CIS 520, Machine Learning, Fall 2020
Homework 8
Due: Monday, December 7th, 11:59pm
Submit to Gradescope

Your name here

1 Reinforcement Learning

1. Formulate an MDP for the `Chain` environment by specifying each component of the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$. For the components p and r, write down $p(s'|s,a)$ and $r(s,a,s')$ for each setting of s, a, s' for which the probability/reward is non-zero:

   The MDP formulation is as follows:

   (a) The set of states $\mathcal{S} = \{0, 1, 2, 3, 4\}$

   (b) The set of actions $\mathcal{A} = \{f, b\}$

   (c)

$$p(s_{next} \mid s, a = forward) = \begin{cases} 0.9, & \text{if } n \neq 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$p(s_{next} \mid s, a = backward) = \begin{cases} 0.9, & \text{if } n = 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$r(s, a, s_{next}) = \begin{cases} 2, & \text{if } s_{next} = 0 \\ 0, & s_{next} = s + 1 \\ 10, & \text{otherwise} \end{cases}$$

2. Find the optimal state-value function V^*, i.e. find the optimal value $V^*(s)$ for each state s:

   $V^* = \begin{bmatrix} 39.66450035 & 44.45932227 & 50.36422227 & 57.65422227 & 66.65422227 \end{bmatrix}$

3. Using your solution to the second part above, find an optimal deterministic policy π^*, i.e. find an optimal action $\pi^*(s)$ for each state s:

   $Q^* = \begin{bmatrix} 39.66450035 & 44.45932227 & 50.36422227 & 57.65422227 & 66.65422227 \\ 37.91786176 & 38.44930276 & 39.10540276 & 39.91540276 & 40.92712529 \end{bmatrix}$

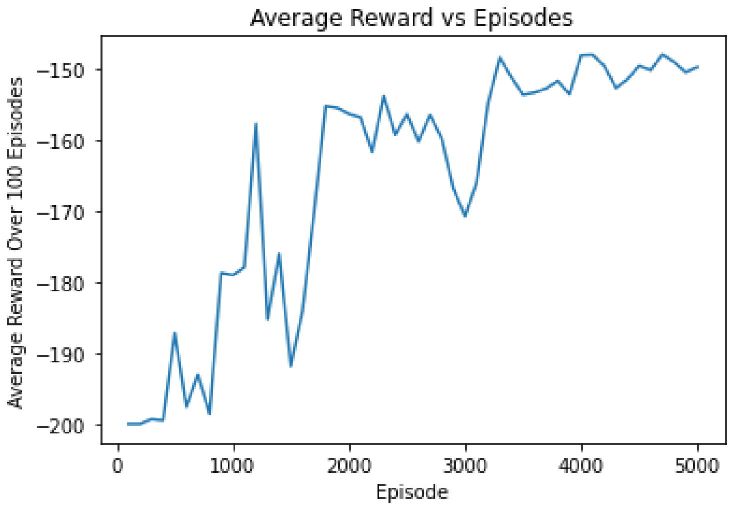
   The optimal policy is: $\pi^* = \begin{bmatrix} \{f, f, f, f, f\} \end{bmatrix}$

2 Reinforcement Learning: Q-Learning

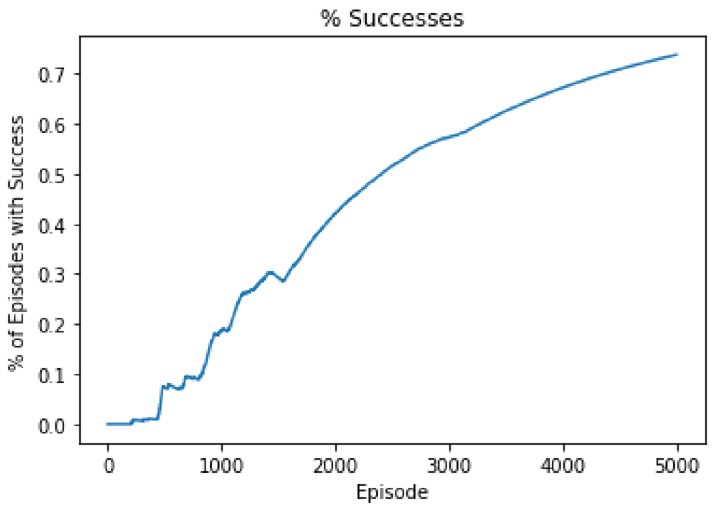
2.1 "Traditional" Q-Learning

After running Q-learning using the parameters provided in the notebook, use the code provided to plot the **Average Reward, Success Rate, and Car Final Position vs Episodes**. Also give an image of your car reaching the flag in the final episode.

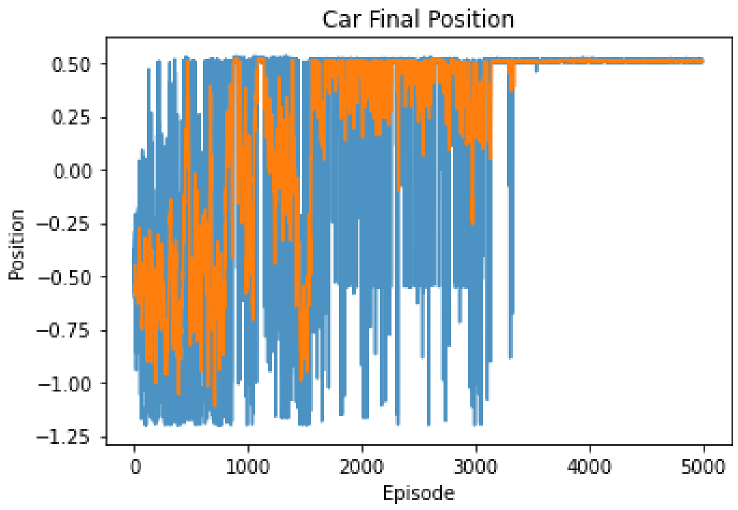
avg reward



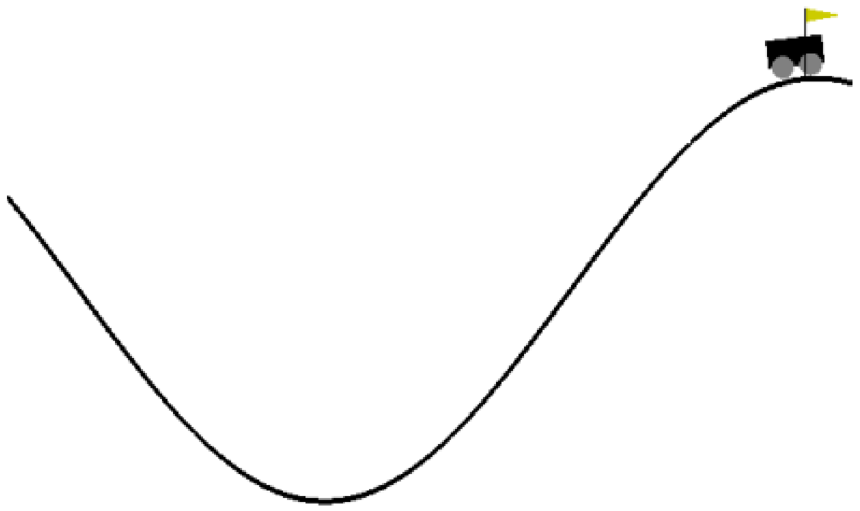
success rate



car final position



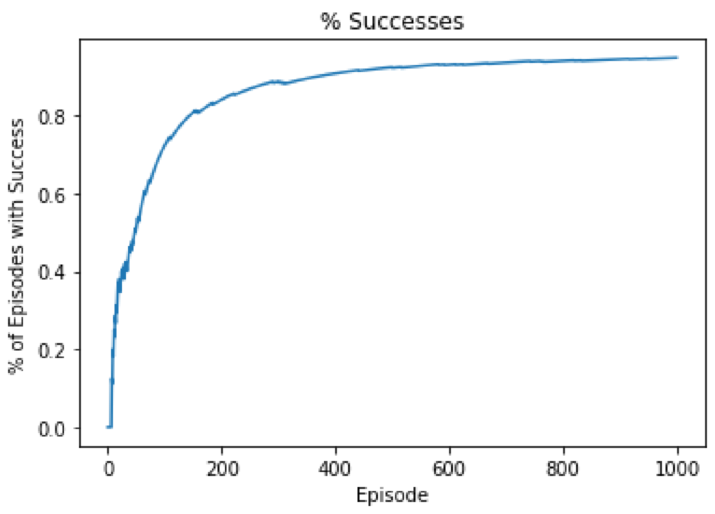
car success image



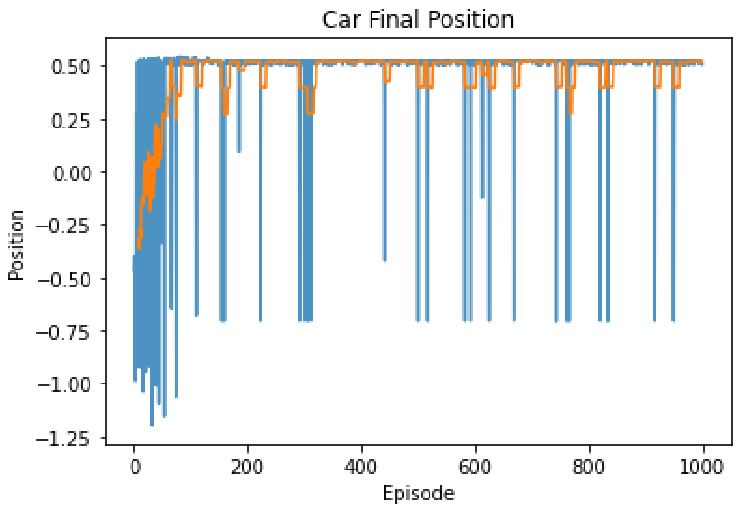
2.2 Deep Q-Learning

After running deep Q-learning using the parameters provided in the notebook, plot the **Success Rate, and Car Final Position vs Episodes**.

success rate, deep



car position, deep



2.3 Questions

1. As part of the Deep QLearning implementation, you implemented a reward_shaping function to aid in the learning process. Compare this to the original reward structure in part 2.1 – why do you think this modification of the reward is helpful?

   The original reward mechanism may take too long to converge, so with modified reward, the DQN basically learnt faster of the going upward in this task.

2. Compare the **Success Rate** and **Car Final Position** plots between your two implementations. Which algorithm is learning a successful policy more quickly? Briefly comment on potential reasons for any differences in performance.

   DQN learnt faster because of the modified reward.

   For the traditional Q-learning, it essentially relies on the stochastic process to succeed the task. So the first half part is much more fluctuating as for the exploration.

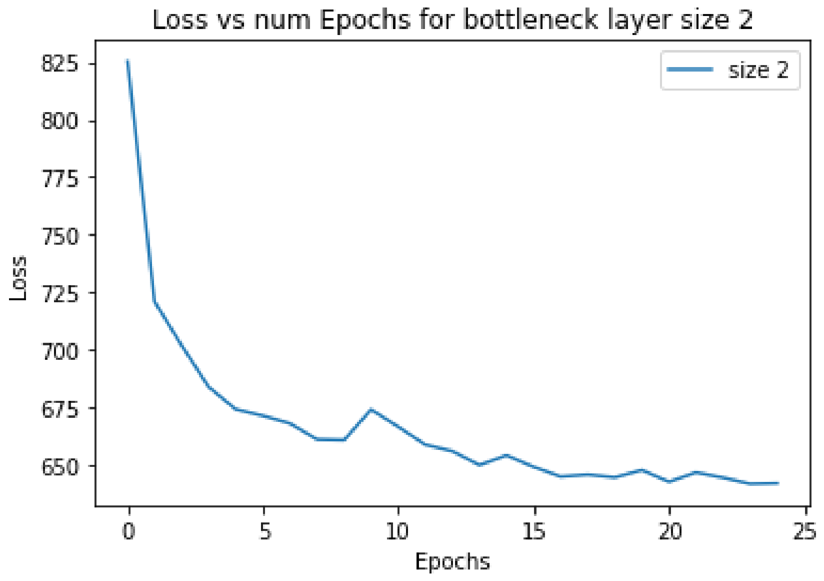
   For the DQN, it learns faster due to the modified reward. The reward explicitly embeds the go-up behavior for the model. So it converges faster.

3 Autoencoders

3.1 Part 1: Constructing the Autoencoder

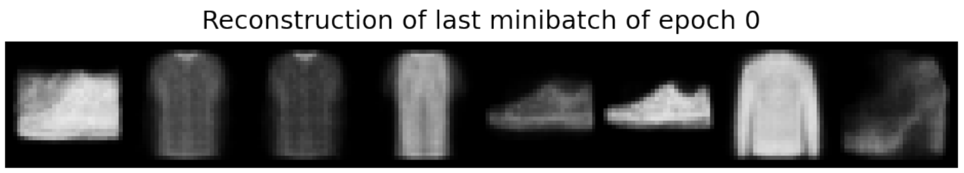
The training curve is given below:

. . . Training curve

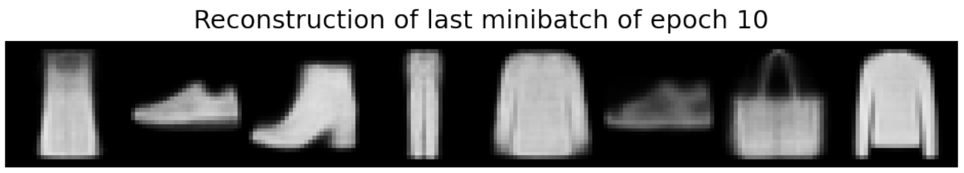


The reconstructions of the last minibatch for epochs 0, 10, and 20 are:

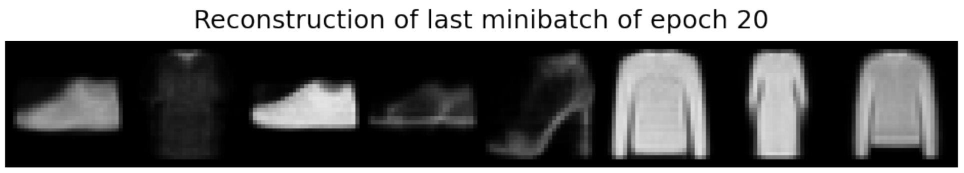
. . . Epoch 0



Epoch 10

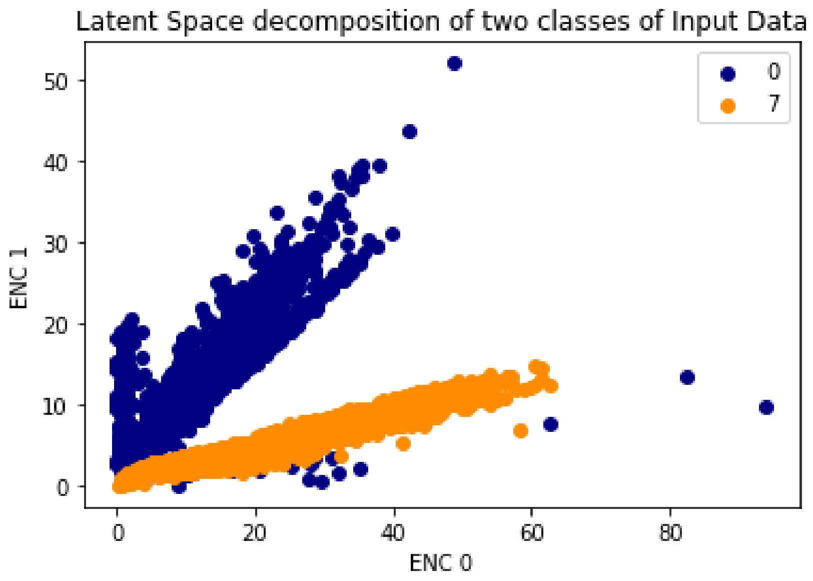


Epoch 20



3.2 Part 2: Latent Space Decomposition

1. Plot of the latent space of this autoencoder: latent space plot



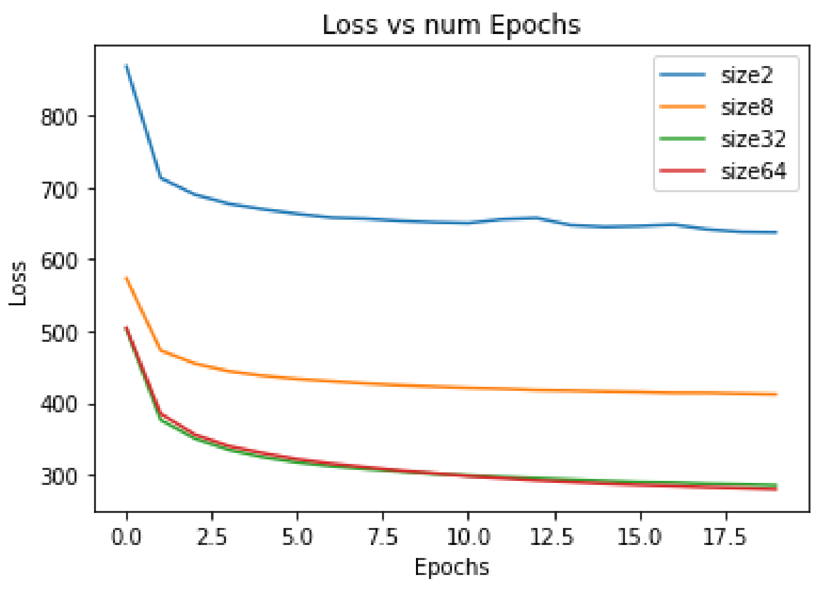
2. From the latent space plot, explain what the encoding has done to the inputs. How is this effect related to what PCA does? Why is this useful?

   The encoder basically map the images to the 2D latent space, while what the pca does is to map the images into the space that have same number of dimensions and sort them by the information contained in each dimension. Unlike PCA, the encoder is able to learn the similarities and differences between images and extract them as a few numbers, while the PCA can only condense the information without any filtering.

3.3 Part 3: Reconstruction Error vs Bottleneck Layer

1. Report the combined training curves (mean epoch loss vs epochs) for all the bottleneck layer sizes $[2, 8, 32, 64]$. Also report the reconstructed images for Epoch 20 for each of the configurations.

   The training curves of all the configurations: training curve plot



The reconstructions for sizes $[2, 8, 32, 64]$ for epoch 20 of the last minibatch:

bottleneck 2

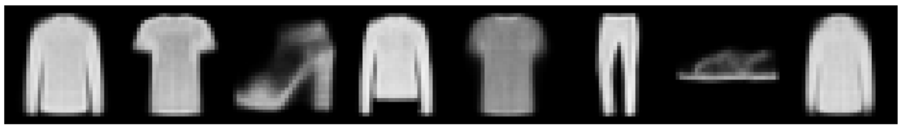


Figure 1: Size 2

bottleneck 8

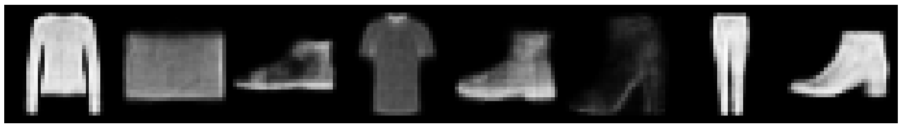


Figure 2: Size 8

bottleneck 32

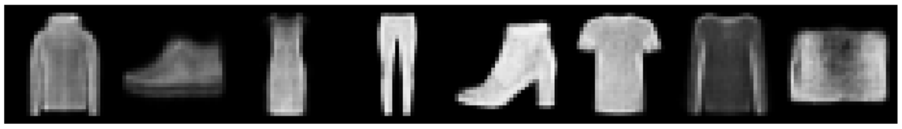


Figure 3: Size 32

bottleneck 64

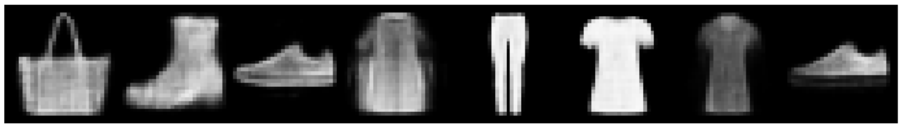
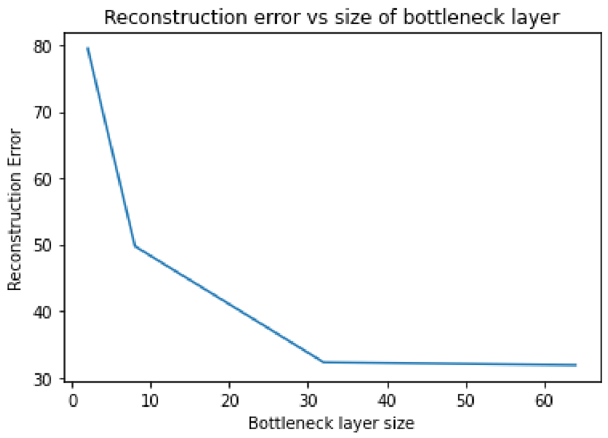


Figure 4: Size 64

2. The reconstruction error for images of class 0 as the bottleneck layer size increases:

   reconstruction vs size



   What are you observing? How does what you see relate to PCA? What does this tell you about how you can potentially work with a high-dimensional data-space?

   The loss as well as the error reduces as the bottleneck layer size goes up, just as what we observed when we increase the number of dimensions reduced by PCA. However, in the PCA case, the reconstruction error reduced when increasing the dimension decreased greatly as the added dimensions are less important, while the reduced error in the encoder case drops slower. For high-dimensional data-space, mapping it to a comparatively lower dimensional latent space via encoder might help us to reduce the workload while preserving most of the information we need.