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

Yuezhong Chen; Sheil Sarda

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) In the states of 0 1 2 or 3, we have:

   $p(0|i,f) = 0.1, p(0|i,b) = 0.9$ and $p(i+1|i,f) = 0.9, p(i+1|i,b) = 0.1$
   $r(i,f,0) = 2, r(i,f,i+1) = 0$ and $r(i,b,0) = 2, r(i,b,i+1) = 0$

   When the state is 4, we have:

   $p(0|4,f) = 0.1, p(0|4,b) = 0.9$ and $p(4|4,f) = 0.9, p(4|4,b) = 0.1$
   $r(4,f,0) = 2, r(4,f,4) = 10$ and $r(4,b,0) = 2, r(4,b,4) = 10$

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

   $V^* = \begin{bmatrix} 40.742 & 45.525 & 51.430 & 58.720 & 67.720 \end{bmatrix}$

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:

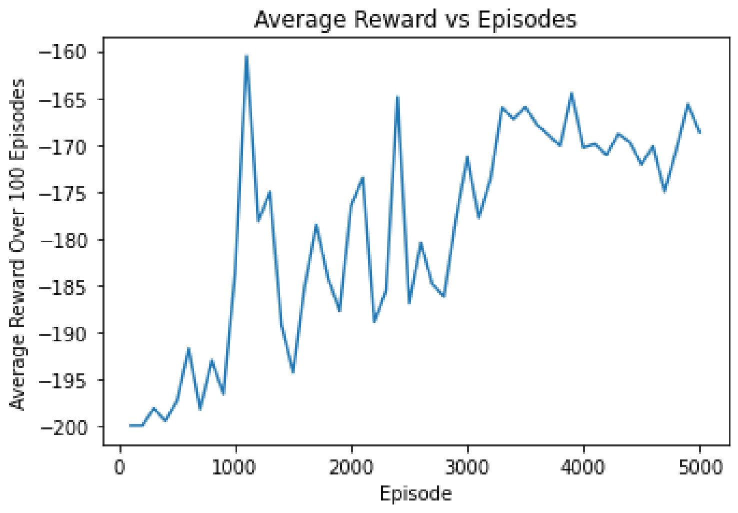
$$Q^* = \begin{bmatrix} 40.742 & 38.898 \\ 45.525 & 39.430 \\ 51.430 & 40.086 \\ 58.720 & 40.896 \\ 67.720 & 41.896 \end{bmatrix}$$

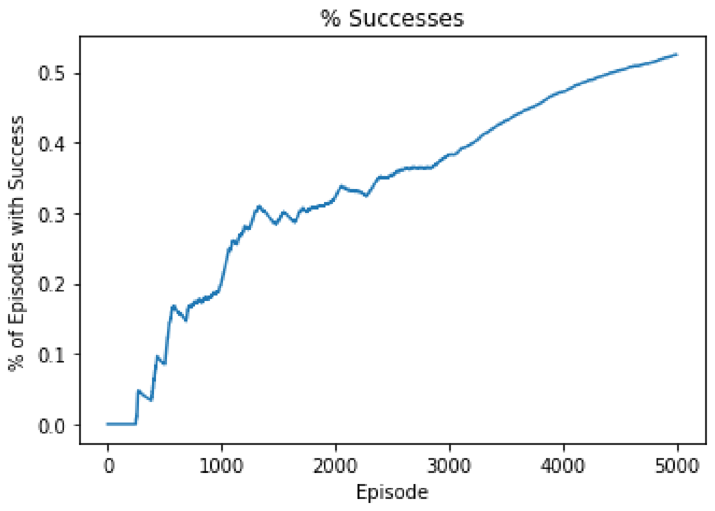
   The optimal policy is: $\pi^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

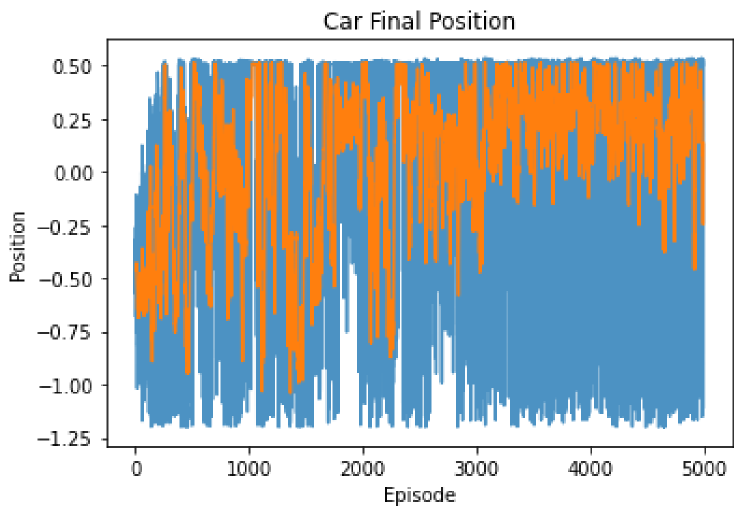
2 Reinforcement Learning: Q-Learning

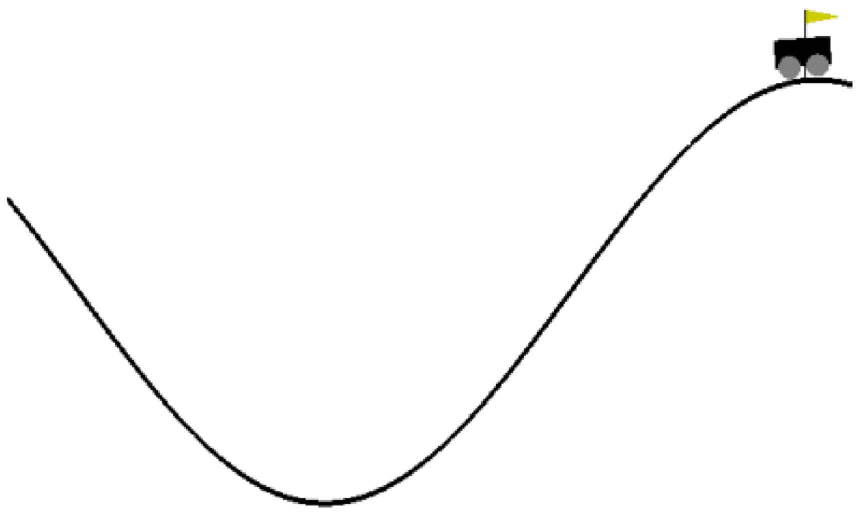
2.1 "Traditional" Q-Learning

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.



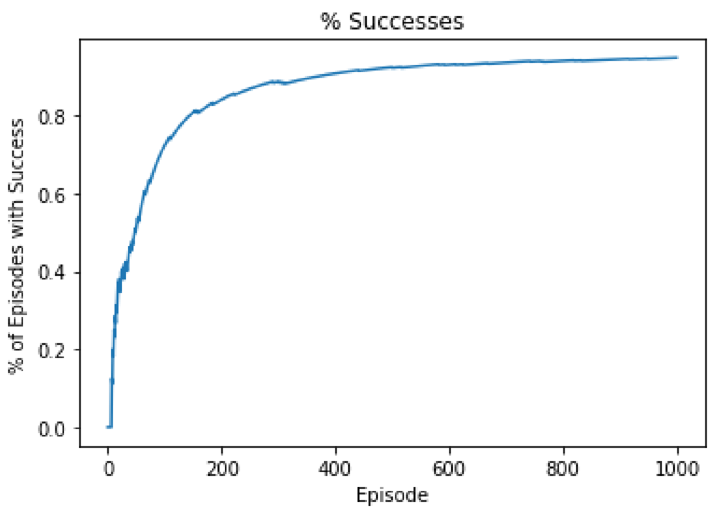


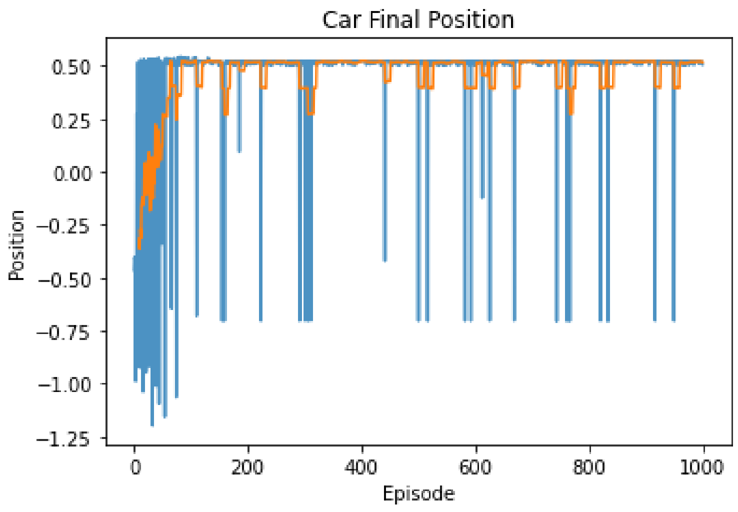


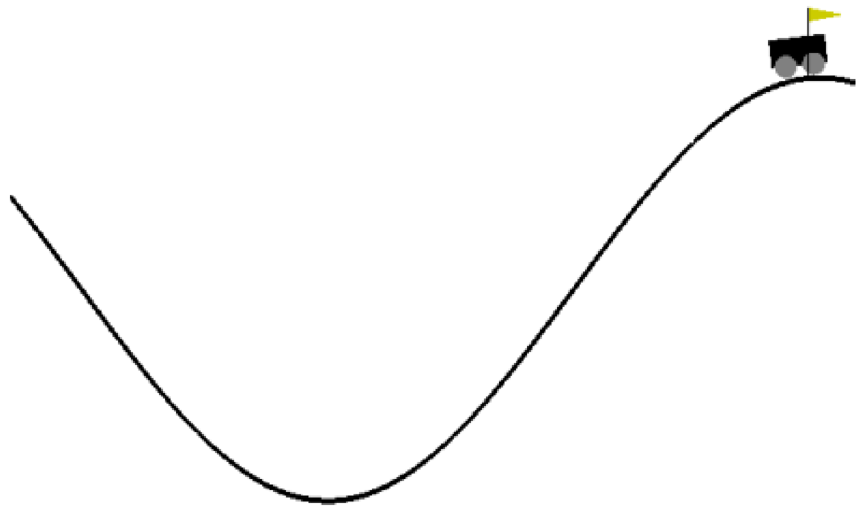


2.2 Deep Q-Learning

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


% Successes
% of Episodes with Success
Episode
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000


Car Final Position
Position
Episode
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
0
200
400
600
800
1000



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?
   Description of the reward shaping function: the reward is always 0.5 + the position of the car in the next state, and if the position of the car in the next state is greater than 05, add +1 on top of the computed reward. The original reward structure in 2.1 is un-shaped.

   In general, Reward shaping augments the natural reward signal in order to provide more frequent feedback on appropriate behaviors during the learning process. This modification of the reward is helpful because it gives the model more feedback during the training process which guides it towards successful outcomes earlier.

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.

   Differences between the Deep Q-Learning and Standard Q-Learning models:

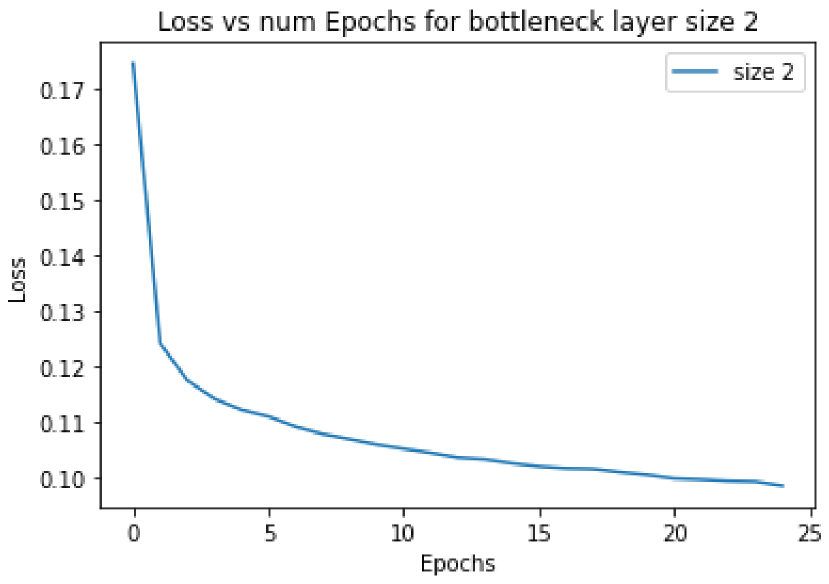
   (a) Car reaches the final position (0.5) within 1000 iterations in the Deep Q-Learning model, whereas the standard Q-Learning model takes 5000 iterations.
   (b) Success rate of the Deep Q-Learning model approaches 0.8 as an asymptote within the first 200 episodes, and then steadily makes progress towards a 100% success rate.
   (c) The Standard Q-Learning model follows a more linear trajectory in its success rate, and is only able to achieve a 50% success rate at the end of 5000 iterations.

   Based on the above observations, we can conclude that the Deep Q-Learning model learns a successful policy quicker. The potential reasons for these performance differences include: (a) presence of a reward shaping function which provides the model with more feedback during the training process; (b) increased model complexity to better capture the complexity of the optimal policy.

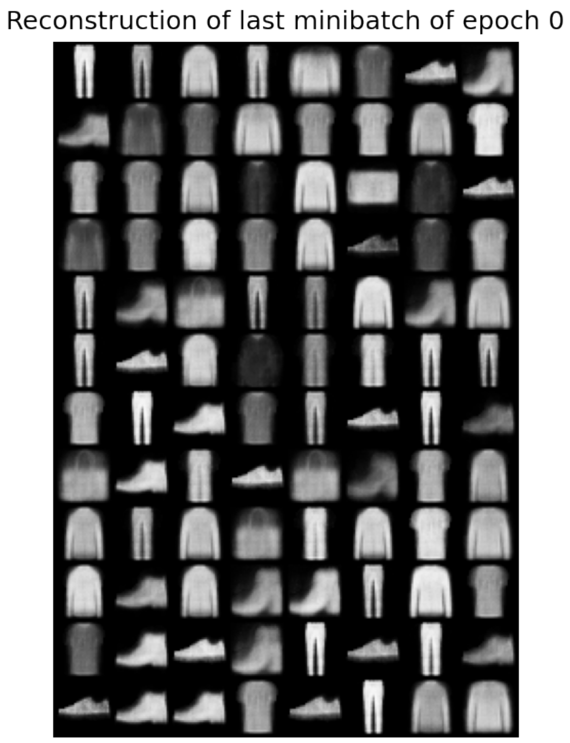
3 Autoencoders

3.1 Part 1: Constructing the Autoencoder

The training curve is given below:

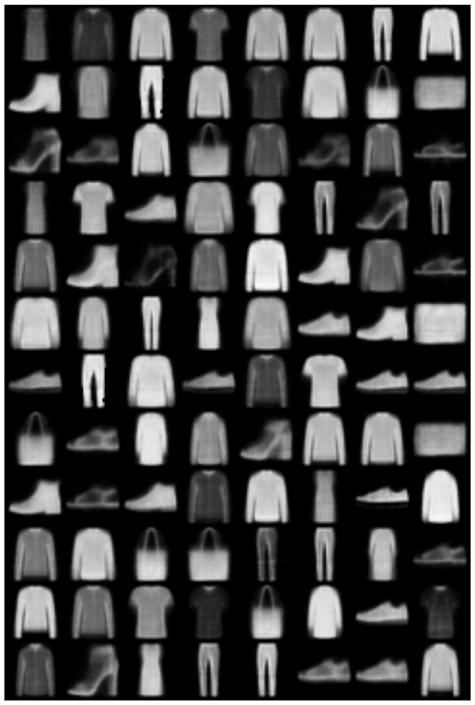


The reconstructions of the last minibatch for epochs 0, 10, and 20 are:



Reconstruction of last minibatch of epoch 10

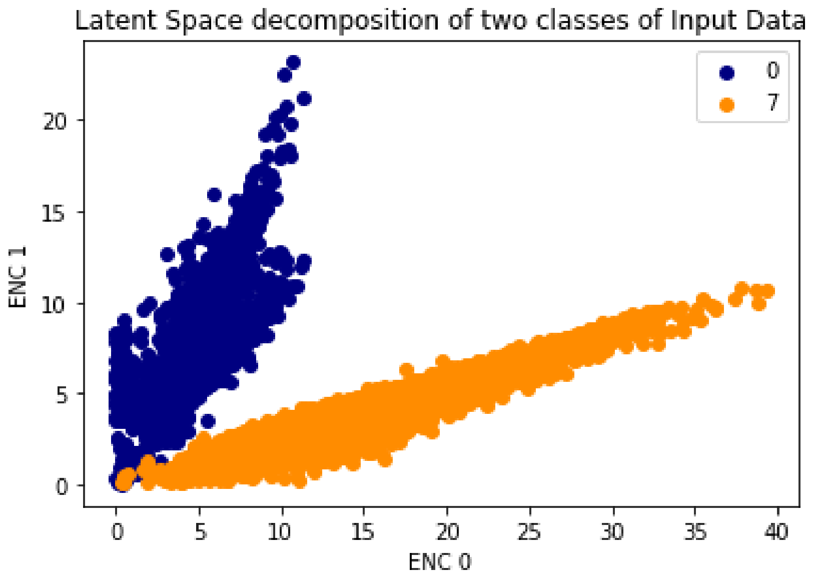
Reconstruction of last minibatch of epoch 20



3.2 Part 2: Latent Space Decomposition

1. Plot of the latent space of this autoencoder:

Changed random seed to 40 to get a better latent space plot since the original one was a vertical line.



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?

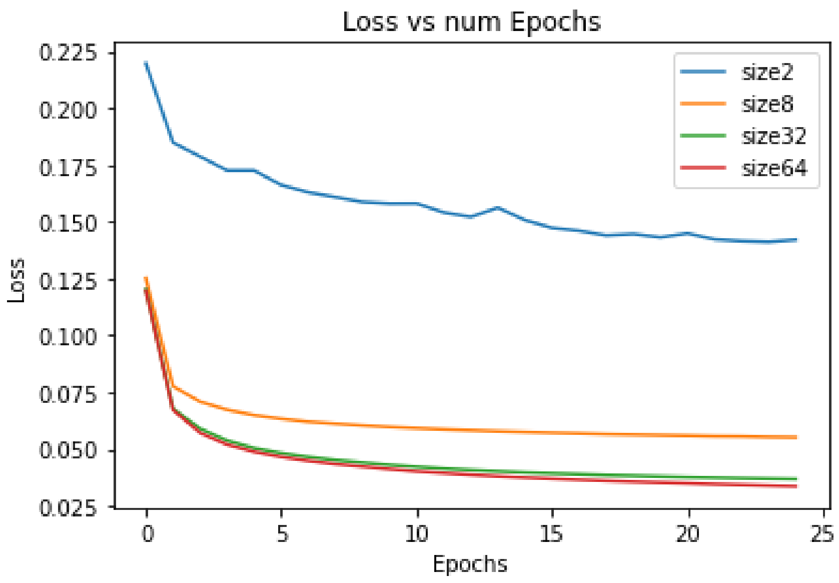
   The latent space is where encoded vectors lie. The encoder decreases the dimensionality of the input up to the latent space. PCA is looking for the best linear subspace of the initial space described by an orthogonal basis of new features such that the error of approximating the data by their projections on this subspace is as small as possible.

   PCA and the Encoder are similar in that both are building new features using the features of the input dataset. The capability to compress data can be used for a variety of tasks such as removing noise / occlusions from images, developing lossy compression algorithms and machine translation.

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:



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

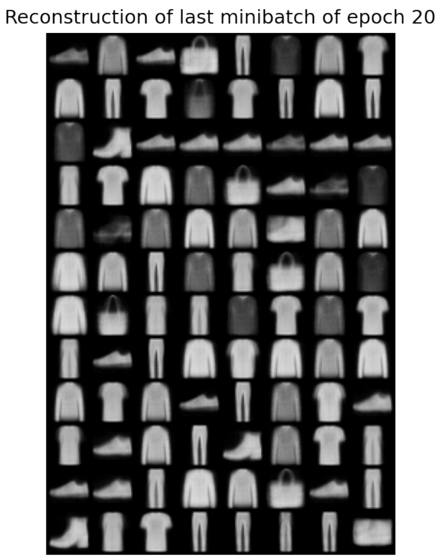


Figure 1: Size 2

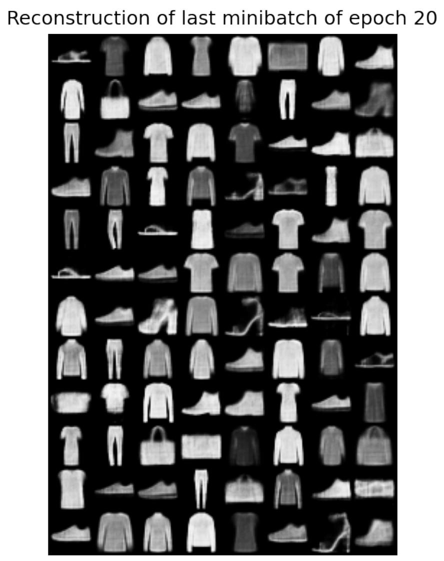


Figure 2: Size 8

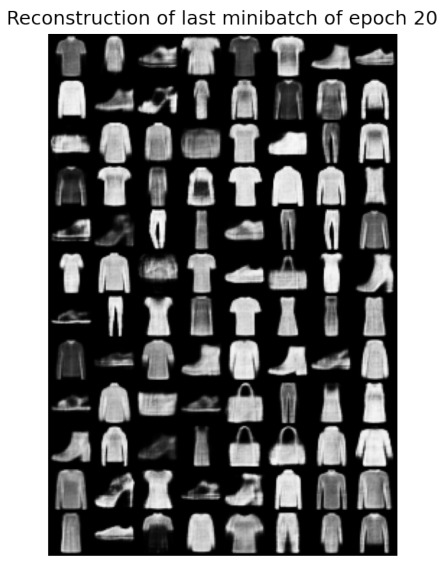


Figure 3: Size 32

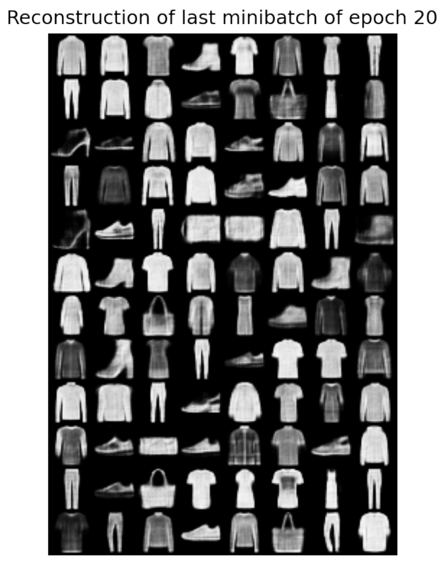
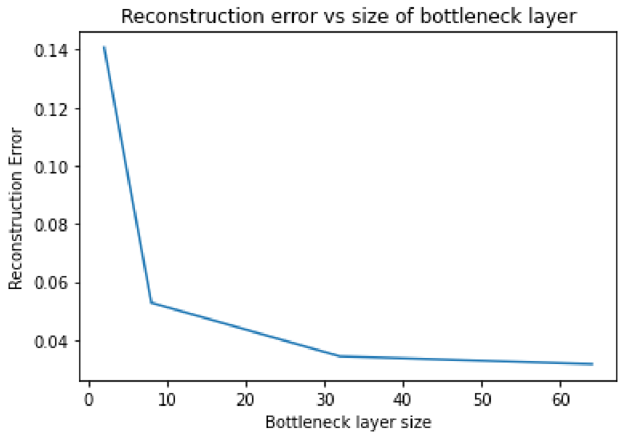


Figure 4: Size 64

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



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?

Observations based on how reconstruction error changes with bottleneck layer size, and similarities to PCA:

(a) Inflection point at the bottleneck layer size of 32, indicating that 32 is the least number of dimensions required for optimally encoding the dataset.

(b) Similar to how the PCA analysis helps determine the top k features which account for most of the variance seen in the data, this analysis tells us that 32 features explain most of the variance in this dataset.

Solutions for dealing with high-dimensional data spaces: (a) PCA to generate new features which are linear combinations of input features; (b) In a Vanilla Autoencoder, the latent space forms a bottleneck which forces the model to learn an effective compression of the data.