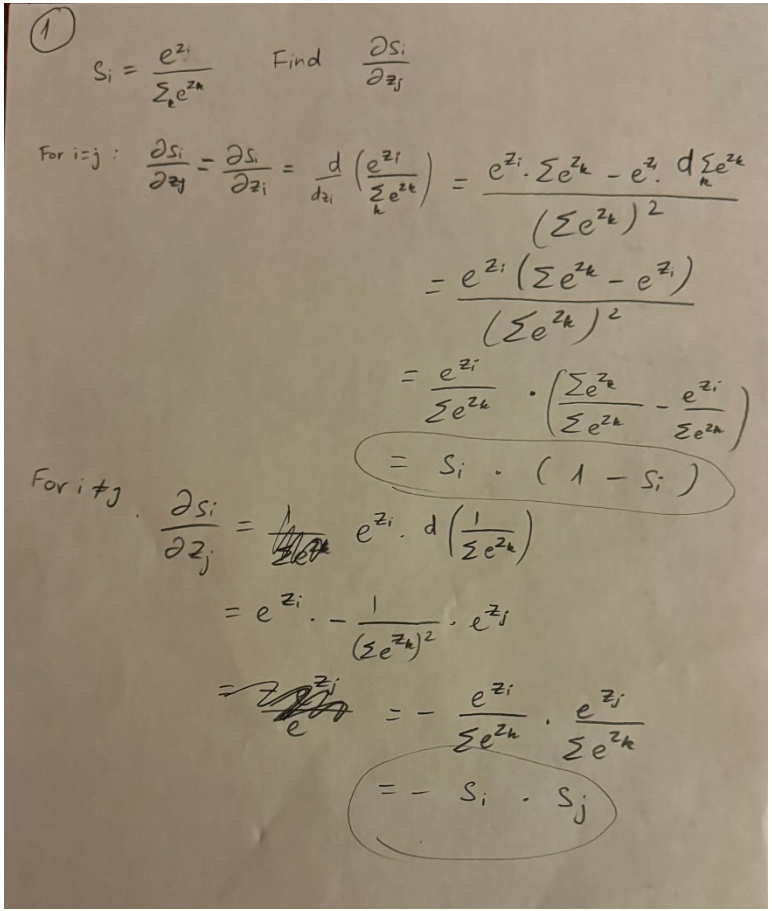
Assignment 1 Theory Problem Set

DO NOT TAG

Name: Trung Pham
GT Email: tpham328@gatech.edu

Theory PS Q1. Feel free to add extra slides if needed.



(1)

$S_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$ Find $\frac{\partial S_i}{\partial z_j}$

For $i=j$: $\frac{\partial S_i}{\partial z_j} = \frac{\partial S_i}{\partial z_i} = \frac{d}{dz_i}\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right) = \frac{e^{z_i}\cdot\sum e^{z_k} - e^{z_i}\cdot d\sum_k e^{z_k}}{(\sum e^{z_k})^2}$

$= \frac{e^{z_i}(\sum e^{z_k} - e^{z_i})}{(\sum e^{z_k})^2}$

$= \frac{e^{z_i}}{\sum e^{z_k}} \cdot \left(\frac{\sum e^{z_k}}{\sum e^{z_k}} - \frac{e^{z_i}}{\sum e^{z_k}}\right)$

$= S_i \cdot (1 - S_i)$

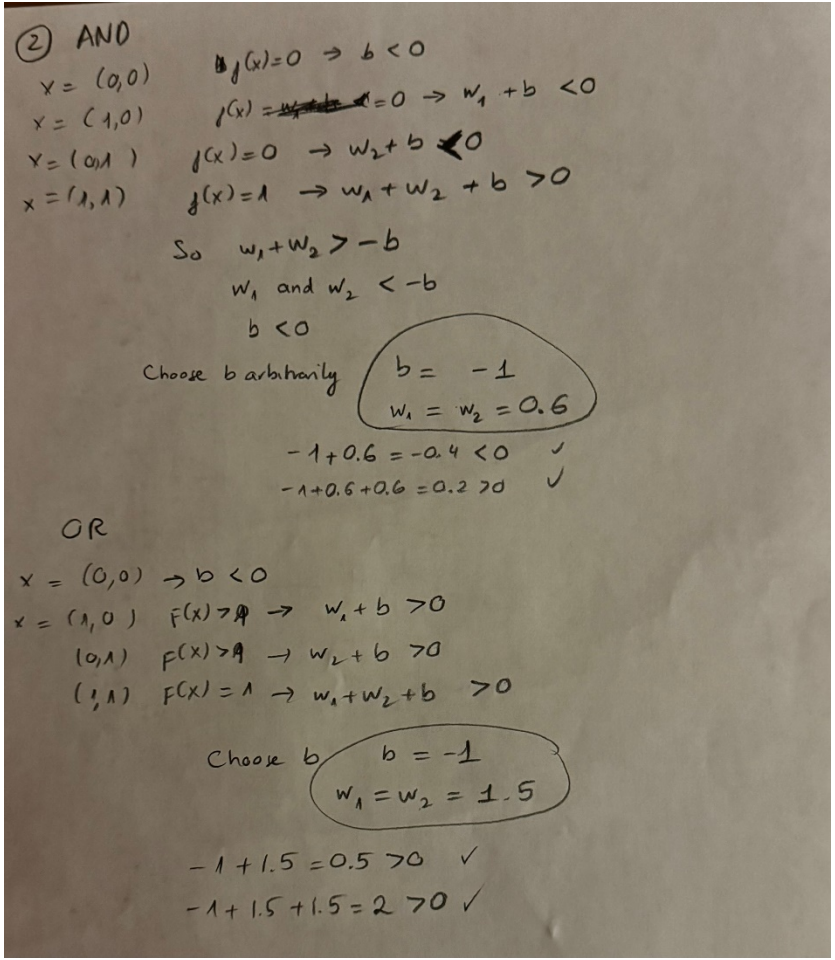
For $i \neq j$: $\frac{\partial S_i}{\partial z_j} = \frac{1}{\sum e^{z_k}} e^{z_i} \cdot d\left(\frac{1}{\sum e^{z_k}}\right)$

$= e^{z_i} \cdot -\frac{1}{(\sum e^{z_k})^2} \cdot e^{z_j}$

$= \frac{e^{z_i}}{e^{z_j}} = -\frac{e^{z_i}}{\sum e^{z_k}} \cdot \frac{e^{z_j}}{\sum e^{z_k}}$

$= -S_i \cdot S_j$

Theory PS Q2. Feel free to add extra slides if needed.



Theory PS Q3. Feel free to add extra slides if needed.

(3) Plot 4 points (x_1, x_2), prog no linier boundary can seperate them

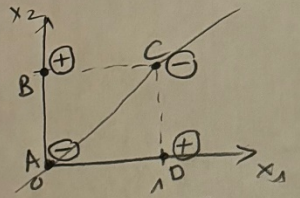
$XOR(x_1, x_2) = f(x_1, x_2)$

$(x_1, x_2) = (0,1)$ $F(x_1, x_2) = 1$ ⊕

$(x_1, x_2) = (1,0)$ $F(x_1, x_2) = 1$ ⊕

$(x_1, x_2) = (0,0)$ $F(x_1, x_2) = 0$ ⊖

$(x_1, x_2) = (1,1)$ $F(x_1, x_2) = 0$ ⊖



From the graph, we see that there is no way to seperate ⊕ points B and D with ⊖ point A and C, because B, D are accros from AC line/plane. Cannot seperate AC, BD with linier boundary

→ cannot be represented by linier model.

Assignment 1 Paper Review

DO NOT TAG

Name: Trung Pham
GT Email: tpham328@gatech.edu

Provide a short preview of the paper of your choice. “Weight Agnostic Neural Networks” by Adam Gaier and David Ha

Traditional neural network start with a network architecture and train weights through back propagation process. The paper propose a new approach where we focusing on searching for Weight Agnostic Neural Network (WANNs) architectures without learning weights. A single shared weight is applied to all connection, and only network topology is updated.

The paper shift traditional focus, emphasizing the importance of neural network dimension and topology. Some network that are discovered used very few connections but solved complex problem effectively. This simpleness also help interpretation. The paper demonstrated that network architecture possess useful inductive bias, and foster further research in network search. Skipping weight training also reduce computation task.

There are a few drawbacks. Network found during the experiment did not match the performance of convolutional neural network. Although future research can bring this gap closer. The approach in the paper also lack scalability due to applying to only a few typical task, none of them is super high dimensional.

My take away is that neural network architecture is also important. It would be more effective to apply a hybrid approach where we both find WANNs and train weights on those models. Furthermore, the paper encourage more search in the topic to discover new architectures that “possess inductive biases for practical domain” as well as “train with algorithm that may not require gradient computation.”

Paper specific Q1. Feel free to add extra slides if needed.

From the perspective of search, we see that the target search have been change. We no longer search for weights but the network topology that works with any shared weights.
Because of our target has change, our search space also has change, from weight scalars to structure.
Search method is also changed. In weight training, we relied on backpropagation and gradient descending to find optimal weights. The operator used to search network topology is based on neuroevolutionary algorithm (NEAT) so it is a kind of evolutionary strategy to find robust network structure.

Paper specific Q2. Feel free to add extra slides if needed.

Neural network architecture can contain significant representational power, even when with a shared weight value.

The method for determining weights does matter because arbitrary function might or might not be represented well by just searching topology, so the method of determining weights is still necessary. Again, a full parameterized neural network can approximate any function but WANNs cannot.

The paper mentioned that performance of WANNs might not be as good as convolutional neural network (CNN). This indicated that a full parameterized network has greater representational power. The paper indicates the importance of network architectures, showing it can solve some control tasks but not all. There are complex task only conventional deep network with trained weights can do.

Assignment 1 Writeup

DO NOT TAG

Name:
GT Email:

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	0.9433	0.9217	0.9082	0.7292
Test Accuracy	0.9507	0.9259	0.9125	0.7615

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).

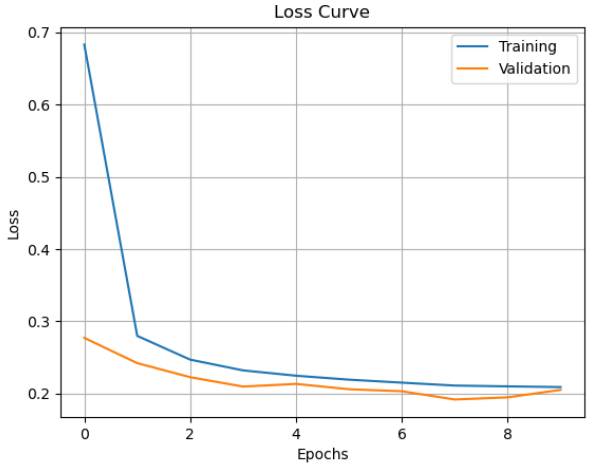


Figure 1: Loss curve with learning rate = 1

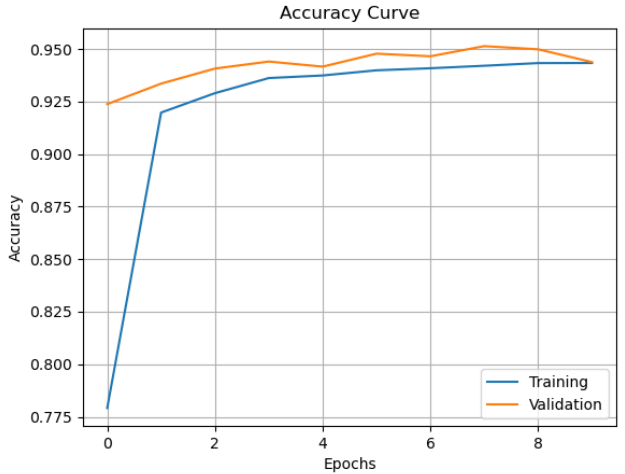


Figure 2: Accuracy curve with learning rate = 1

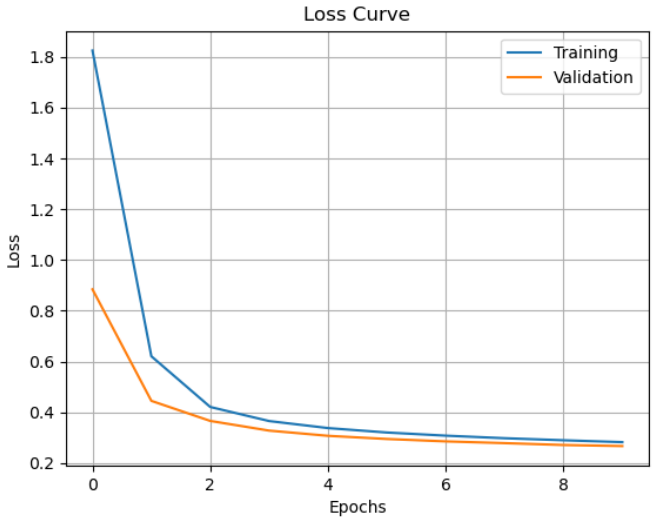


Figure 3: Loss curve with learning rate = 1e-1

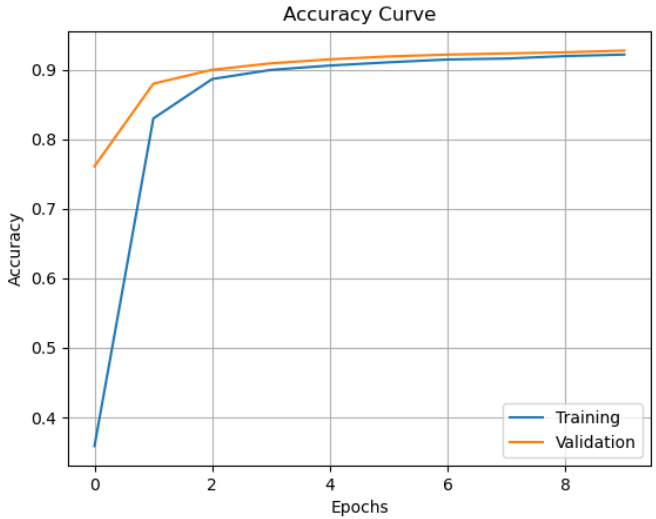


Figure 4: Accuracy curve with learning rate = 1e-1

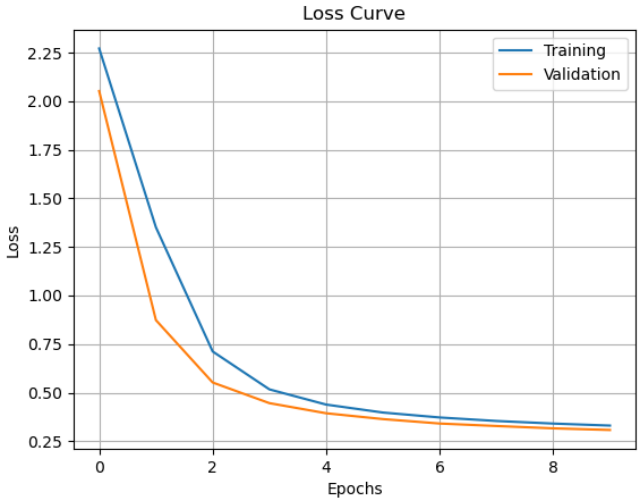


Figure 5: Loss curve with learning rate = 5e-2

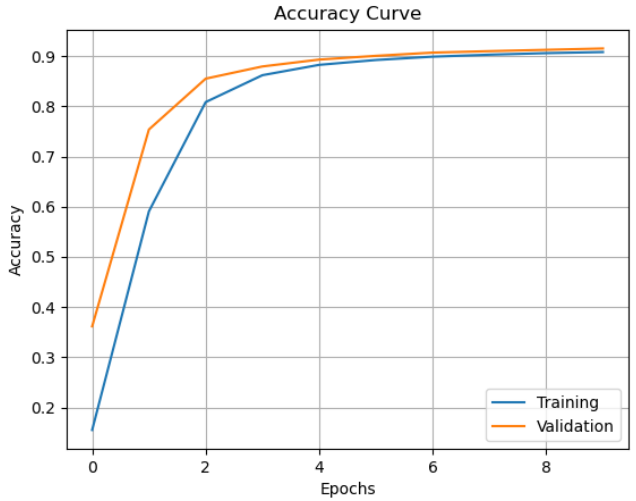


Figure 6: Accuracy curve with learning rate = 5e-2

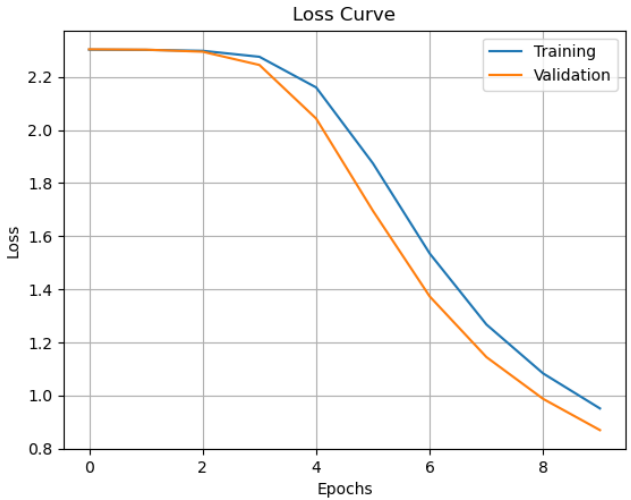


Figure 7: Loss curve with learning rate = 1e-2

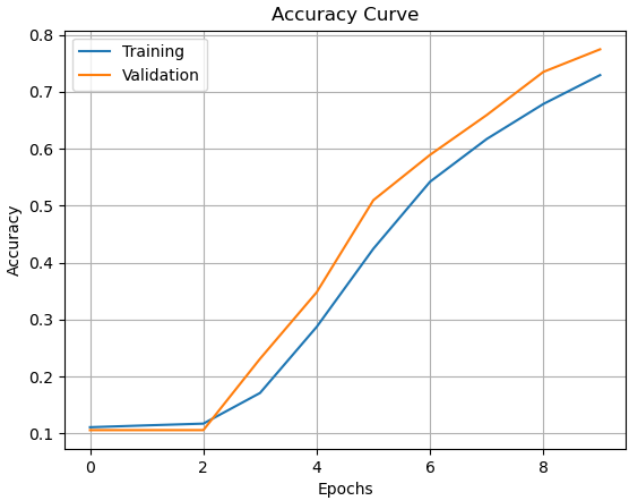


Figure 8: Accuracy curve with learning rate = 1e-2

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. Also, be cognizant of the best way to organize and show the results that best emphasizes your key observations. If you need more than one slide to answer the question, you are free to create new slides.*

Higher learning rate mean the model weights are updated faster based on the gradients. This could lead to faster convergence. Learning rate equals 1 in figure 1 and 2 is really high. We can see sharp decrease in loss and increase in accuracy after first epochs. It still work with current data, but updating weight too fast might lead to over adjustment and fails to converge.

With smaller learning rate (0.1 and 0.05) the model converge a bit more slowly but still good enough, indicating this is a sweet spot for model with epoch equals 10. At high learning rate, epoch can be lower, and vice versa. This is a trade off between learning rate and epoch, slower learning rate, we need to train the model longer.

In case learning rate is 0.01, the model does not converge after 10 epochs, accuracy is low. This indicates the model is underfitted because the weights are not updated fast enough. To fix this, we need to increase epochs or increase the learning rate. Also learning should not be too high like 1 to avoid instability, model over jumping the local minima.

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	0.1015	0.3389	0.8851	0.9214	0.9297
Validation Accuracy	0.1060	0.3097	0.8941	0.9254	0.9349
Test Accuracy	0.1135	0.3687	0.8949	0.9260	0.9342

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).

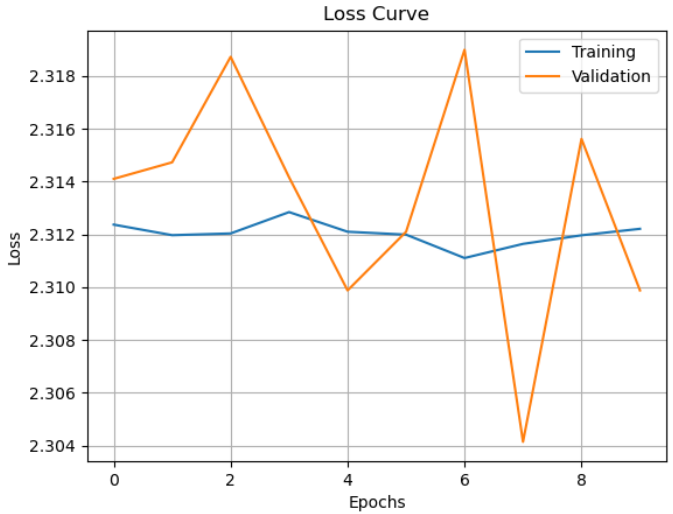


Figure 9: Loss curve with alpha = 1

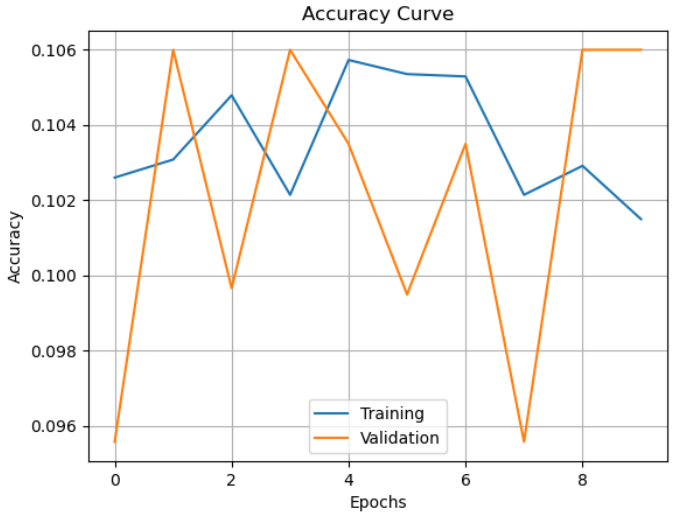


Figure 10: Accuracy curve with alpha = 1

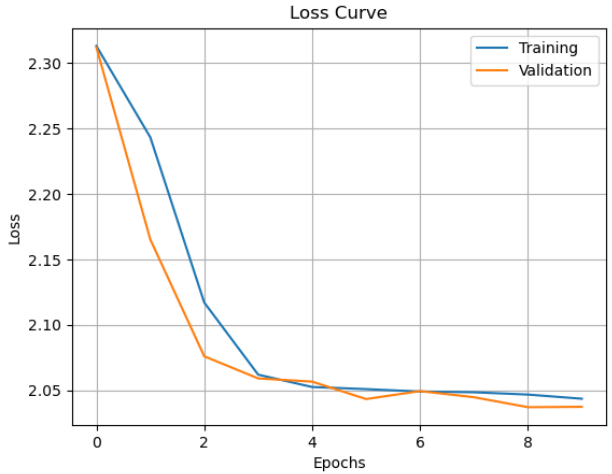


Figure 11: Loss curve with alpha = 1e-1

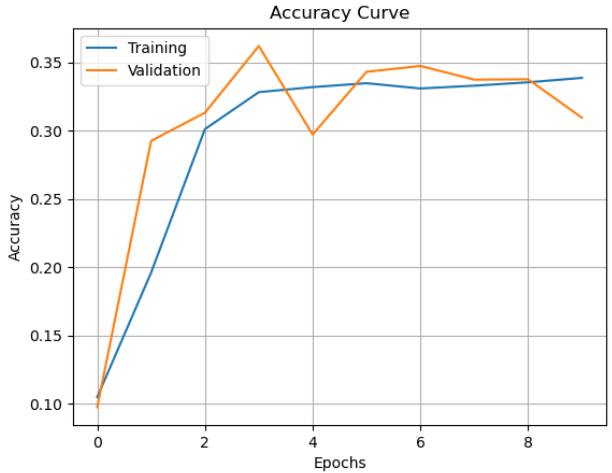


Figure 12: Accuracy curve with alpha = 1e-1

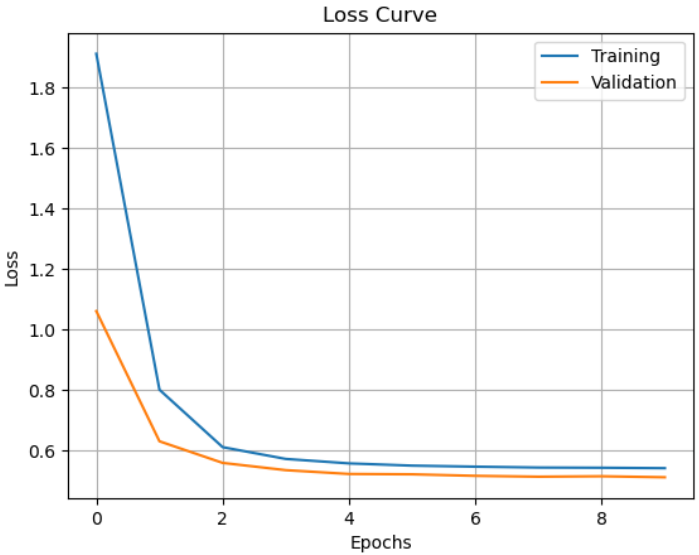


Figure 13: Loss curve with alpha = 1e-2

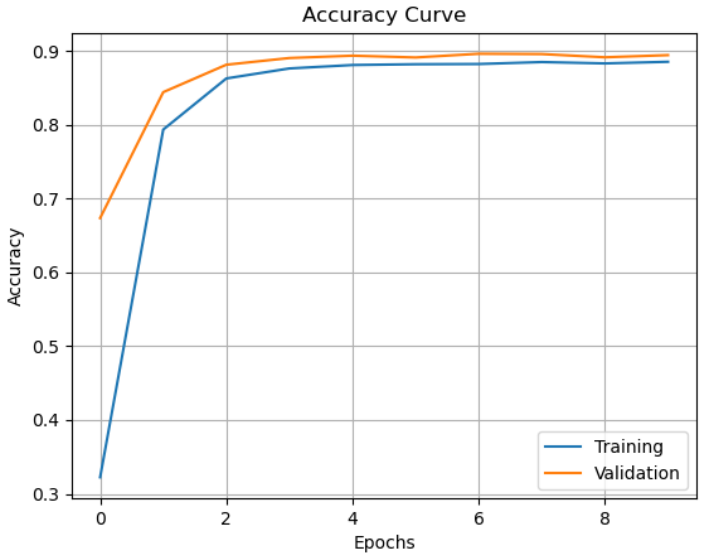


Figure 14: Accuracy curve with alpha = 1e-2

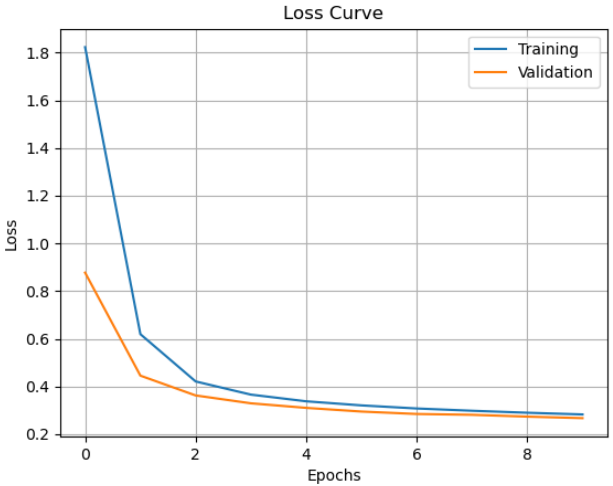


Figure 15: Loss curve with alpha = 1e-3

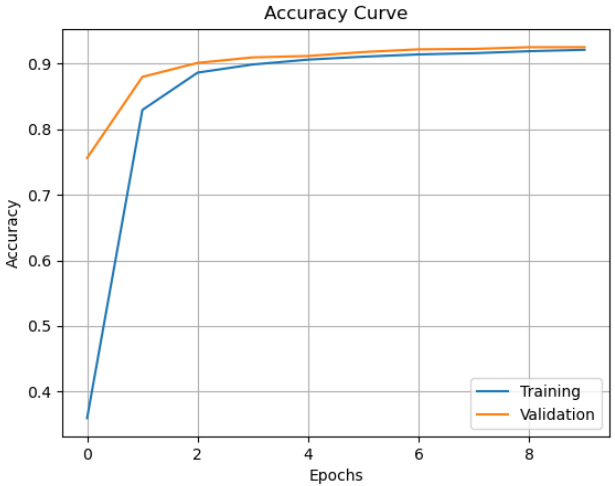


Figure 16: Accuracy curve with alpha = 1e-3

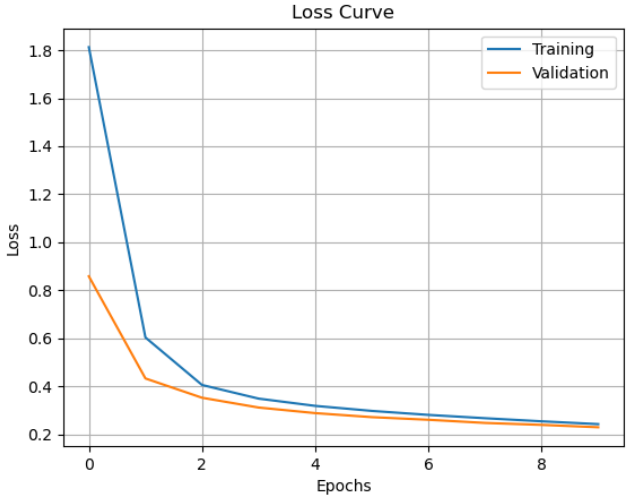


Figure 17: Loss curve with alpha = 1e-4

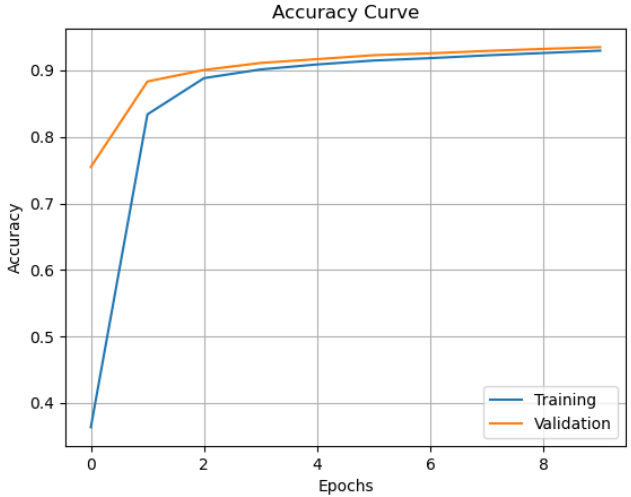


Figure 18: Accuracy curve with alpha = 1e-4

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. Also, be mindful of the best way to organize and show the results that best emphasizes your key observations. If you need more than one slide to answer the question, you are free to create new slides.*

First of all, alpha in L2 regularization penalize the model for having large weights. A large alpha will steer the model toward smaller weights. If alpha is too large (like alpha = 1), the model tried to keep all the weights small, and might fails to capture meaningful patterns, leading to random prediction (loss curve and accuracy curve bouncing around). Model is underfitted.

When alpha equal 0.1, the model have improvement we can see in the loss curve but still high, accuracy is low. Only until alpha is 0.01 or smaller, we have better performance. All last three cases have similar plots. Training and validation curve show converge, accuracy is high. The decease in alpha allowed the model to have higher weight to reduce training loss, but still discourage explore high magnitude enough.

We also see improvement in generalization, the model is not overfitted with lower validation loss and higher validation accuracy as alpha decrease. This improvement is no longer significant after alpha is smaller than 0.001, indicating the sweet spot area. As alpha keep decreasing, training accuracy will keep increase until it hits near 100% indicating overfitting. We have not see it here when alpha equals 0.0001.

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	32
Learning rate	0.55
Reg	0.0005
Hidden size	256

Other unchanged hyper-parameters: epochs = 10, momentum = 0.9

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. Also, be cognizant of the best way to mindful and show the results that best emphasizes your key observations. If you need more than one slide to answer the question, you are free to create new slides.*

Epochs is not included in the hyperparameter tuning. Given epochs = 10, there is a sweet spot for learning rate. Based on previous tuning exercise, the sweet spot is 0.1. Also because of low epochs, we increase learning rate and found out 0.5 is optimal rate for learning.
We also update the batch size lower, because we want to update weight more frequently. This helps generalization while keeping epoch unchanged.
Given the model is complex (computer vision problem) with 784 features (pixels) inputs, we increase the hidden size of the hidden layer to 256. This can lead to overfitting so we also tune the L2 regularization parameter.
Based on previous exercise, we found the sweet spot for alpha is below 0.001, our tuning settle when alpha/reg equals 0.0005

Our final results is : average accuracy of epoch 10 is 0.9613
Validation accuracy is 0.9593
Final accuracy in test data is 0.9617