1 The pseudocode is taken from Least-Squares Policy Iteration. env is the environment from which the (s, a, r, s') (state, action, reward, next state) experience tuple generated, k is the number of dimensions of the feature vector, ϕ is the feature extractor, which is an autoencoder implemented by neural network in this case, and γ is the discount factor. If there are n number of possible actions

```
$LSTDQ(\text{env}, k, \phi, \gamma)$
    $\mathbf{A} \longleftarrow \mathbf{0}$                                   ($k \times k$ matrix)
    $b \longleftarrow \mathbf{0}$                                   ($k \times n$ vector)
    while not done:
        generate $(s, a, r, s', \text{done})$ by interacting with env
        $A \longleftarrow A + \phi(s) \cdot (\phi(s) - \gamma\phi(s'))^T$
        $t \longleftarrow r + \gamma \cdot \max_a \left[\phi^T(s') \cdot \theta_a\right]$
        $b[n] \longleftarrow b[n] + \phi(s) \cdot t$
    $\theta \longleftarrow \mathbf{A}^{-1} \cdot b$
    return $\theta$
```

2 The algorithm is implemented in the `.ipynb` file.

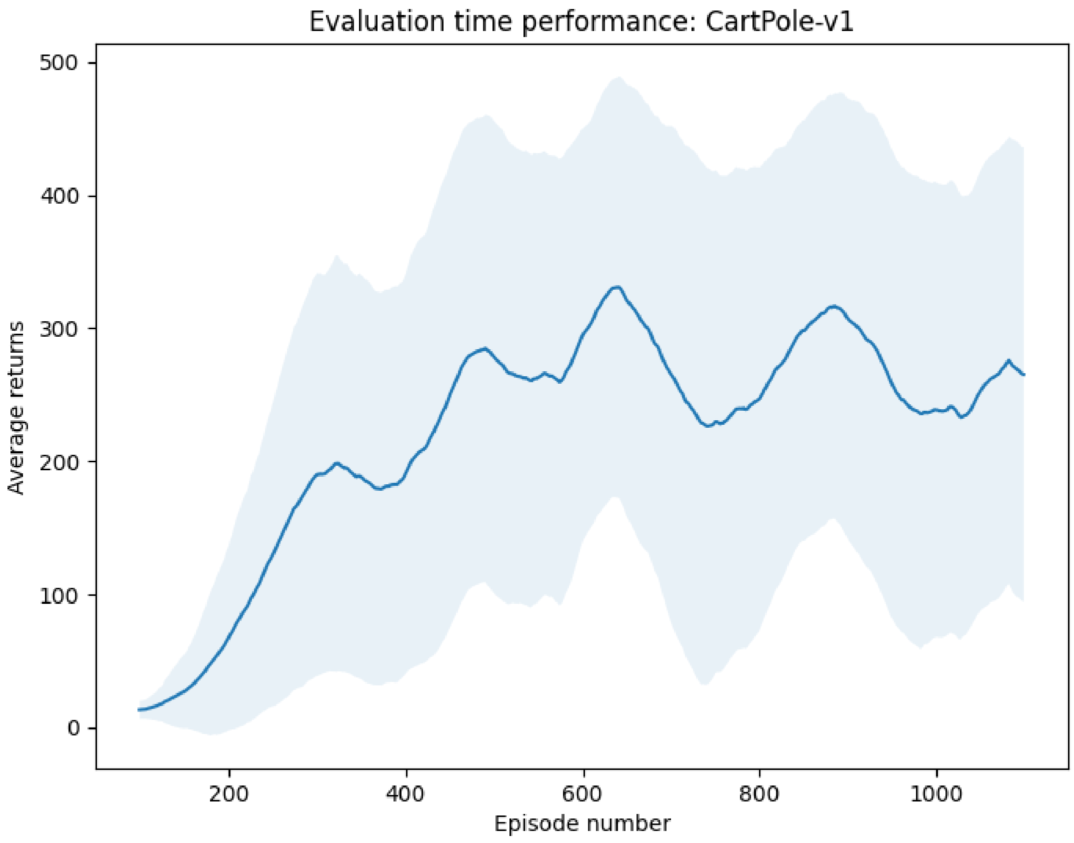

Figure 1: Average returns from 10 training epochs when the agent is evaluated every 10 episodes is plotted as the solid line with one standard deviation of the 10 trials shaded for the `CartPole-v1` environment.

3 LSTDQ algorithm did not perform well on the control tasks. The cumulative returns when the agent is evaluated every tenth episode, averaged over 10 training epochs, over a total of 11000 episodes, is plotted in Fig 1 for the `CartPole-v1` environment.

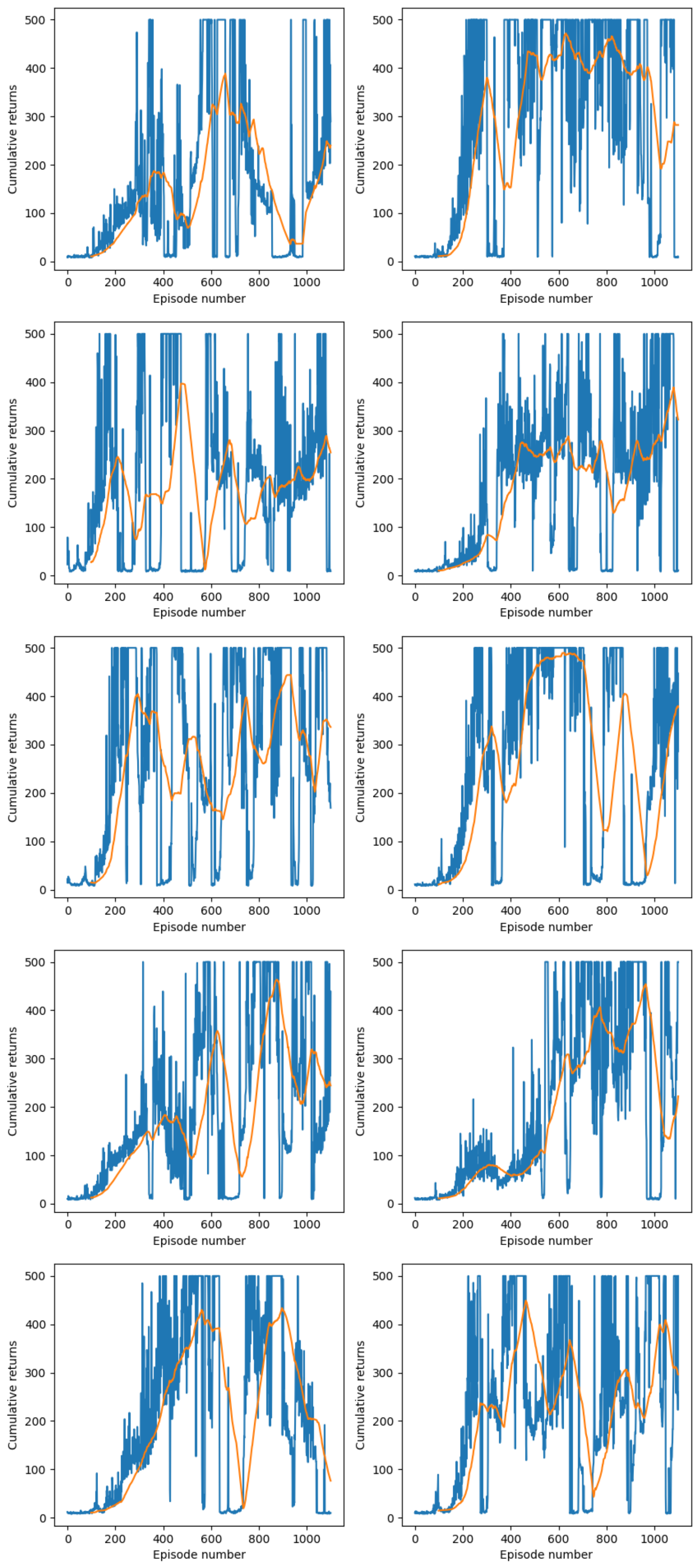

Figure 2: Returns from 10 individual training epochs when the agent is evaluated every 10 episodes is plotted as the blue line, and the moving average over 100 values as the orange line for the `CartPole-v1` environment.

As seen for the plots for the individual training epochs, the learning curves are quite noisy, and therefore the standard deviation does *not* decrease with increasing number of episodes in each epoch. States in the `CartPole-v1` environment are 4×1 dimensional vectors and action space consists of 2 options.

4 For all these experiments 20 episodes of training the autoencoder was followed by 180 episodes of training the LSTDQ agent.

States in `Acrobot-v1` environment are 6×1 dimensional vectors and action space consists of 3 options. LSTDQ could learn to solve the task only in some of the epochs of the training. The LSTDQ agent is prone to *catastrophic interference* where an update to the weights for an action for a state changes the Q-function for all the states, and the agent unable to solve the task in future episodes. The LSTDQ agent failed to solve the task in *one* of the ten epochs as seen in Fig 5.

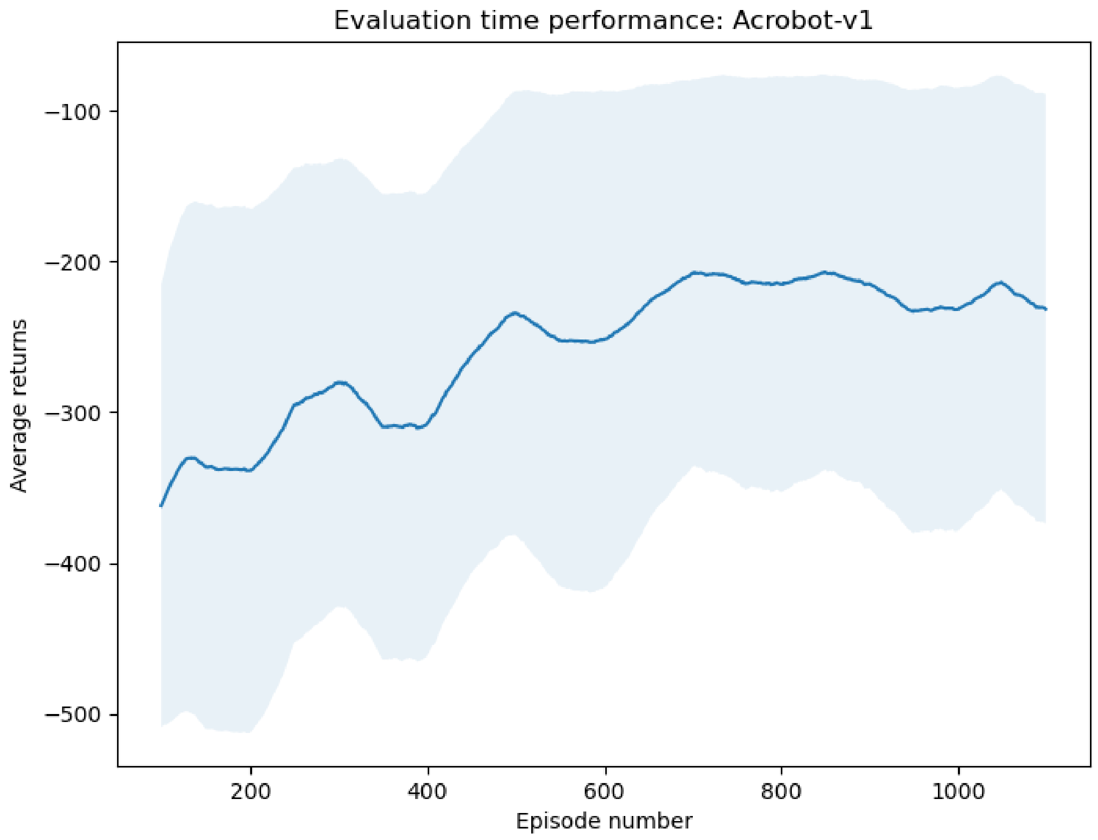

Figure 3: Average returns from 10 training epochs when the agent is evaluated every 10 episodes is plotted as the solid line with one standard deviation of the 10 trials shaded for the `Acrobot-v1` environment.

States in `MountainCar-v0` environment are 2×1 dimensional vectors and action space consists of 3 options. Due to the lower dimensions of the state vectors in `MountainCar-v0` it required a *deeper* neural network to extract the feature vectors. However, the task was solved in *each* of the 10 epochs. While the learning curve for `MountainCar-v0` might look superficially better than `Acrobot-v1`, because the standard deviations are smaller, the maximum number of steps allowed for `Acrobot-v1` is 500 while the same for `MountainCar-v0` is 200. The mean cumulative returns and standard deviation has to be compared with respect to this baseline. The trained LSTDQ agent accumulates rewards of ~ -160 ($\sim$ 40 (or 20%) better than the baseline -200) in `MountainCar-v0`, while it accumulates rewards of ~ -250 ($\sim$ 250 (or 50%) better than the baseline -500) in `Acrobot-v1`. A fairer comparison between the two environments would be when the maximum number of steps in both would be set to the same number, 500.

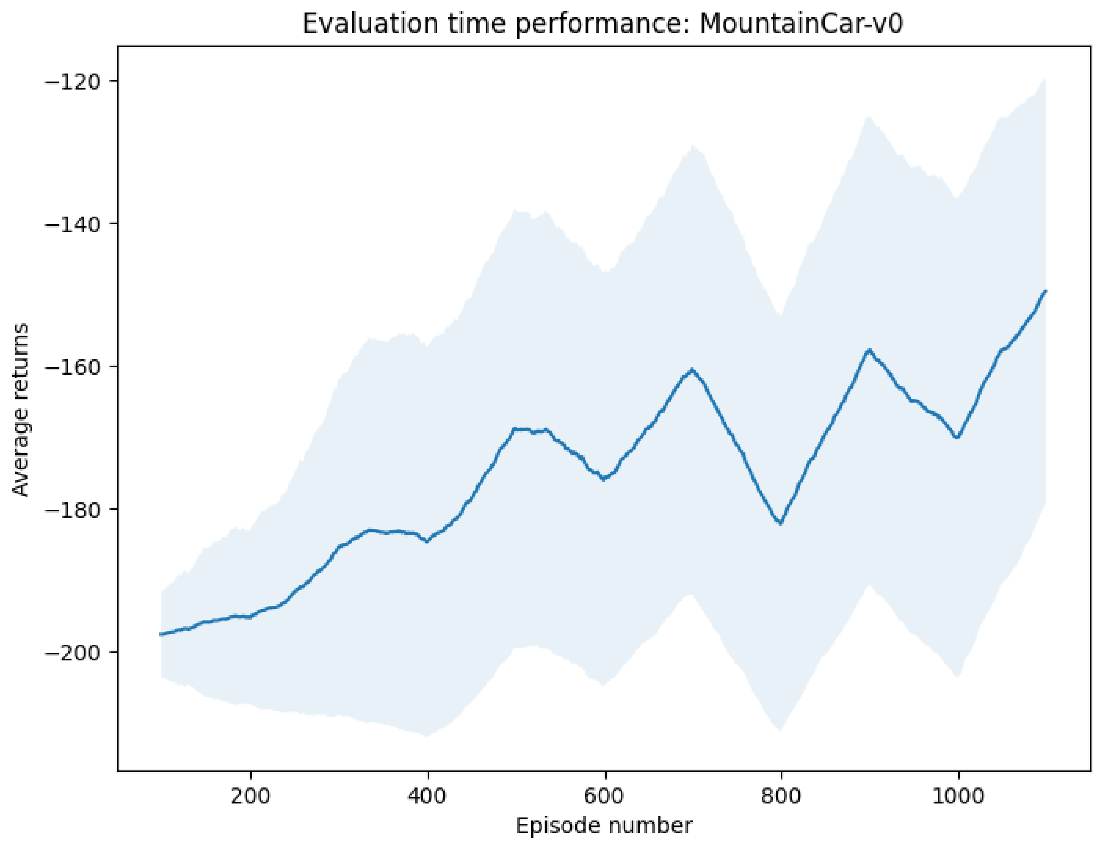

Figure 4: Average returns from 10 training epochs when the agent is evaluated every 10 episodes is plotted as the solid line with one standard deviation of the 10 trials shaded for the `MountainCar-v0` environment.

None of the episodes for the environment `Pendulum-v1` was able to solve the task.

The comparative algorithm stability decreased when moving from `Acrobot-v1` to `CartPole-v1` to `MountainCar-v0` to `Pendulum-v1` (continuous action space, discretising the action space to 3, 5, 7, or even 9 intervals did not solve the task in 1000 episodes). `Acrobot-v1` has a 6 dimensional state, from which it might be easier to extract features, hence LSTDQ shows a more stable performance in that environment.

The absolute stability of LSTDQ algorithm over multiple epochs of the same environment was also higher in `Acrobot-v1`. The standard deviation about the mean cumulative rewards is lower as seen in Fig 3, which indicates larger stability.

Increasing the number of nodes in each layer and increasing the number of layers in the autoencoder improves the performance of the algorithm up to a limit (observed to improve upto (16, 32, 16) nodes in the hidden layers), after which the autoencoder overfits to minimize the reconstruction error instead of extracting meaningful features from the state. 16 dimensional feature vectors were created from the input state in each environment.

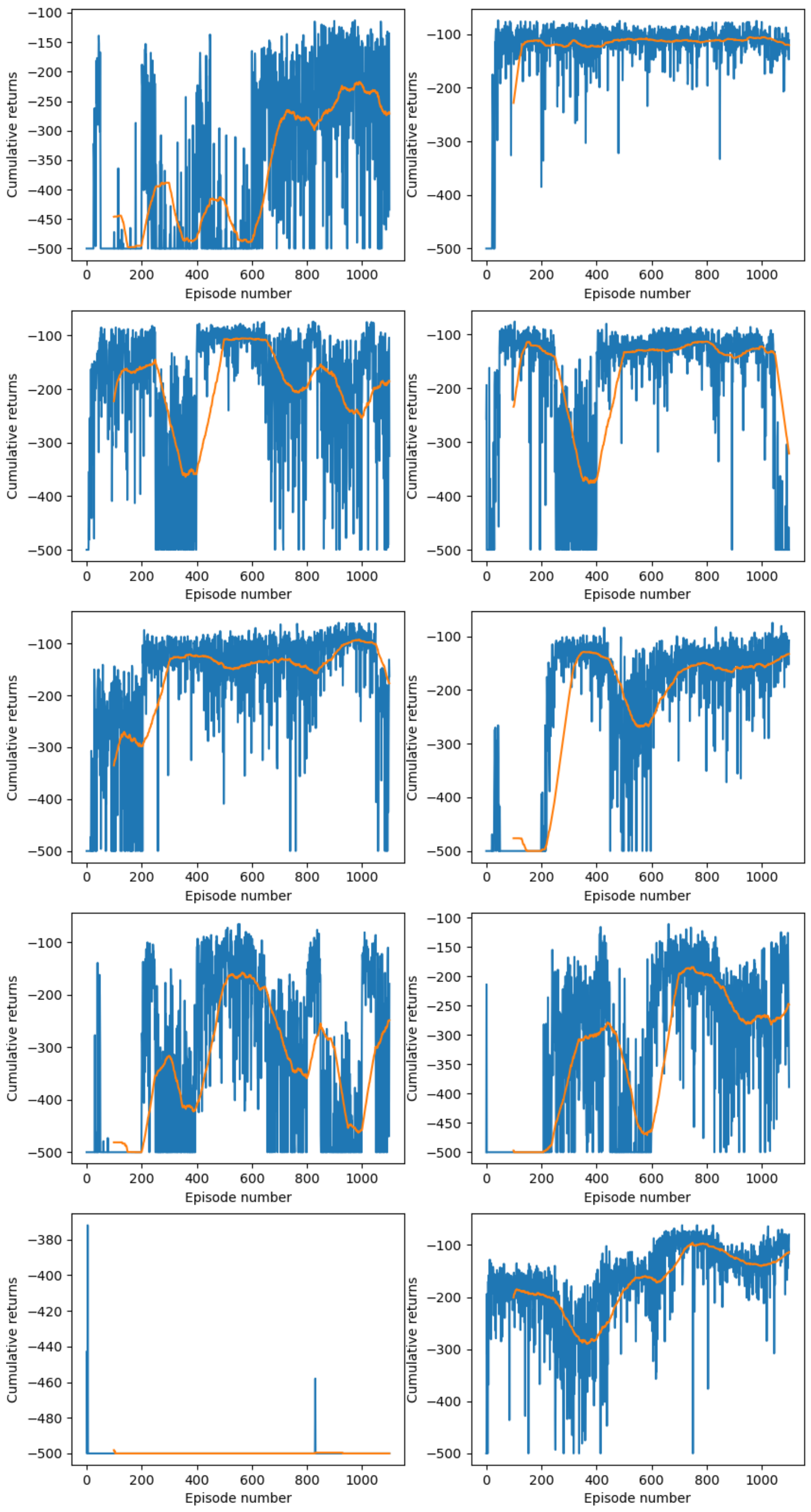

Figure 5: Returns from 10 individual training epochs when the agent is evaluated every 10 episodes is plotted as the blue line, and the moving average over 100 values as the orange line for the `Acrobot-v1` environment.

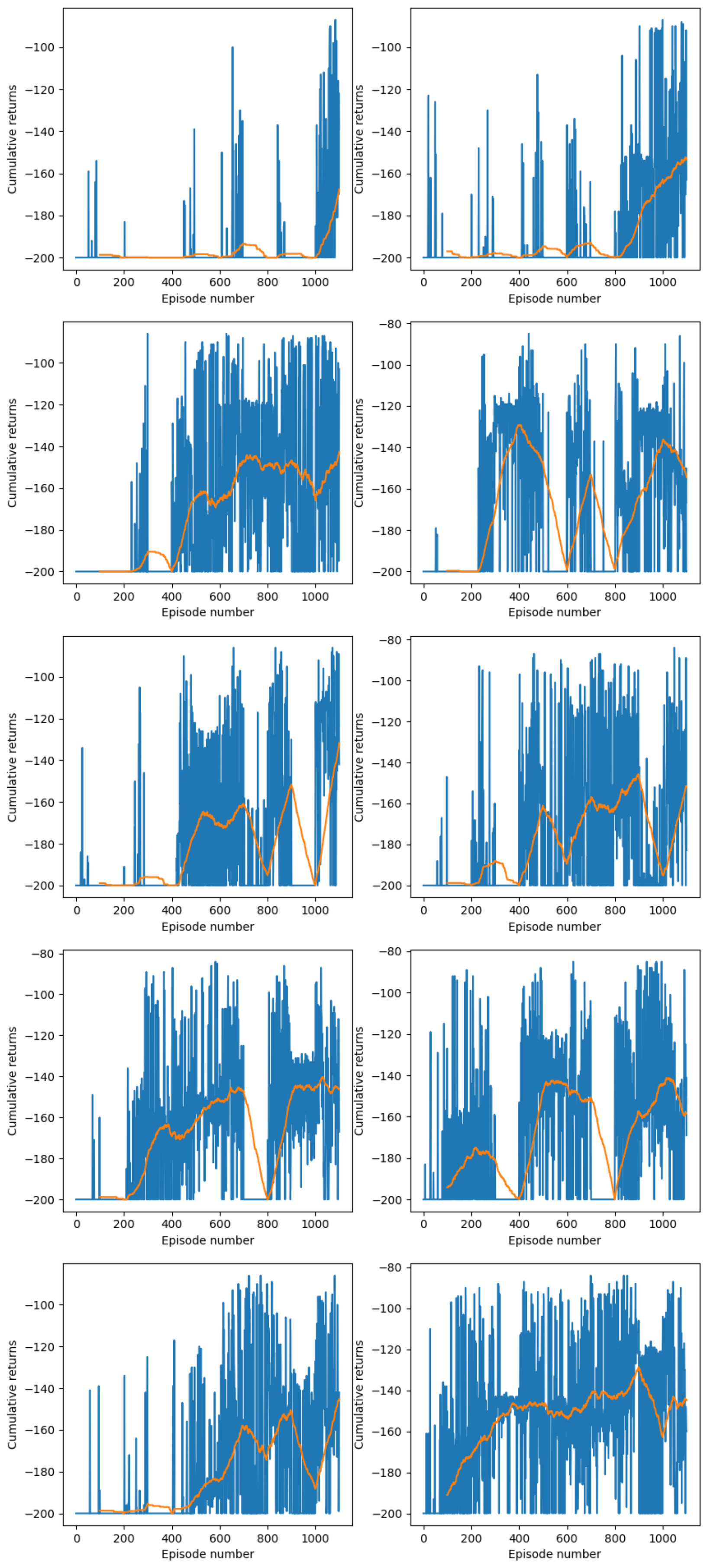

Figure 6: Returns from 10 individual training epochs when the agent is evaluated every 10 episodes is plotted as the blue line, and the moving average over 100 values as the orange line for the `MountainCar-v0` environment.