Assignment 1 Writeup

DO NOT TAG

Name: Bing Yang
GT Email: byang322@gatech.edu

Two-Layer Neural Network

DO NOT TAG

1. Learning Rates

Tune the learning rate of the model with all other default hyper-parameters fixed. Fill in the table below:

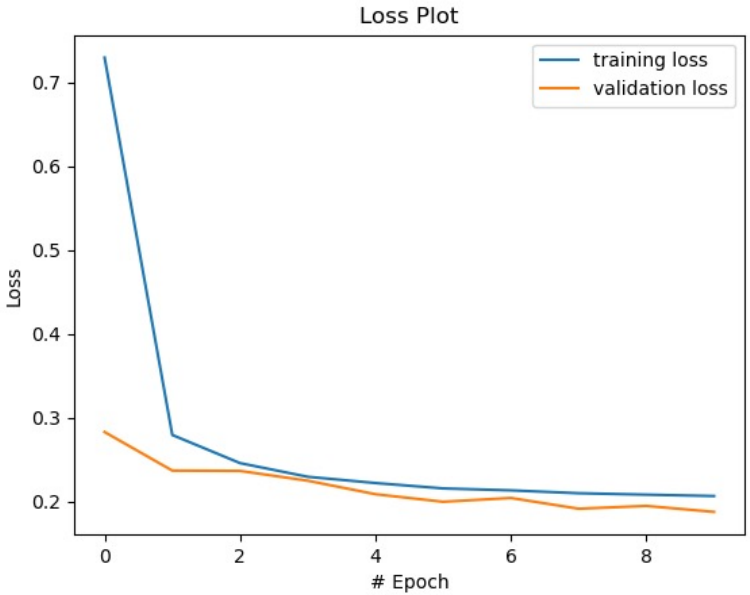
	lr=1	lr=1e-1	lr=5e-2	lr=1e-2
Training Accuracy	0.9438	0.9244	0.9088	0.7303
Test Accuracy	0.9506	0.9213	0.9137	0.7545

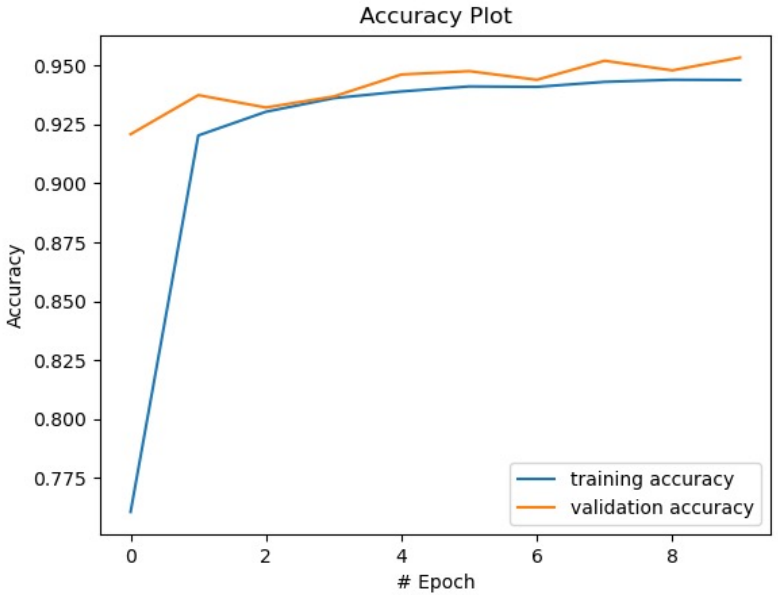
1. Learning Curve

Plot the learning curves using the learning rates from the previous slide and put them below (you may add additional slides if needed).

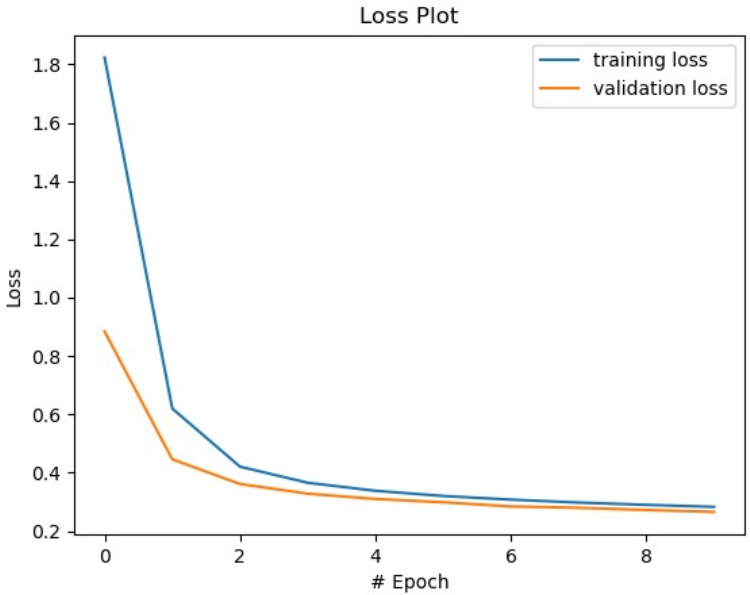
Please see all slides below. Each of the following 4 slides show the results for one learning rate.

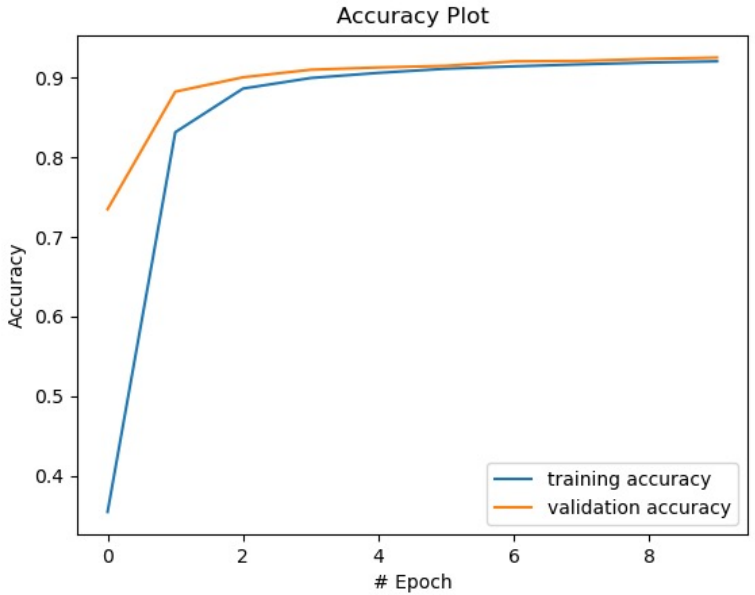
Learning rate = 1.0



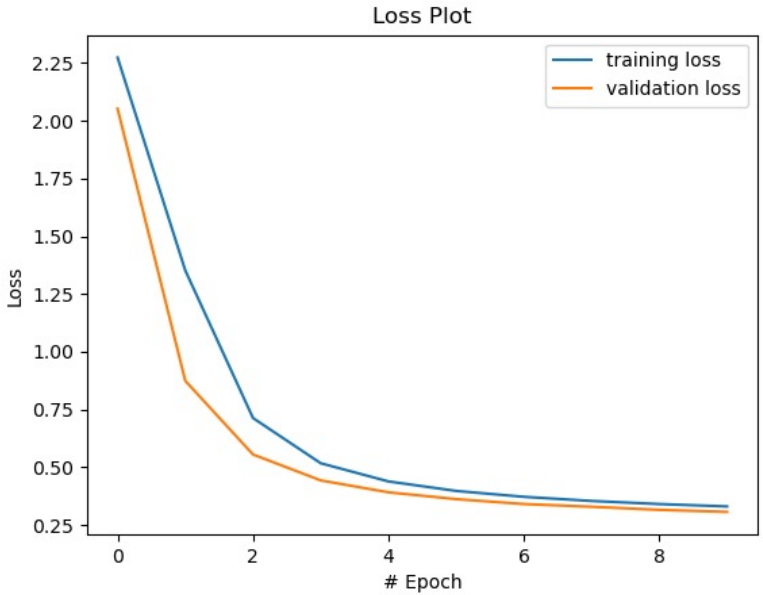


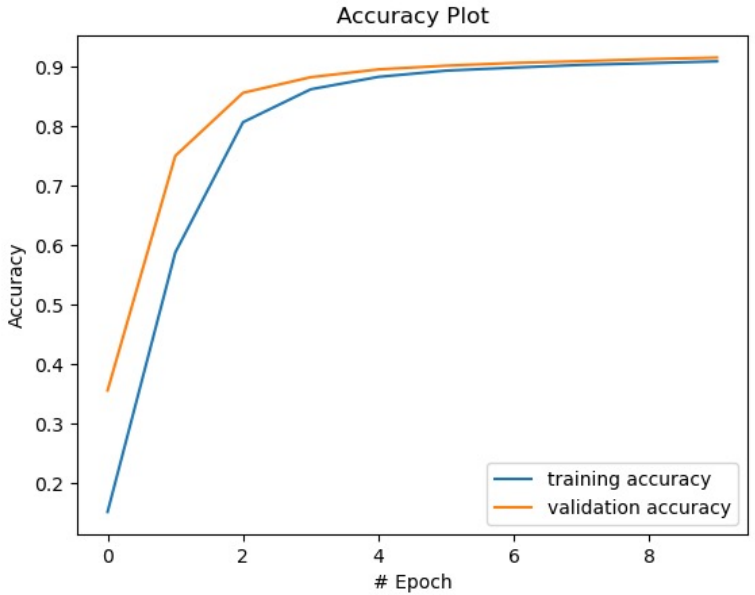
Learning rate = 0.1 (default mode)



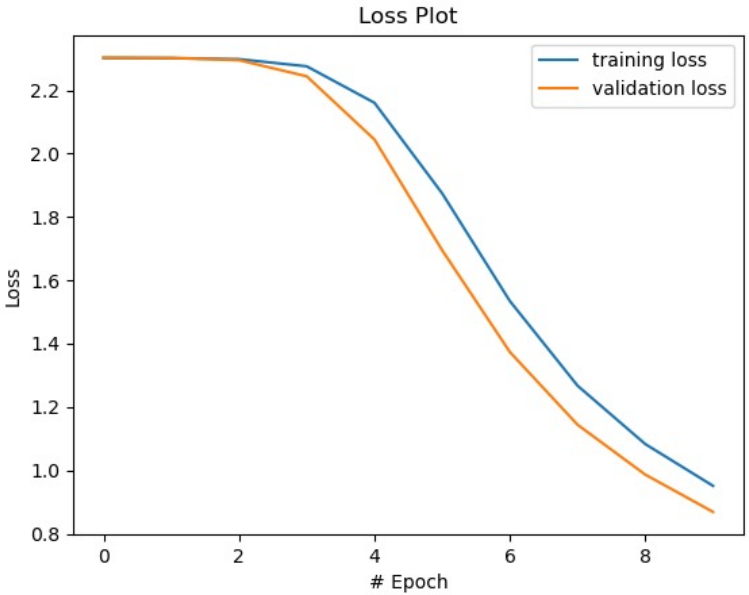


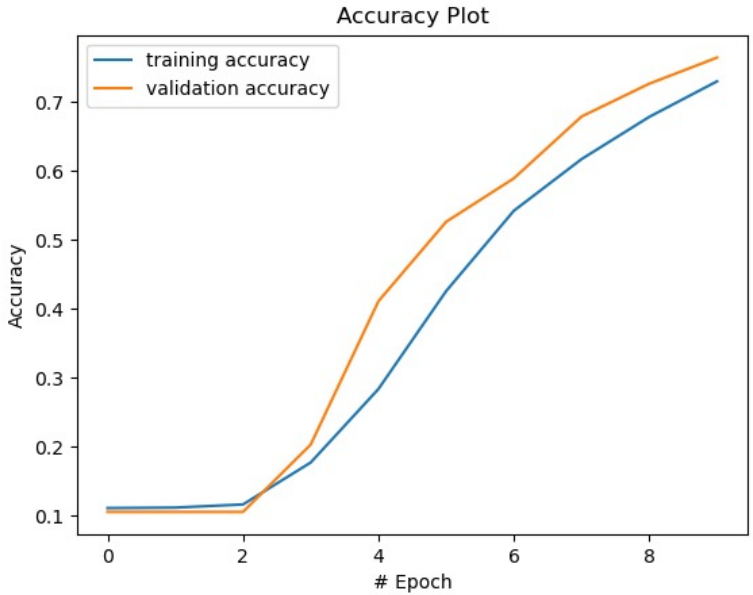
Learning rate = 0.05





Learning rate = 0.01





1. Learning Rates

Describe and Explain your findings: *Explanation should go into **WHY** things work the way they do in the context of Machine Learning theory/intuition, along with justification for your experimentation methodology. **DO NOT** just describe the results, for example, you should explain why the learning rate has the observed effect. If you need more than one slide to answer the question, you are free to create new slides.*

My findings: prediction accuracy drops with smaller learning rate.

Explanation: There are two possible explanations for the poor performance when learning rate is small. First, it is possible that learning stuck at some local minimum because the learning rate is small. This might be the case for learning rate equals 0.05. Second, learning has not reached convergence since the learning rate is too small. This is clearly the case for learning rate 0.01, as we can see from the loss figure that there is no clear plateau at the end of even 10^{th} epoch.

2. Regularization

Tune the regularization coefficient of the model with all other default hyper-parameters fixed. Fill in the table below:

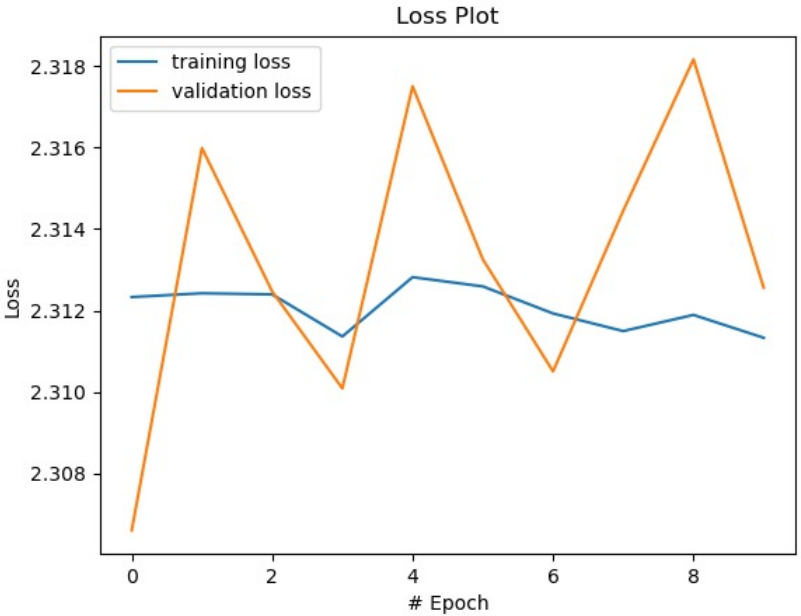
	alpha=1	alpha=1e-1	alpha=1e-2	alpha=1e-3	alpha=1e-4
Training Accuracy	0.1055	0.3390	0.8841	0.9244	0.9301
Validation Accuracy	0.0956	0.3601	0.8941	0.9251	0.9353
Test Accuracy	0.1135	0.3918	0.8907	0.9213	0.9329

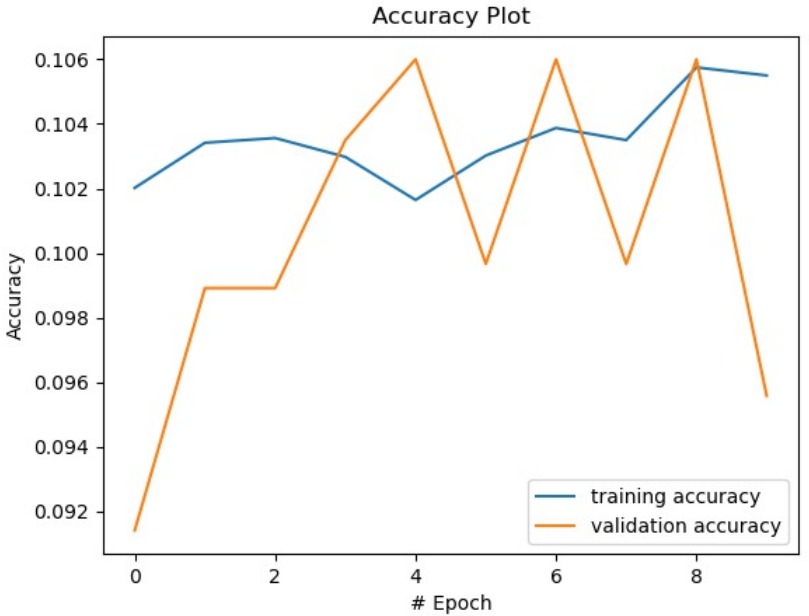
2. Regularization

Plot the learning curves using the regularization coefficients from the previous slide and put them below (you may add additional slides if needed).

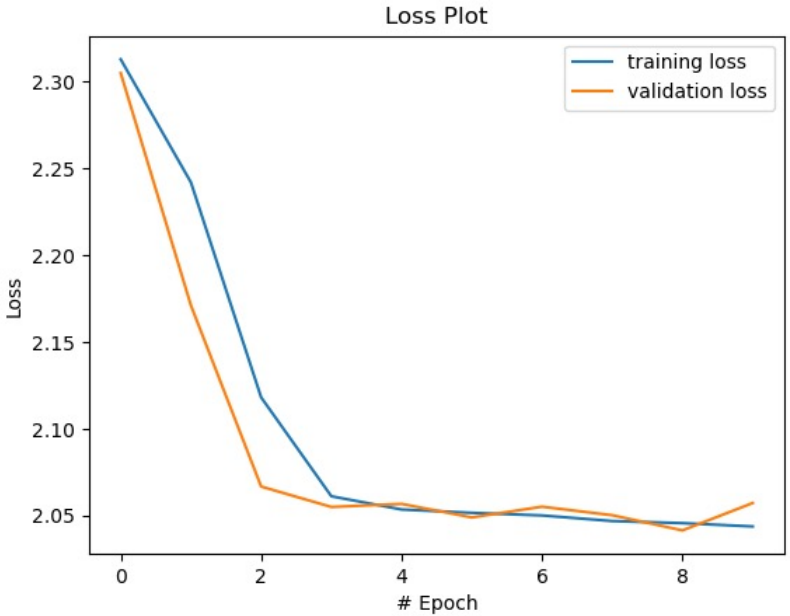
Please see all slides below. Each of the following 5 slides show the results for one regularization parameter.

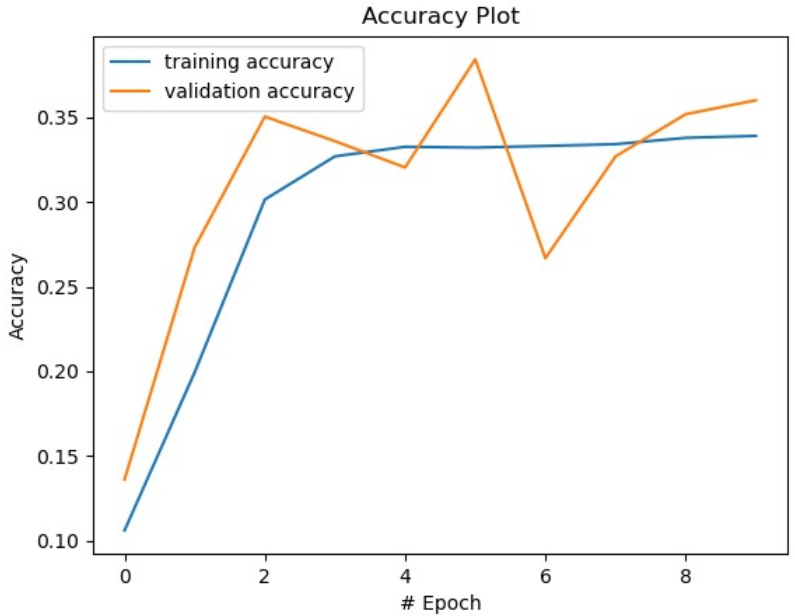
Regularization = 1.0



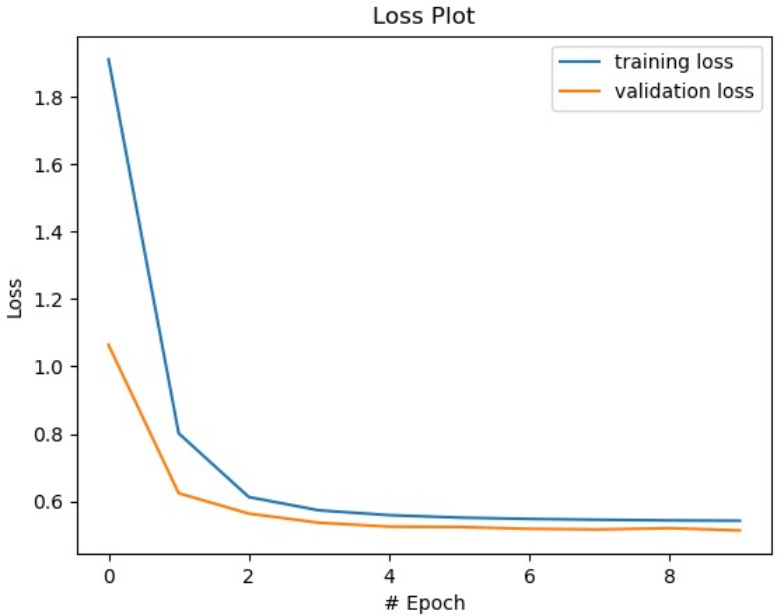


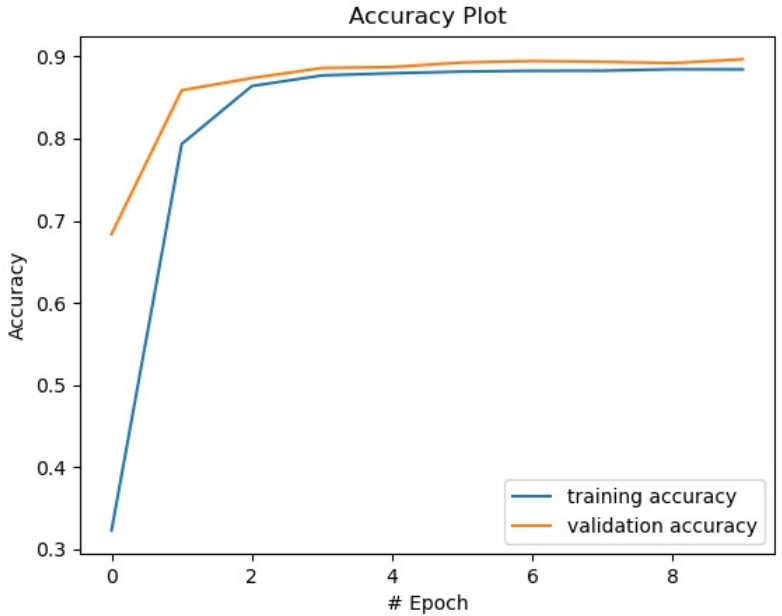
Regularization = 0.1



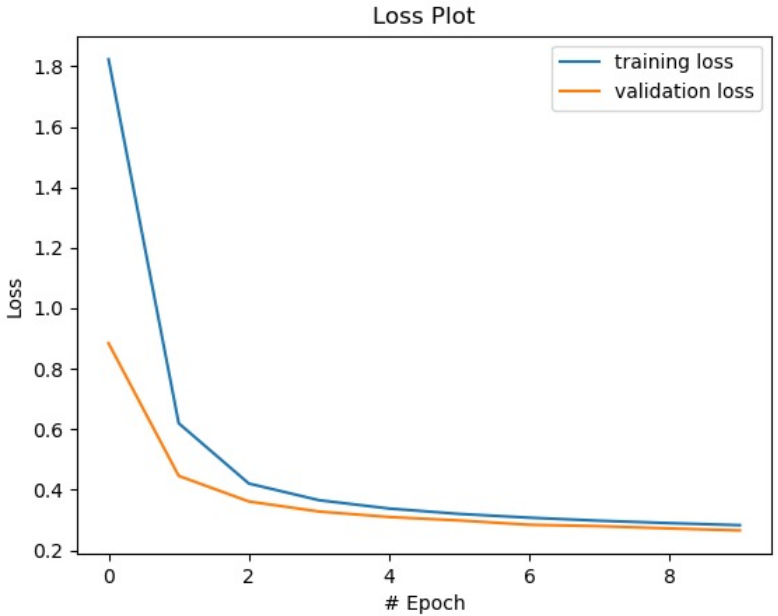


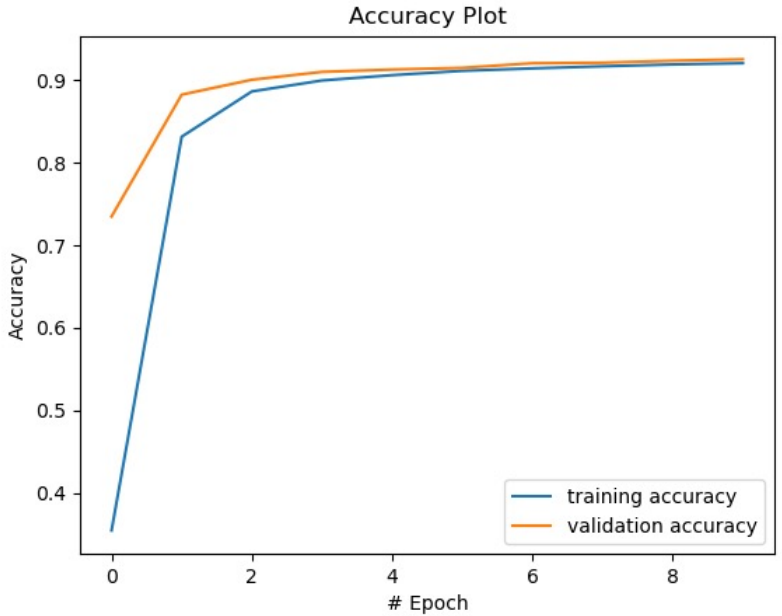
Regularization = 0.01



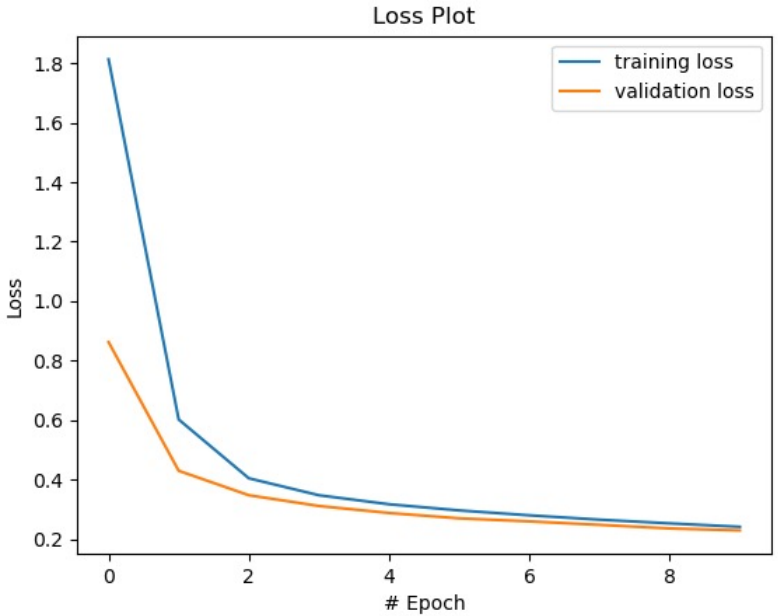


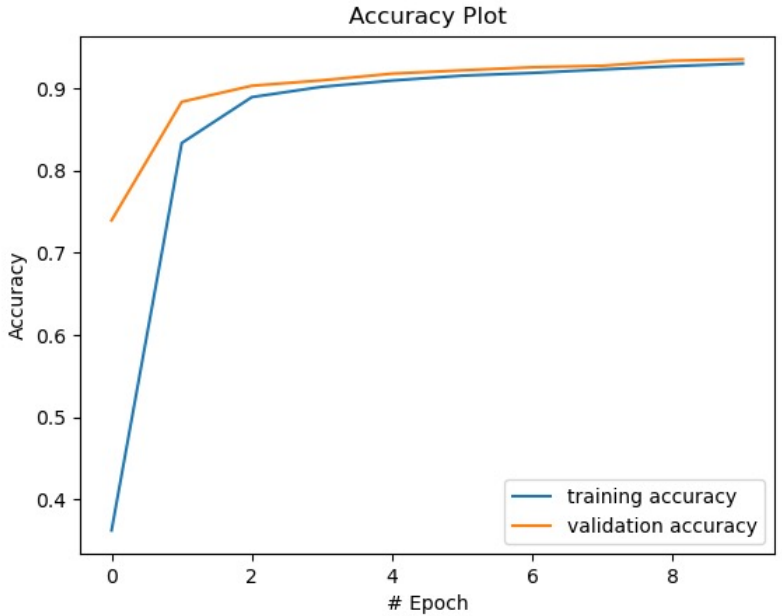
Regularization = 1e3 (default mode)





Regularization = 1e4





2. Regularization

Describe and Explain your findings: *Explanation should go into **WHY** things work the way they do in the context of Machine Learning theory/intuition, along with justification for your experimentation methodology. **DO NOT** just describe the results, for example, you should explain why the regularization value affects performance as well as model weights. If you need more than one slide to answer the question, you are free to create new slides.*

My findings: There are two basic observations. First, prediction accuracy increases with smaller regularization penalty. Second, performance fluctuates during the training process when regularization parameter is larger.

Explanations: When the regularization penalty is too large, parameters estimated though gradient descent tend to be pushed towards zero. When the effect of regularization dominates over gradient descent, learning won't be able to progress and becomes unstable, since most parameters are forced to take very small values not based on the performance of the training. This can explain both the poor performance and large variation for large regularization parameters.

However, it is difficult to say that smaller regularization is better by just comparing 1e3 and 1e4. Note that in all cases learning stops at 10 epochs. It is possible that performance might improve even after 20 epochs for slightly large regularization parameters.

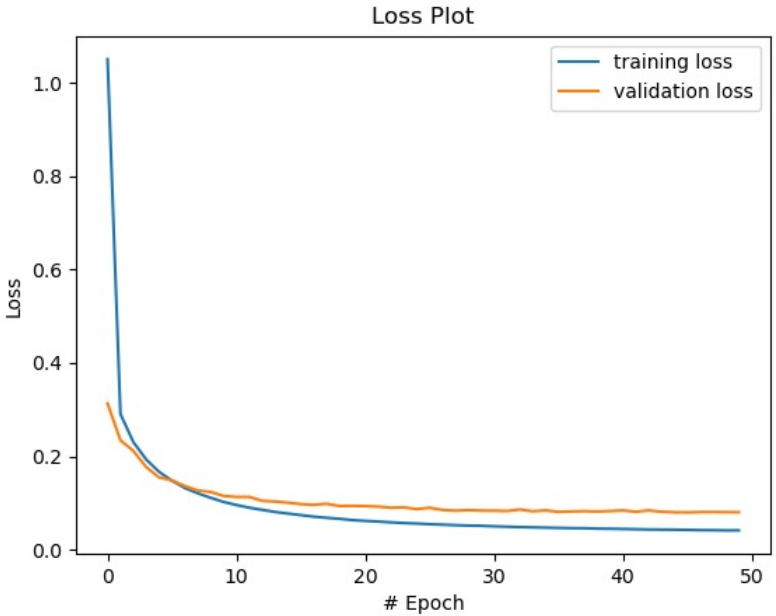
3. Hyper-parameter Tuning

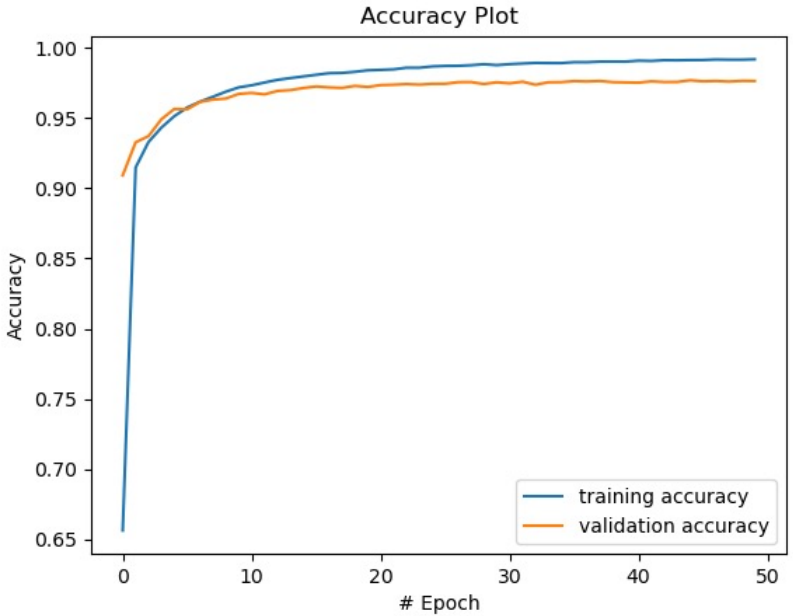
You are now free to tune any hyper-parameters for better accuracy. Create a table below and put the configuration of your best model and accuracy into the table:

Batch size	Learning rate	Regularization	Hidden size	# Epochs	Train Acc	Validation Acc	Test Acc
128	1.0	2e4	256	50	0.9902	0.9763	0.9767

Explain why your choice works: *Explanation should go into **WHY** things work the way they do in the context of Machine Learning theory/intuition, along with justification for your experimentation methodology. **DO NOT** just describe the results, you should explain the reasoning behind your choices and what behavior you expected. If you need more than one slide to answer the question, you are free to create new slides.*

Summarize plot





Explanations

- The first set of parameters I want to change is to increase the number of hidden nodes and increase the number of epochs.
  - Increasing the number of hidden nodes can increase the power of the model to fit the data. However, this might lead to overfitting issue. So, I changed regularization parameter from 1e4 (the best regularization in part 2.2) to 2e4 to control overfitting.
  - To make sure learning reaches stable state, I increase the number of epochs from 10 to 50.
- Normally I should use small learning rate to gradually improve the accuracy. However, setting learning rate to even 0.1 makes the training process too slow. I have tried learning rate 0.05 and run for 1,000 epochs and only got 96.5% accuracy on test data. Therefore, I use 1.0 for the learning rate to speed up the convergence.
- I observed that the trained model can have really poor performance on a small number of batches no matter how well it can perform on the validation set. My intuition is that batch size might be too small. That is why I changed batch size from 64 to 128.
- It is well known that simple FC neural network can reach >98.5% accuracy over mnist dataset. I think one reason my model did not perform that well is because I did not use a small enough learning rate to explore the parameter space. With higher computational power my model can be improved.