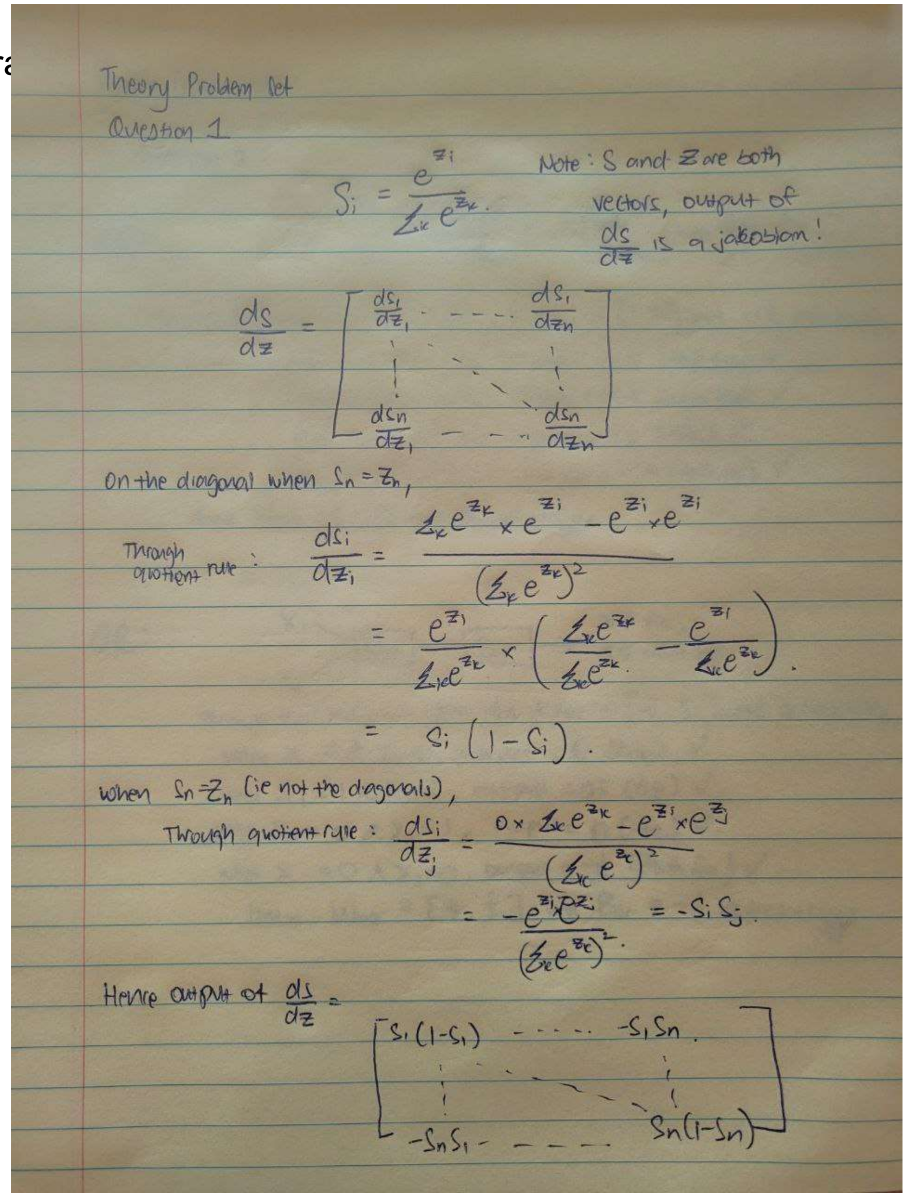
Assignment 1 Theory Problem Set

DO NOT TAG

Name: Marcus Tan
GT Email: mtan75@gatech.edu

Theory PS Q1. Feel free to add extra



Theory Problem Set

Question 1

$$S_i = \frac{e^{z_i}}{\sum_k e^{z_k}}.$$

Note: S and Z are both vectors, output of $\frac{ds}{dz}$ is a jakobiam!

$$\frac{ds}{dz} = \begin{bmatrix} \frac{ds_1}{dz_1} & \cdots & \frac{ds_1}{dz_n} \\ \vdots & \ddots & \vdots \\ \frac{ds_n}{dz_1} & \cdots & \frac{ds_n}{dz_n} \end{bmatrix}$$

On the diagonal when $S_n = Z_n$,

Through quotient rule:

$$\frac{ds_i}{dz_i} = \frac{\sum_k e^{z_k} \times e^{z_i} - e^{z_i} \times e^{z_i}}{(\sum_k e^{z_k})^2}$$

$$= \frac{e^{z_i}}{\sum_k e^{z_k}} \times \left(\frac{\sum_k e^{z_k}}{\sum_k e^{z_k}} - \frac{e^{z_i}}{\sum_k e^{z_k}} \right).$$

$$= S_i (1 - S_i).$$

when $S_n = Z_n$ (ie not the diagonals),

Through quotient rule:

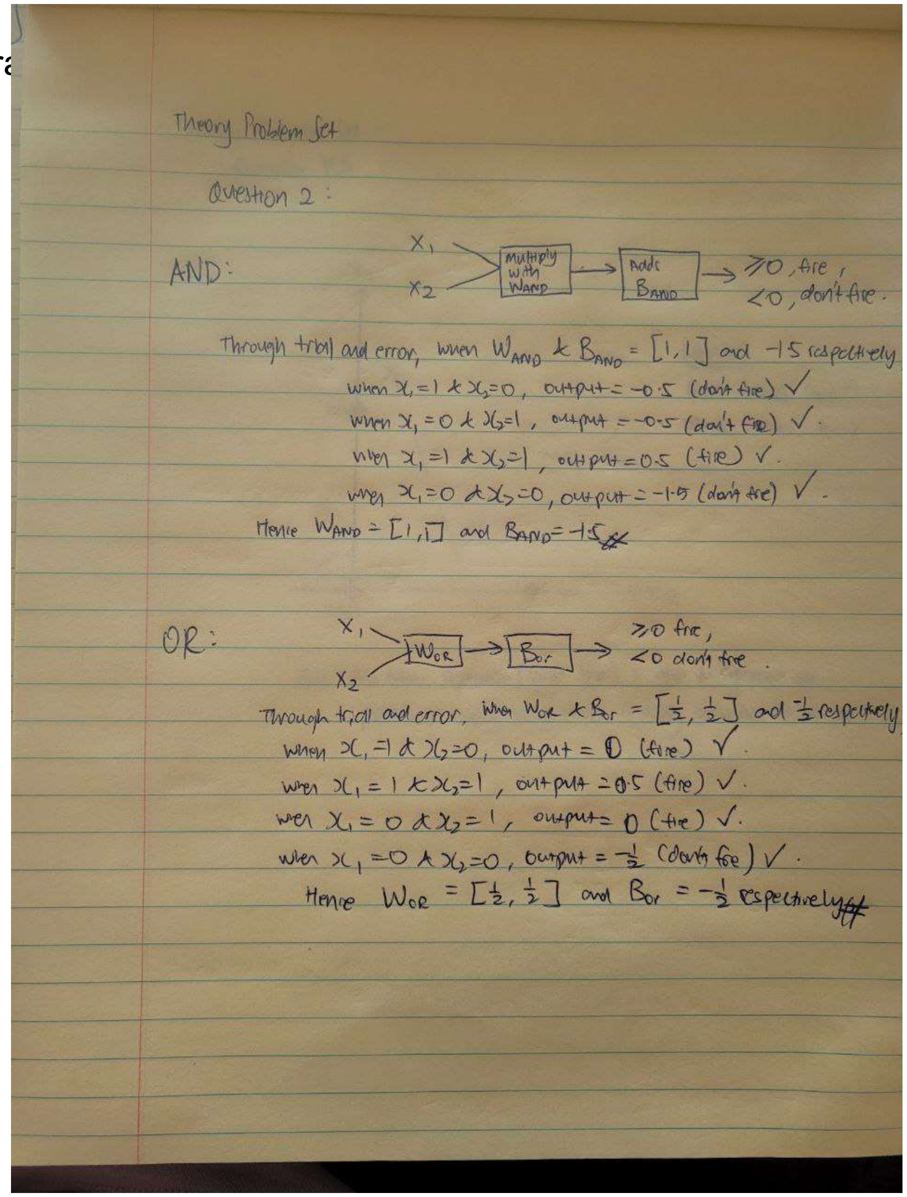
$$\frac{ds_i}{dz_j} = \frac{0 \times \sum_k e^{z_k} - e^{z_i} \times e^{z_j}}{(\sum_k e^{z_k})^2}$$

$$= \frac{-e^{z_i} e^{z_j}}{(\sum_k e^{z_k})^2} = -S_i S_j.$$

Hence output of $\frac{ds}{dz}$ =

$$\begin{bmatrix} S_1(1-S_1) & \cdots & -S_1 S_n \\ \vdots & \ddots & \vdots \\ -S_n S_1 & \cdots & S_n(1-S_n) \end{bmatrix}$$

Theory PS Q2. Feel free to add extra



Theory Problem Set

Question 2 :

AND:

$$\begin{tikzcd} x_1 \arrow[dr] & & & \\ & \boxed{\text{Multiply with } W_{AND}} \arrow[r] & \boxed{\text{Adds } B_{AND}} \arrow[r] & \geq 0 \text{, fire ,} \\ x_2 \arrow[ur] & & & < 0 \text{, don't fire .} \end{tikzcd}$$

Through trial and error, when W_{AND} & B_{AND} = $[1,1]$ and -1.5 respectively

when $x_1 = 1$ & $x_2 = 0$, output $= -0.5$ (don't fire) ✓

when $x_1 = 0$ & $x_2 = 1$, output $= -0.5$ (don't fire) ✓.

when $x_1 = 1$ & $x_2 = 1$, output $= 0.5$ (fire) ✓.

when $x_1 = 0$ & $x_2 = 0$, output $= -1.5$ (don't fire) ✓.

Hence $W_{AND} = [1,1]$ and $B_{AND} = -1.5$ #

OR:

$$\begin{tikzcd} x_1 \arrow[dr] & & & \\ & \boxed{W_{OR}} \arrow[r] & \boxed{B_{or}} \arrow[r] & \geq 0 \text{ fire ,} \\ x_2 \arrow[ur] & & & < 0 \text{ don't fire .} \end{tikzcd}$$

Through trial and error, when W_{OR} & B_{or} = $[\frac{1}{2}, \frac{1}{2}]$ and $-\frac{1}{2}$ respectively

when $x_1 = 1$ & $x_2 = 0$, output $= 0$ (fire) ✓.

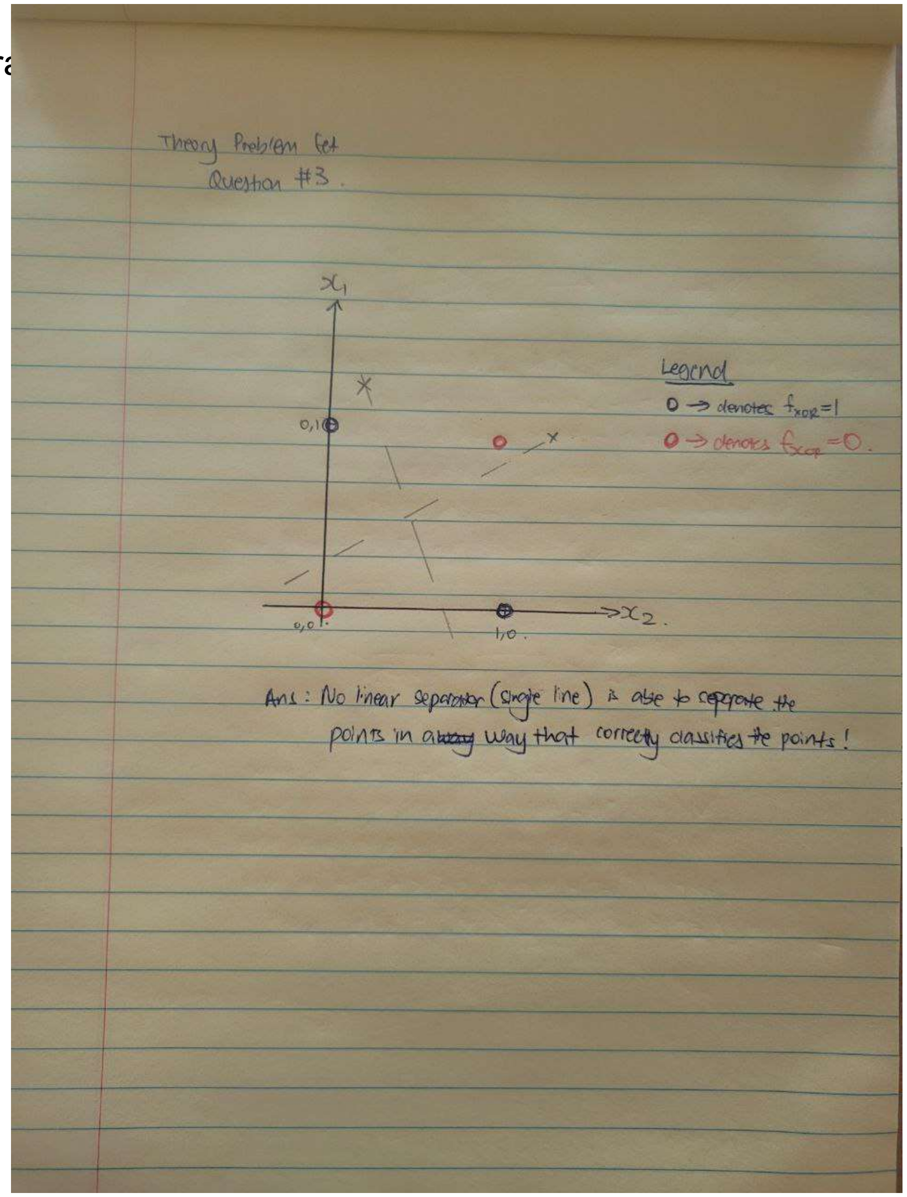
when $x_1 = 1$ & $x_2 = 1$, output $= 0.5$ (fire) ✓.

when $x_1 = 0$ & $x_2 = 1$, output $= 0$ (fire) ✓.

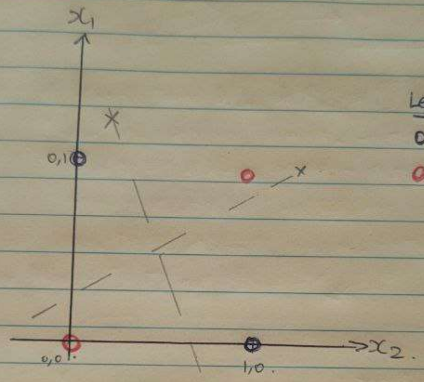
when $x_1 = 0$ & $x_2 = 0$, output $= -\frac{1}{2}$ (don't fire) ✓.

Hence $W_{OR} = [\frac{1}{2}, \frac{1}{2}]$ and $B_{or} = -\frac{1}{2}$ respectively #

Theory PS Q3. Feel free to add extra



Theory Problem Set
Question #3.



Legend
● → denotes $f_{XOR}=1$
● → denotes $f_{XOR}=0$.

Ans: No linear separator (single line) is able to separate the points in a ~~way~~ way that correctly classifies the points!

Assignment 1 Paper Review

DO NOT TAG

Name:
GT Email:

Provide a short preview of the paper of your choice.

Question: What is the main contribution of this paper? In other words, briefly summarize its key insights.
I reviewed paper 2 and found this paper interesting, as it discusses a unique research into neural network that I had not encountered before. It explores **(1)** how neural networks excel in predicting accurate outcomes, **(2)** performs well on the training dataset even when trained on random labels, and **(3)** examines the role of both implicit and explicit regularizations.

For me, the success of neural networks is evident, particularly given their widespread adoption in recent years across various practical applications, such as computer vision and recommender systems. Being able to produce good training dataset results (even with random labels) is not surprising.

Question: What is your personal takeaway from this paper?
What I found particularly interesting is the discussion on regularizations. Explicit regularizations, like dropout and weight decay, and implicit ones, such as minimizing norms via stochastic gradient descent (SGD) and early stopping, are crucial for improving generalizability. I used to think of regularizations as penalties that lead to lower training scores, which seemed counter to how we want models to perform. However, this paper offers a different perspective, showing that regularizations, whether explicit or implicit, can occasionally encourage better generalization, as demonstrated by the results.

However, one area that is not discussed in this paper is the topic of detecting labels that are clean/random (such as in the case of anomaly detection). Since this paper talked about generating random labels and the implications of these labels, the practical question that we should ask is, how do we identify if our dataset is contaminated with random labels so that we ensure that the "cleanest" data is presented to the model? The topic of anomaly detection has been largely discussed in Machine Learning context, and hence it would be interesting to further add to this writeup, the author's thoughts on identifying clean datasets.

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

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

Question: Why do you think neural networks learn more meaningful, generalizable representations when there are meaningful patterns in the data?
The paper discusses how neural networks can learn more meaningful and generalizable representations when significant patterns exist within the data. It demonstrates that even a simple two-layer ReLU network, with 2n + d parameters, possesses vast expressiveness. When meaningful representations are present, this expressiveness allows the network to effectively map out functions and uncover relationships between variables.

Question: How does this finding align or not align with your understanding of machine learning and generalization?
This finding aligns with my understanding of neural networks, which I perceive as "accuracy-at-all-cost" models, distinct from traditional machine learning approaches. The expressive power of neural networks allows data scientists to address complex problems without needing to identify the specific statistical distribution that best fits the dataset, and yet achieve high accuracy. However, the paper also underscores a critical consideration: the increased complexity that comes with deeper neural networks can make them challenging to interpret and explain, highlighting the need to carefully balance accuracy with interpretability.

Question: What are some strengths and weaknesses of this paper?
Elaborating further on the earlier point, the key weakness in this paper is that it delves too deeply into theoretical application and have not discussed on the practical application of the results. Deep learning/neural network has many vast applications in our daily lives – especially in scenarios where accuracy is of utmost importance. However, in many business settings today, getting extremely accurate prediction is not exactly what businesses need all the time. Many analyst today still rely on rule-based logic or simple linear regressions because they’re easily explainable – which is what is required from stakeholders. To increase the practical application of the results in this paper, it would help for this paper to discuss further on the expressiveness vs explainability of neural networks in general.

Assignment 1 Writeup

DO NOT TAG

Name:
GT Email:

Two-Layer Neural Network

DO NOT TAG

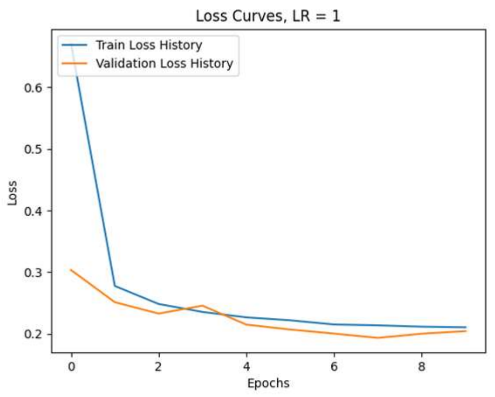
1. Learning Rates

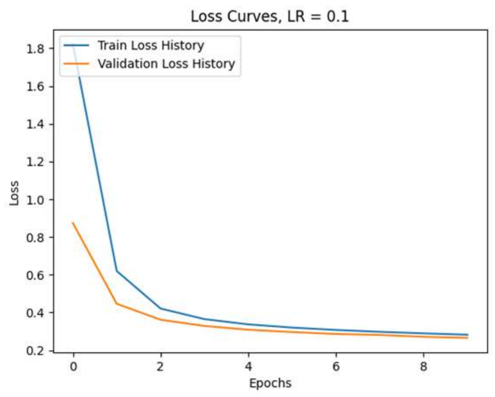
Tune the learning rate of the model with all other default hyper-parameters fixed. Fill in the table below:

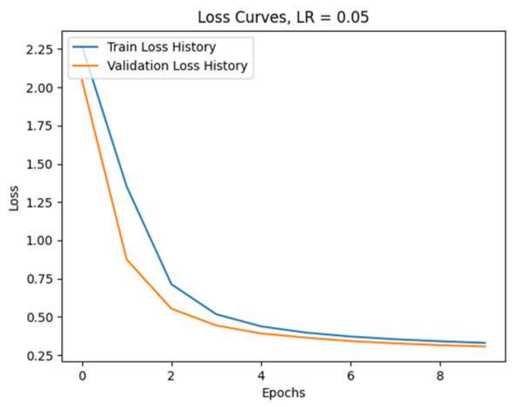
	lr=1	lr=1e-1	lr=5e-2	lr=1e-2
Training Accuracy	0.9429	0.9214	0.9089	0.6182
Test Accuracy	0.9450	0.9238	0.9136	0.7605

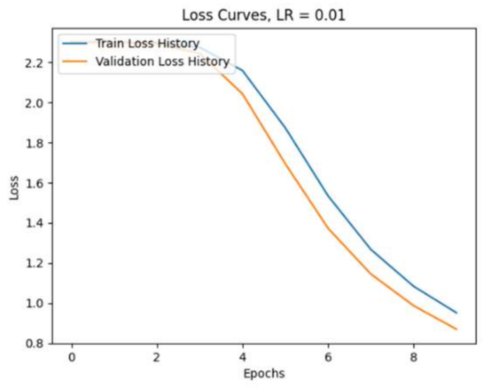
1. Learning Curve

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









1. Learning Rates

Describe and Explain your findings:

[Observation #1] Accuracy decreases across training set and test set as learning rate decreases
[**Observation #2]** Learning curve turns from an elbow curve into an inverted S-shape as learning rate decrease

[Explanation #1 for Both Observations] Generally, it's expected that accuracy may decrease as the learning rate decreases. A lower learning rate results in smaller weight updates during each iteration, slowing down the optimization process. With a fixed epoch size of 10, the model may not have enough time to converge, leading to reduced accuracy. This is evident in the learning curve, where the loss curve begins to form an inverted S-shape, indicating that convergence has not yet occurred. As discussed in the required reading for the week (https://arxiv.org/pdf/1803.09820), when using a lower learning rate, it is often necessary to adjust other parameters, such as increasing the number of epochs or batch size, to ensure smoother/more reliable results.

However, it's important to be cautious with a high learning rate as well. A high learning rate can cause the model to overshoot the minimum of the loss function, leading to non-convergence. While this issue may not be immediately evident in the charts, it underscores the need for a balanced learning rate. Achieving the right balance can improve convergence and, ultimately, model performance.

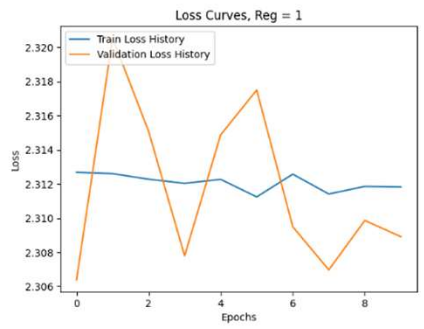
2. Regularization

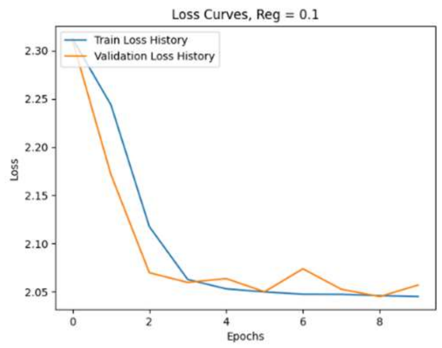
Tune the regularization coefficient of the model with all other default hyper-parameters fixed. Fill in the table below:

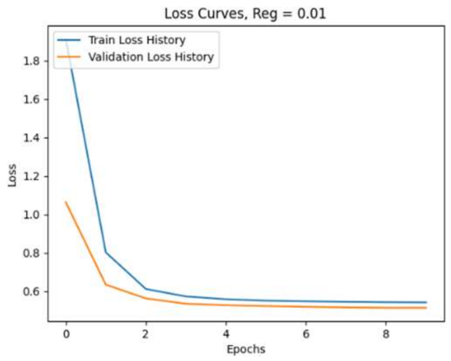
	alpha=1	alpha=1e-1	alpha=1e-2	alpha=1e-3	alpha=1e-4
Training Accuracy	0.104	0.336	0.884	0.922	0.930
Validation Accuracy	0.104	0.317	0.894	0.926	0.934
Test Accuracy	0.103	0.328	0.893	0.926	0.933

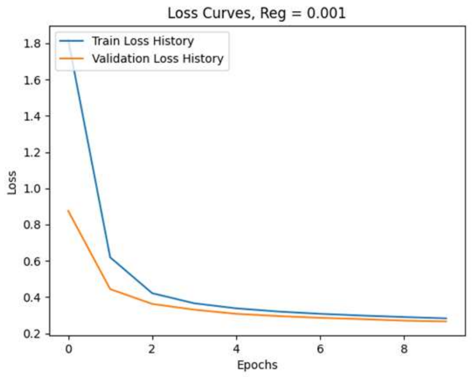
2. Regularization

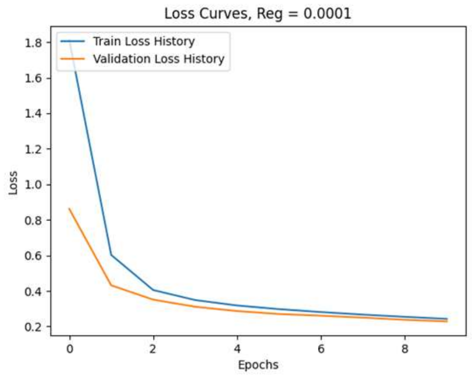
Plot the learning curves using the regularization coefficients from the previous slide and put them below (you may add additional slides if needed).











2. Regularization

Describe and Explain your findings:

[Observation #1] Lower values of regularization results in better accuracy.

[Explanation #1] The regularization method applied here is L2 loss, which imposes a penalty equal to the sum of the squared values of the weights. This can become problematic for large models, such as ours with 784 inputs. As indicated in the table, using a regularization coefficient of 1 is overly restrictive, leading to low accuracy across training, testing, and validation due to excessive penalization of the model's weights. However, as we reduce the regularization, its impact becomes less obstructive, allowing the model to achieve better accuracy, as reflected in the improved scores.

[**Observation #2]** Test and validation accuracy are higher than train accuracies

[Explanation #2] As highlighted in the lecture, it's not uncommon to observe higher accuracies on the validation and test sets compared to the training set for two main reasons:

1.The validation and test sets are not subjected to regularization, unlike the training set. Regularization can lead to slightly worse accuracy/loss in the training set as the model is penalized to prevent overfitting.
2.Training accuracy is calculated after each batch, and the model used for these calculations might not be the best version. In contrast, the validation and test sets are evaluated using the best model produced during training, which can result in higher accuracies on these sets.

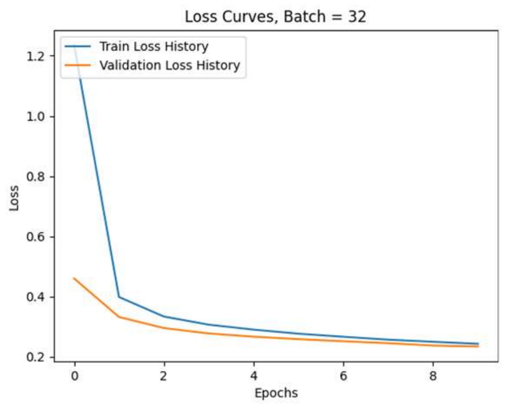
3. Hyper-parameter Tuning – #1: Batch Size

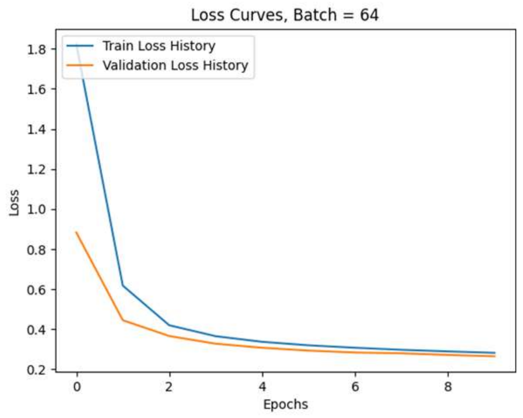
You are now free to tune any hyper-parameters for better accuracy. Create a table below and put the configuration of your best model and accuracy into the table:

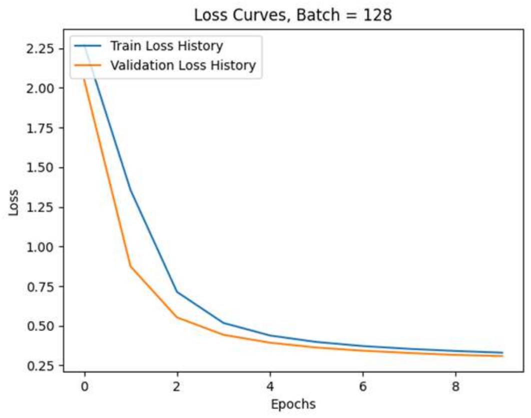
	Batch = 32	Batch = 64	Batch = 128	Batch = 256
Training Accuracy	0.933	0.922	0.908	0.887
Validation Accuracy	0.937	0.929	0.914	0.895
Test Accuracy	0.936	0.926	0.914	0.894

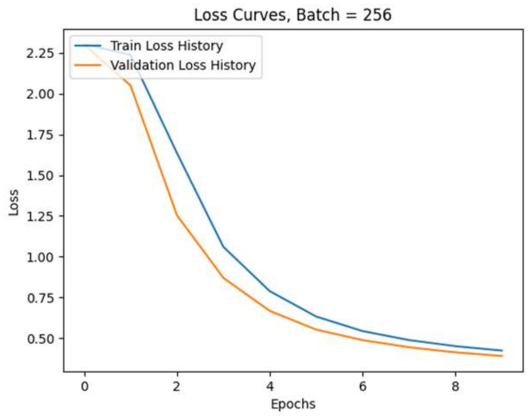
3. Batch Size

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









3. Additional Hyper-parameter Tuning - #1: Batch Size

Explain:

[Observation #1] Lower values of batch size results in better accuracy.

[Explanation #1] We observe that lower batch sizes lead to better accuracy, likely due to more frequent updates during training. Smaller batches result in the model updating its weights more often, which helps it explore the loss landscape more thoroughly.

Additionally, as mentioned in the lecture, smaller batch sizes introduce more noise into the training process. This noise acts as implicit regularization, helping the model generalize better to unseen data by reducing the risk of overfitting to the training set. While smaller batches can help with regularization, the stochastic nature of gradient descent is amplified, leading the optimization process to converge to local minima.

[**Observation #2]** Learning curve turns from an elbow curve into an inverted S-shape as batch size increase

[Explanation #2] Similar to the effects observed with learning rates, larger batch sizes require a larger learning rate to offset the increased variability in weight updates due to the smaller sample size. As illustrated by the loss curve, a larger batch size (e.g., 256) leads to slower convergence, primarily because the learning rate is too small. This issue can be addressed by increasing the learning rate proportionally to the batch size.

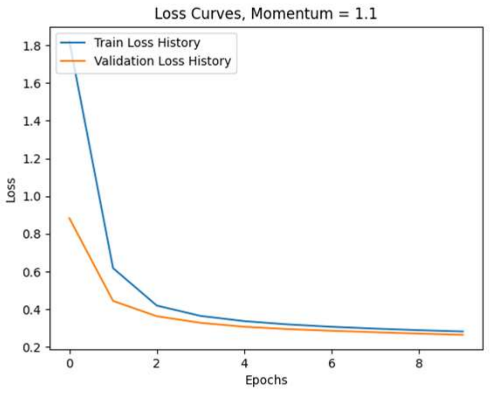
3. Hyper-parameter Tuning – #2: Momentum

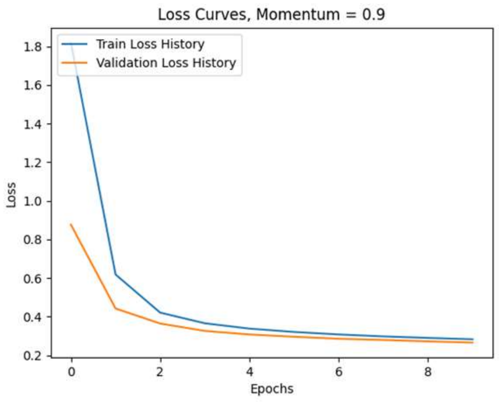
You are now free to tune any hyper-parameters for better accuracy. Create a table below and put the configuration of your best model and accuracy into the table:

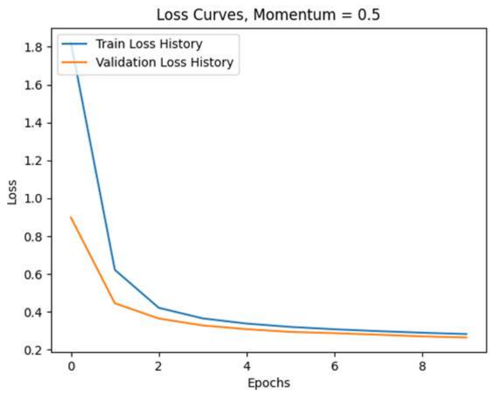
	Momentum = 1.1	Momentum = 0.9	Momentum = 0.7	Momentum = 0.5
Training Accuracy	0.921	0.921	0.921	0.921
Validation Accuracy	0.927	0.927	0.928	0.927
Test Accuracy	0.925	0.924	0.927	0.926

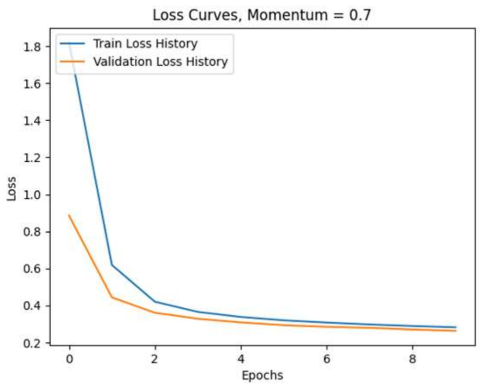
3. Momentum

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









3. Hyper-parameter Tuning - #2: Momentum

Explain:

[Observation #1] The optimal momentum is 0.7, as values that are too high (e.g., 1.1) or too low (e.g., 0.5) are lower in accuracy for training and test sets. The changes in accuracy is relatively smaller when compared to other hyperparameters that we have tuned earlier.

[Explanation #1] As noted in the lecture, momentum generally has a smaller impact on final accuracy compared to learning rates or weight decay. The primary goal of fine-tuning momentum is to extract that remaining small percentage of accuracy, such as in our case. In standard deep learning applications like ours, algorithms are generally less sensitive to momentum. While momentum can aid in escaping local minima, if you have an adequate batch size and an appropriate learning rate, its effect is minimal. In such cases, we can focus less on adjusting momentum, as your model will naturally converge to a suitable local minimum.

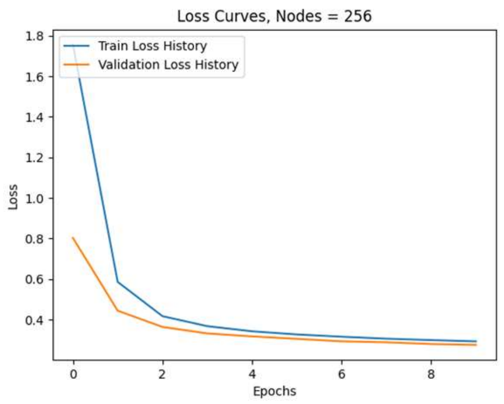
3. Additional Hyper-parameter Tuning – #2: Nodes

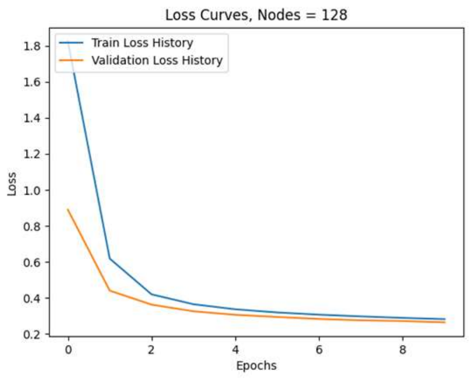
You are now free to tune any hyper-parameters for better accuracy. Create a table below and put the configuration of your best model and accuracy into the table:

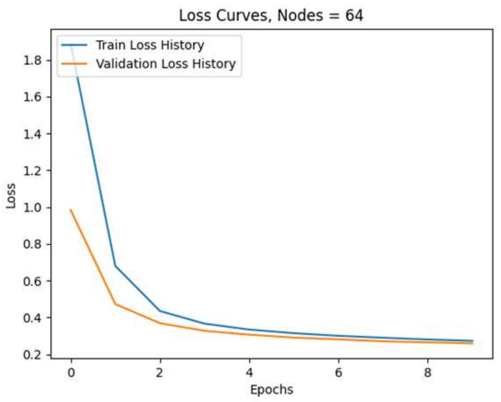
	# Nodes = 256	# Nodes = 128	# Nodes = 64	# Nodes = 32
Training Accuracy	0.918	0.921	0.925	0.926
Validation Accuracy	0.924	0.927	0.930	0.931
Test Accuracy	0.923	0.925	0.929	0.931

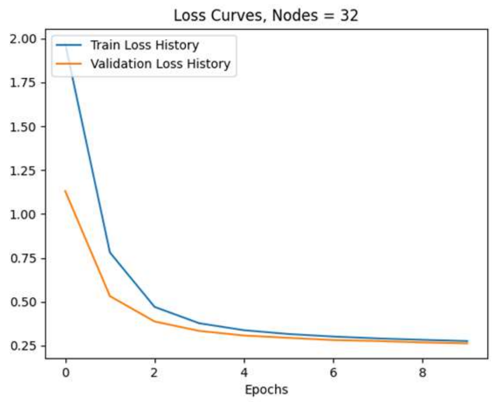
3. Nodes

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









3. Hyper-parameter Tuning - #2: Nodes

Explain why your choice works:

[Observation #1] Larger number of nodes in the hidden layer does not always equate to better test and validation accuracy.

[Explanation #1] I initially expected that increasing the number of nodes would improve accuracy. However, I observed that as the number of nodes decreases, test and validation accuracy increase. This suggests that the model might be overfitting when using a larger number of nodes. Overfitting occurs when the model becomes too complex, capturing noise in the training data rather than generalizing well to unseen data. By reducing the number of nodes, the model is simplified, leading to better generalization, as reflected in the higher test and validation accuracy. In essence, fewer nodes in this case helped to balance model complexity, preventing the model from fitting too closely to the training data and improving its ability to perform on new data.

Tuning – Combining Multiple Hyperparameters

	HP#1: Nodes = 32 HP#2: Batch = 32 HP#3: LR = 1 Epoch = 50	HP#1: Nodes = 32 HP#2: Batch = 32 HP#3: Reg = 1e-4 Epoch = 50	HP#1: Reg = 1e-4 HP#2: Batch = 32 HP#3: LR =1 Epoch = 50 (OPTIMAL)	HP#1: Nodes = 32 HP#2: Reg = 1e-4 HP#3: LR =1 Epoch = 50
Training Accuracy	0.939	0.979	0.989	0.981
Validation Accuracy	0.949	0.966	0.975	0.965
Test Accuracy	0.949	0.964	0.978	0.969
Accuracy on Default Hyperparameters	Train Accuracy: 0.921, Validation Accuracy: 0.928, Test Accuracy: 0.927			

Explanation – Combining Multiple Hyperparameters

Methodology:
Using the four optimal hyperparameters identified earlier, I experimented with different combinations involving three hyperparameter changes. Additionally, based on previous learning rate adjustments, I observed the necessity of extending the number of epochs to ensure convergence on the best result.

Observation:
Through various combinations of the four optimal hyperparameters, we determined that each combination outperformed the baseline parameters. The most effective combination is: (1) Learning Rate: 1, (2) Regularization: 0.0001, (3) Batch Size: 32, with Epochs: 50.

Explanation:
The observed results are consistent with the individual explanations provided for each hyperparameter. The combination of these settings has allowed us to explore a more effective minimum. Initially, I anticipated that some combinations might not outperform the baseline, but the results were unexpectedly positive largely. This is because the baseline was underperforming from lower epochs and larger regularization indicating ineffective learning/lack of convergence. However, as called out in the lecture, it’s important to note that simply combining optimal hyperparameters might not always guarantee better results that the baseline, as different hyperparameter combinations can sometimes lead to suboptimal minima.