2.1: I found that adjusting to the learning rate of 0.02, the model would give NaNs. This could be that the gradients of the model is growing too quickly. When the learning rate was set at 0.0002, the loss never changed and stayed around 2.100s. These events could be a result of vanishing gradient descent. At 0.002, the model gave better and more consistent results with around 30-40% accuracy, but the loss still jumped around a lot from 0.5s to 1.5s.

2.2: With the learning rate of 0.002. The hidden dimension size of 10 resulted in terrible losses and was very inaccurate, most likely caused by less complexity with the benefit of faster training time and saving computational costs. At hidden size of 100, the model produced better accuracy with the loss dipping around 0.8, 0.9 at later epochs while starting off with 2.0 at the earlier epochs. It seems adding a little bit of complexity helped the model learn better.

2.3: At 100000 epoch, the model was very inconsistent with losses and eventually hit NaNs. This could be a result of overtraining or training for too long. For 1000 epochs, the training was most likely too short and the model was very inaccurate. I believe 50000 epochs is the best choice for getting the most consistent results.

2.4: I decided to use the ADAM optimizer to test on the same parameters and it resulted in decent outcomes. For each epoch. However, the accuracy was only 0.35 and epochs 20000 to 25000 were very inconsistent which indicates a vanishing gradient descent. I then used cross-entropy loss function with Adagrad optimizer. I noticed my model kept overfitting with huge loss values for each epoch. I tested multiple hyperparameters and ended up with hidden size of 100, 50000 epoch, a learning rate of 0.0002. I was unable to get over 40% until i learned how to incrementally decrease learning rate as the model was learning.