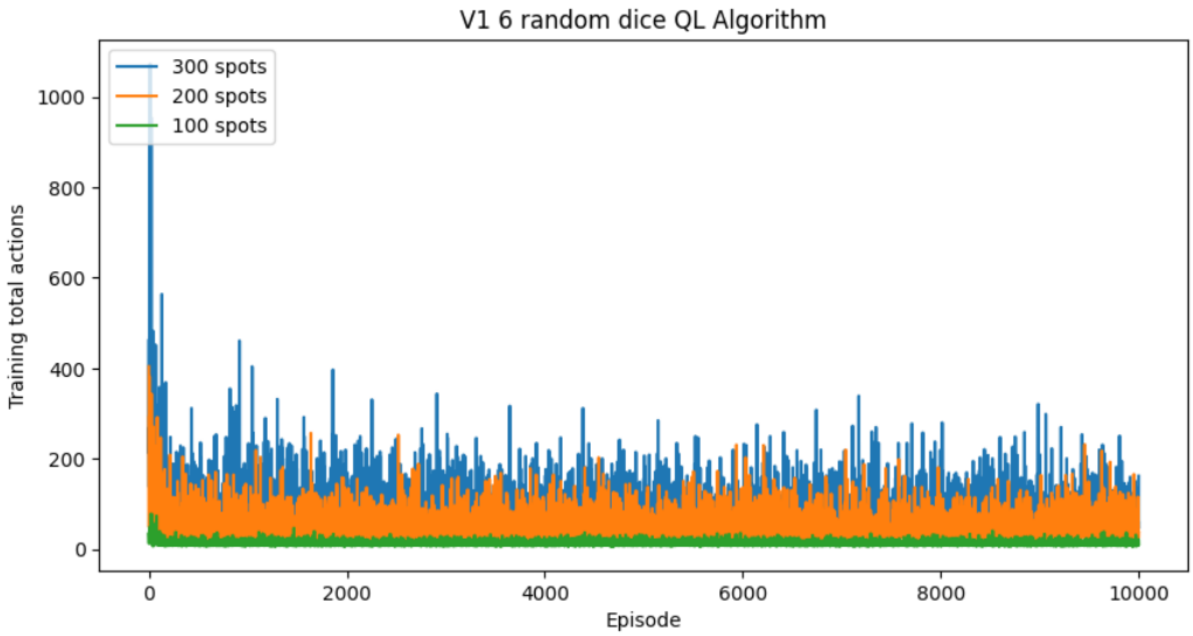

V1 4 random dice QL Algorithm
300 spots
200 spots
100 spots
Training total reward
1000
500
0
−500
−1000
−1500
0
2000
4000
6000
8000
10000
Episode



V1 4 random dice QL Algorithm
300 spots
200 spots
100 spots
Training total actions
1000
800
600
400
200
0
0
2000
4000
6000
8000
10000
Episode



V1 6 random dice QL Algorithm
300 spots
200 spots
100 spots
Training total reward
1000
500
0
−500
−1000
−1500
0
2000
4000
6000
8000
10000
Episode



4. Discussion

Temporal difference (TD) learning refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo methods, and perform updates based on current estimates, like dynamic programming methods. When $\lambda = 1$, only the last term is kept and this is essentially Monte Carlo method, as the state, action process goes all the way to the end, when $\lambda = 0$, the term reduces to 1-step TD method, and for $0 < \lambda < 1$, the method becomes a mixed of weighted TD(n) method. Sometimes, when I implemented the experiments it was hard to tailor the hyper-parameters when I tried to have fewer steps during the training. Compared to Q-Learning, TD lambda conversely predicts rewards and action more efficiently. However Q-Learning with eps is more flexible to find a potentially optimal action during each step. It is such a time-consuming assignment I have ever done, but I can learn more about the math about each RL algorithm. It is quite impressive. I learned a lot about the trade-off between Q-Learning and TD lambda.