Assignment 1 Theory Problem Set

Name: Burak Aksu
GT Email: Baksu3@gatech.edu

1 Question 1

We need to derive the gradient of the softmax function $s(z)$ with respect to the logits z, where:

$$s_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Let's call the denominator $S = \sum_k e^{z_k}$ for convenience. So $s_i = \frac{e^{z_i}}{S}$.
We want to find $\frac{\partial s_i}{\partial z_j}$ for all i, j. There are two cases to consider:
Case 1: $i = j$
Using the quotient rule:

$$\frac{\partial s_i}{\partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{e^{z_i}}{S}\right) = \frac{e^{z_i} \cdot S - e^{z_i} \cdot e^{z_i}}{S^2} = \frac{e^{z_i}(S - e^{z_i})}{S^2}$$

This simplifies to:

$$\frac{\partial s_i}{\partial z_i} = \frac{e^{z_i}}{S} \cdot \frac{S - e^{z_i}}{S} = s_i(1 - s_i)$$

Case 2: $i \neq j$
Here, the numerator e^{z_i} doesn't depend on z_j, but the denominator S does:

$$\frac{\partial s_i}{\partial z_j} = \frac{0 \cdot S - e^{z_i} \cdot e^{z_j}}{S^2} = -\frac{e^{z_i} e^{z_j}}{S^2} = -s_i s_j$$

Final Result:

$$\frac{\partial s_i}{\partial z_j} = \begin{cases} s_i(1 - s_i) & \text{if } i = j \\ -s_i s_j & \text{if } i \neq j \end{cases}$$

This can be written compactly as:

$$\frac{\partial s_i}{\partial z_j} = s_i(\delta_{ij} - s_j)$$

where δ_{ij} is the Kronecker delta.

2 Question 2

We need to find weights and biases for AND and OR functions using $f(x) = 1$ if $w^T x + b \geq 0$, else 0.

AND Function:

For AND, we need the output to be 1 only when both inputs are 1. Let's try $w = [1, 1]$ and $b = -1.5$.

Checking all cases:

$$(0,0): \quad 1 \cdot 0 + 1 \cdot 0 - 1.5 = -1.5 < 0 \rightarrow 0 \tag{1}$$
$$(0,1): \quad 1 \cdot 0 + 1 \cdot 1 - 1.5 = -0.5 < 0 \rightarrow 0 \tag{2}$$
$$(1,0): \quad 1 \cdot 1 + 1 \cdot 0 - 1.5 = -0.5 < 0 \rightarrow 0 \tag{3}$$
$$(1,1): \quad 1 \cdot 1 + 1 \cdot 1 - 1.5 = 0.5 \geq 0 \rightarrow 1 \tag{4}$$

So $w_{AND} = [1, 1]$ and $b_{AND} = -1.5$.

OR Function:

For OR, we need the output to be 1 when at least one input is 1. Let's try $w = [1, 1]$ and $b = -0.5$.

Checking all cases:

$$(0,0): \quad 1 \cdot 0 + 1 \cdot 0 - 0.5 = -0.5 < 0 \rightarrow 0 \tag{5}$$
$$(0,1): \quad 1 \cdot 0 + 1 \cdot 1 - 0.5 = 0.5 \geq 0 \rightarrow 1 \tag{6}$$
$$(1,0): \quad 1 \cdot 1 + 1 \cdot 0 - 0.5 = 0.5 \geq 0 \rightarrow 1 \tag{7}$$
$$(1,1): \quad 1 \cdot 1 + 1 \cdot 1 - 0.5 = 1.5 \geq 0 \rightarrow 1 \tag{8}$$

So $w_{OR} = [1, 1]$ and $b_{OR} = -0.5$.

3 Question 3

We need to prove that XOR cannot be represented by a linear threshold function.

The XOR function outputs:

$$(0,0) \to 0 \tag{1}$$
$$(0,1) \to 1 \tag{2}$$
$$(1,0) \to 1 \tag{3}$$
$$(1,1) \to 0 \tag{4}$$

For a linear threshold function to work, we need a line $w_1x_1 + w_2x_2 + b = 0$ that separates the positive cases from negative cases.

Looking at this geometrically, we need points $(0,1)$ and $(1,0)$ on the positive side and points $(0,0)$ and $(1,1)$ on the negative side. But this is impossible since $(0,0)$ and $(1,1)$ are diagonally opposite corners, and $(0,1)$ and $(1,0)$ are the other two corners of the unit square. No single line can separate these diagonal pairs.

Formal proof by contradiction:

Assume there exist w_1, w_2, b such that:

$$w_1 \cdot 0 + w_2 \cdot 1 + b \geq 0 \quad (\text{for } (0,1) \to 1) \tag{5}$$
$$w_1 \cdot 1 + w_2 \cdot 0 + b \geq 0 \quad (\text{for } (1,0) \to 1) \tag{6}$$
$$w_1 \cdot 0 + w_2 \cdot 0 + b < 0 \quad (\text{for } (0,0) \to 0) \tag{7}$$
$$w_1 \cdot 1 + w_2 \cdot 1 + b < 0 \quad (\text{for } (1,1) \to 0) \tag{8}$$

This gives us:

$$w_2 + b \geq 0 \tag{1}$$
$$w_1 + b \geq 0 \tag{2}$$
$$b < 0 \tag{3}$$
$$w_1 + w_2 + b < 0 \tag{4}$$

From (1) and (3): $w_2 \geq -b > 0$

From (2) and (3): $w_1 \geq -b > 0$

But then $w_1 + w_2 + b \geq -b + (-b) + b = -b > 0$, which contradicts (4).

Therefore, XOR cannot be represented using a linear threshold function.

Assignment 1 Paper Review

Name: Burak Aksu
GT Email: Baksu3@gatech.edu

Review:

I chose to review "Understanding Deep Learning Requires Rethinking Generalization" by Zhang et al. This paper fundamentally challenges how we think about why neural networks generalize well in practice. The authors demonstrate through systematic experiments that state-of-the-art convolutional networks can perfectly memorize training data with completely random labels, achieving zero training error even when true images are replaced with random noise. This finding undermines traditional statistical learning theory explanations for generalization based on model complexity measures like VC dimension or Rademacher complexity.

The main contribution lies in showing that deep networks have sufficient capacity to memorize entire datasets, yet they still generalize well on real data with meaningful patterns. This creates a paradox for classical theory, which suggests that models capable of memorizing random data should overfit catastrophically. The experiments are methodical and convincing, testing various architectures on CIFAR-10 and ImageNet while manipulating label corruption levels. The authors also show that explicit regularization techniques like dropout and weight decay don't prevent this memorization ability, though they can improve generalization on real data. The theoretical construction proving that simple two-layer networks can express any labeling with sufficient parameters provides formal backing for their empirical observations.

The paper's strength in dismantling existing theoretical frameworks is also its main limitation. While it effectively demonstrates that traditional capacity-based explanations are insufficient, it offers limited positive theory about what actually drives generalization. The discussion of implicit regularization through SGD provides some direction, but feels incomplete compared to the thoroughness of their negative results. My personal takeaway is that this work represents a crucial paradigm shift in deep learning theory. It forced the field to move beyond simplistic overfitting narratives and consider more nuanced explanations involving optimization dynamics and inductive biases. The paper opened important questions about why neural networks find generalizable solutions when so many memorization solutions exist, questions that continue driving theoretical research today.

Question 1:

If neural networks can “memorize” the data, which is the only thing they can do for random label assignments that don’t correlate with patterns in the data, why do you think neural networks learn more meaningful, generalizable representations when there are meaningful patterns in the data?

Answer:

From a search perspective, the traditional view treats learning as optimization over a weight space to minimize training error while controlling model complexity. This paper reveals a fundamental problem with this framing. When networks can perfectly fit random labels, the search space contains countless solutions that achieve zero training loss but differ dramatically in generalization ability. The memorization experiments show that the search problem isn't simply about finding any global minimum of training loss, but about finding the right kind of minimum among many possible ones.

The paper suggests that the search process itself, particularly through SGD, acts as an implicit bias toward solutions that generalize. This reframes learning from a capacity control problem to an optimization path problem. Rather than restricting the search space through explicit regularization, the algorithm's dynamics guide the search toward beneficial regions of the loss landscape. SGD appears to have strong preferences for certain types of solutions over others, even when multiple solutions achieve identical training performance. This perspective implies that understanding generalization requires analyzing not just what solutions exist mathematically, but which ones SGD actually discovers in practice.

The search lens also highlights why architecture matters beyond just representational capacity. Different architectures create different optimization landscapes that may bias the search process in distinct ways. The experiments showing that some architectures generalize better than others, despite similar memorization capabilities, suggest that the interaction between search algorithm and parameter space geometry plays a crucial role. This view emphasizes that successful deep learning emerges from the interplay between the search space defined by architecture and the search process implemented by the optimizer, rather than from either component alone.

Question 2:

How does this finding align or not align with your understanding of machine learning and generalization?

Answer:

This paper reveals that architectures themselves have enormous representational power even with fixed weight determination methods like SGD. The experiments show that standard CNN architectures can represent any labeling of training data when weights are learned through SGD, demonstrating universal finite-sample expressivity in practice. The theoretical construction proving that two-layer networks with sufficient parameters can express arbitrary functions on finite samples formalizes this representational capacity. However, the method for determining weights absolutely matters for what actually gets represented during learning.

SGD doesn't randomly sample from all possible representations that an architecture could theoretically express. Instead, it systematically biases learning toward certain types of solutions based on initialization, gradient flow dynamics, and the optimization landscape. When presented with real data containing meaningful patterns, SGD consistently finds representations that capture these patterns rather than memorizing arbitrary associations. This suggests a strong interaction between the optimization process and the underlying data structure that guides learning toward meaningful solutions.

The paper implies these components have fundamentally different but complementary roles rather than equivalent representational power. Architecture provides the raw computational substrate and defines what functions could possibly be represented, while the optimization method determines which of these many possible representations actually gets learned. This interaction explains why some architectures work better for certain tasks, not necessarily because they have fundamentally different representational capacity, but because they combine with SGD dynamics to create stronger inductive biases toward appropriate solution types for those domains. The representational power question thus becomes less about what can be represented in principle and more about what will be represented through the specific learning process used in practice.

Assignment 1 Writeup

Name: Burak Aksu
GT Email: Baksu3@gatech.edu

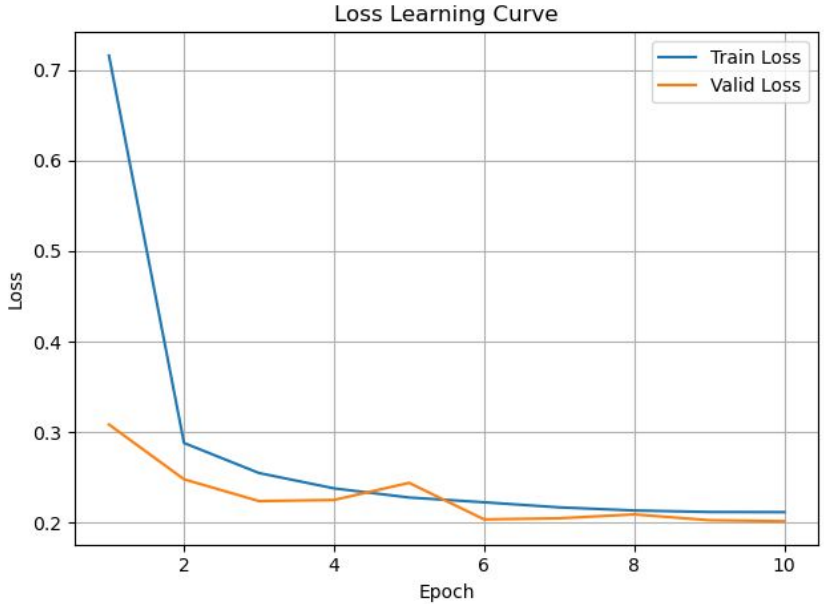
Two-Layer Neural Network

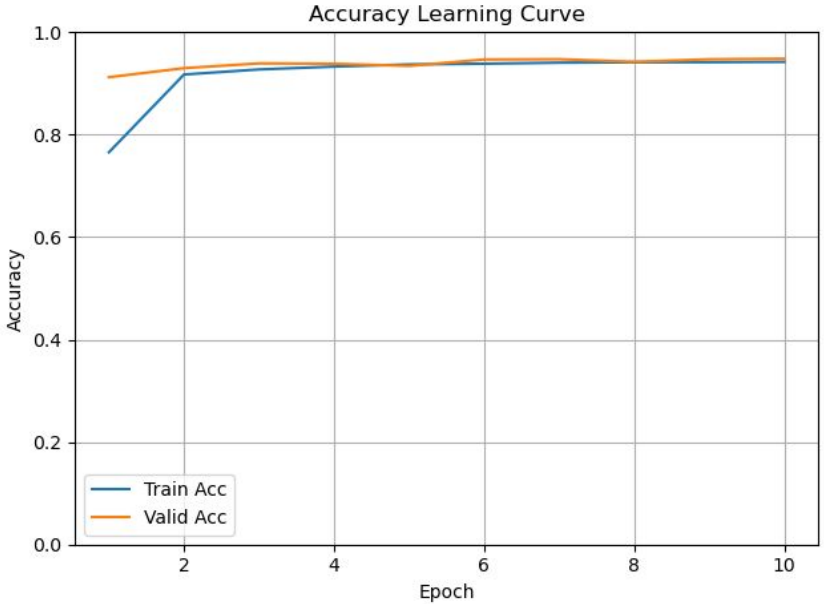
1. Learning Rates

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

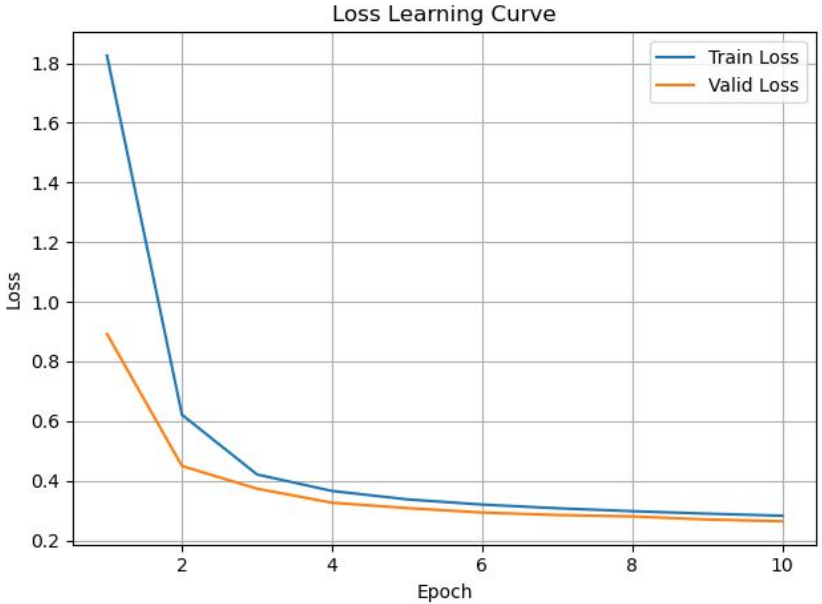
	lr=1	lr=1e-1	lr=5e-2	lr=1e-2
Training Accuracy	94.23%	92.10%	90.86%	72.98%
Test Accuracy	94.64%	92.72%	91.37%	75.35%

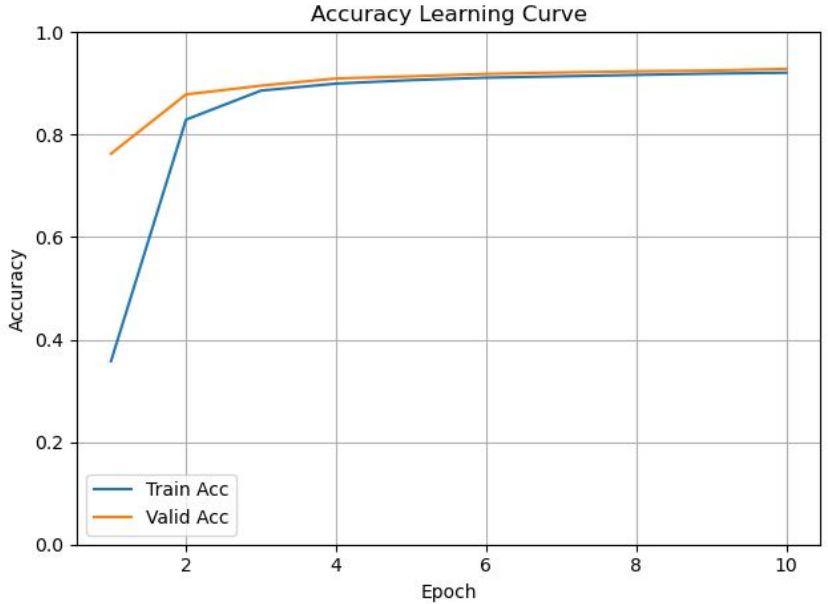
1. Learning Curve (Learning Rate 1.0)



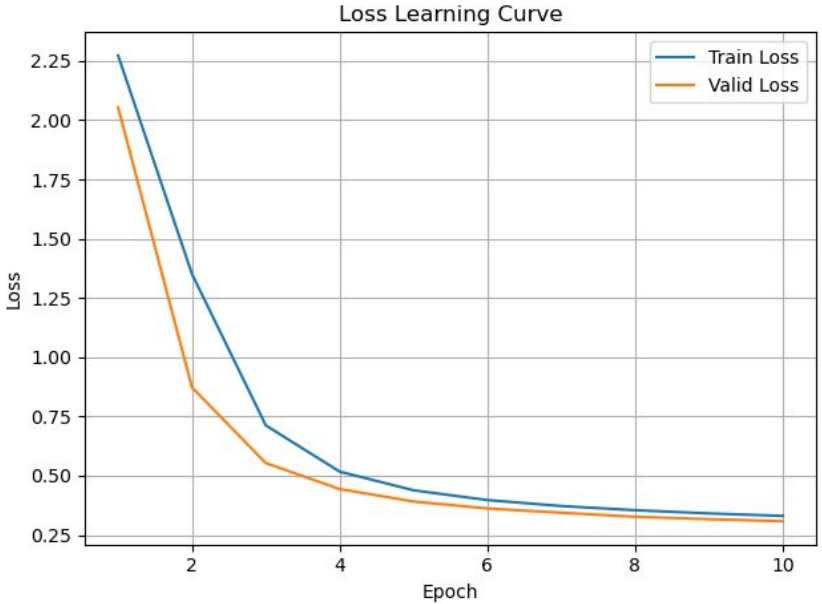


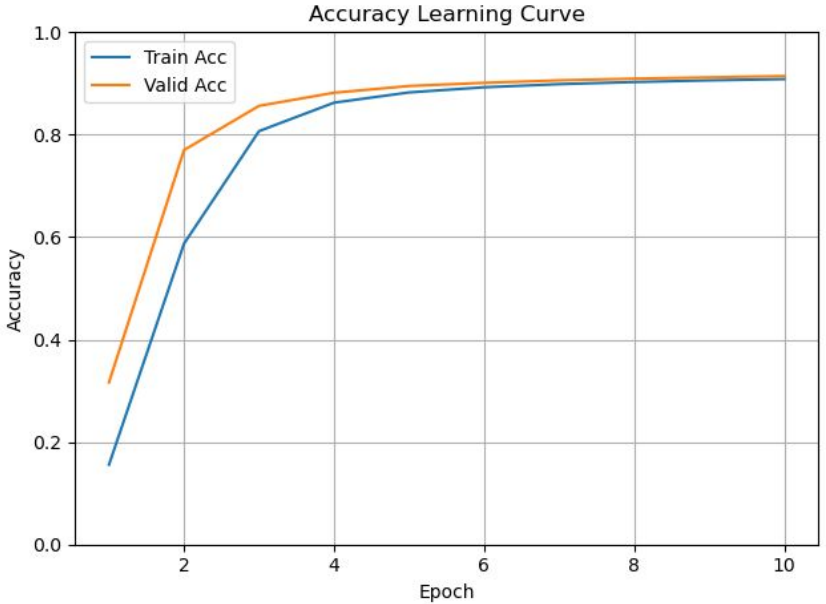
1. Learning Curve (Learning Rate 1e-1)



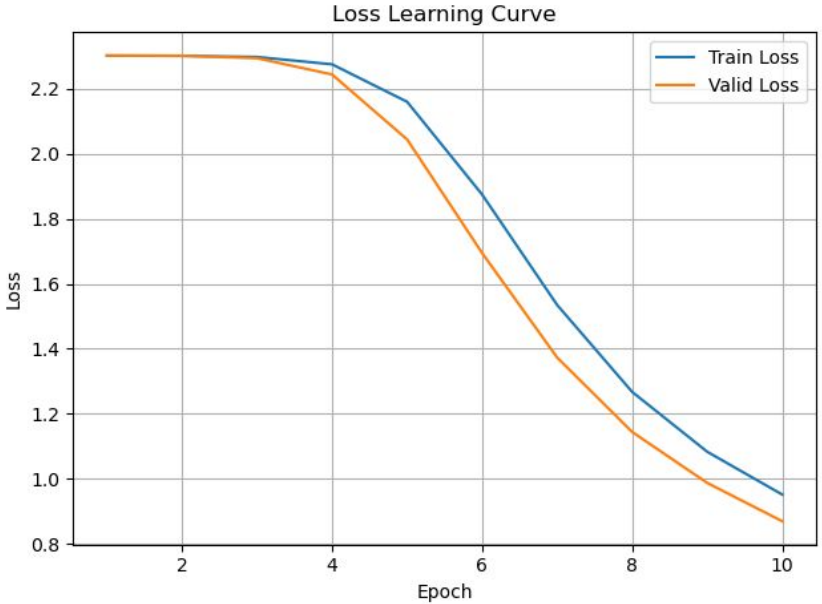


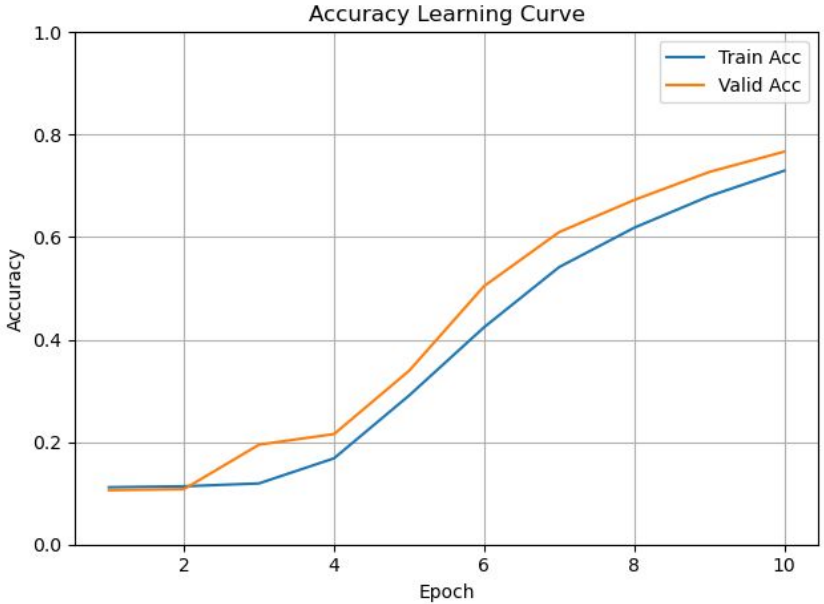
1. Learning Curve (Learning Rate 5e-2)





1. Learning Curve (Learning Rate 1e-2)





1. Learning Rates

The learning rate experiments reveal fundamental insights about the optimization dynamics in neural network training. Looking across the different learning rates from 1e-2 to 1.0, we observe a clear pattern where moderate to high learning rates (5e-1, 1e-1, and 1.0) achieve significantly better performance than the very low learning rate of 1e-2. The 1e-2 learning rate produces the worst results, with training and test accuracies struggling to exceed 75%. This occurs because the learning rate is too conservative for the optimization landscape of this particular task. When gradient steps are too small, the model gets trapped in poor local minima or takes an extremely long time to escape suboptimal regions of the loss surface. The learning curves show this clearly, with both loss and accuracy improving very slowly and plateauing at mediocre levels. Essentially, the model lacks the momentum needed to navigate the complex, high-dimensional parameter space effectively.

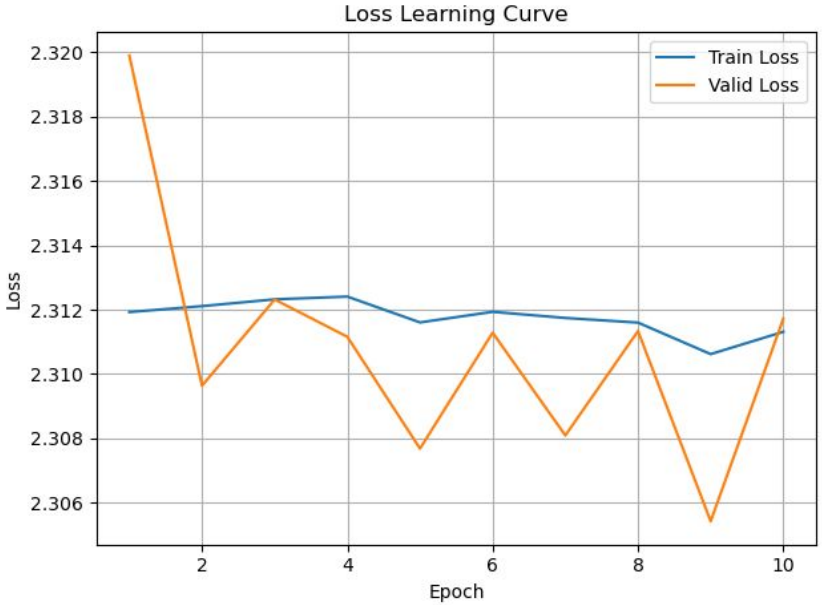
The dramatic improvement we see when moving to 5e-1 demonstrates a sweet spot where the learning rate provides sufficient step size to escape local minima while maintaining stability. Both training and test accuracies jump to above 90%, indicating that the model can now find much better solutions in the same number of epochs. The learning curves show faster convergence and higher final performance, suggesting the optimization process is efficiently navigating toward better regions of the loss landscape. Interestingly, the very high learning rates of 1e-1 and 1.0 continue to perform well, which might seem counterintuitive since conventional wisdom suggests that high learning rates cause instability. However, for this particular network architecture and dataset, these aggressive learning rates appear to provide beneficial exploration that helps the model escape poor local optima. The fact that test accuracy remains high indicates that the learned representations are still generalizable despite the aggressive optimization. The key insight here relates to the bias-variance tradeoff in optimization. Very low learning rates create high bias by constraining the model to explore only a limited region around the initialization, preventing discovery of better solutions. Higher learning rates introduce more variance in the optimization path but enable broader exploration of the parameter space, ultimately finding superior solutions that generalize well.

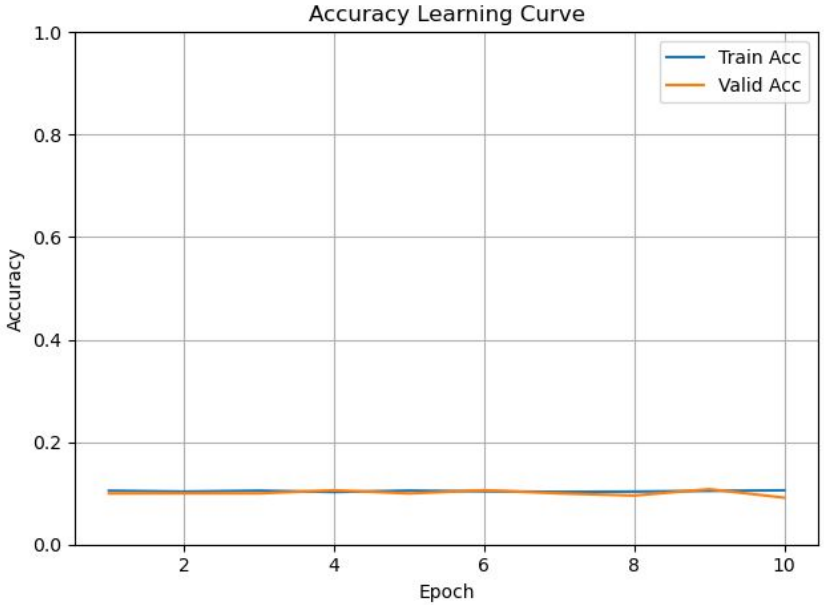
2. Regularization

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

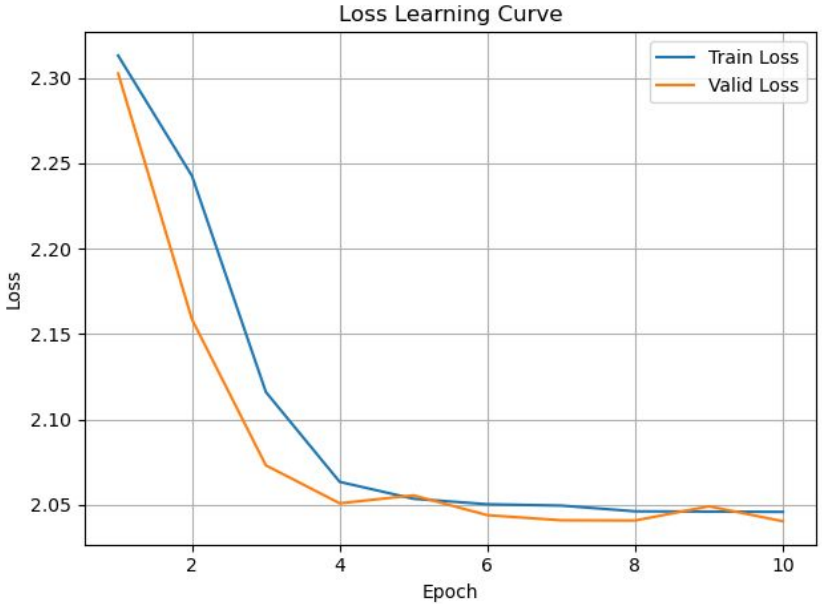
	alpha=1	alpha=1e-1	alpha=1e-2	alpha=1e-3	alpha=1e-4
Training Accuracy	10.59%	33.94%	88.46%	92.18%	93.01%
Validation Accuracy	10.81%	38.77%	89.43%	92.53%	93.47%
Test Accuracy	10.28%	38.77%	89.18%	92.43%	93.29%

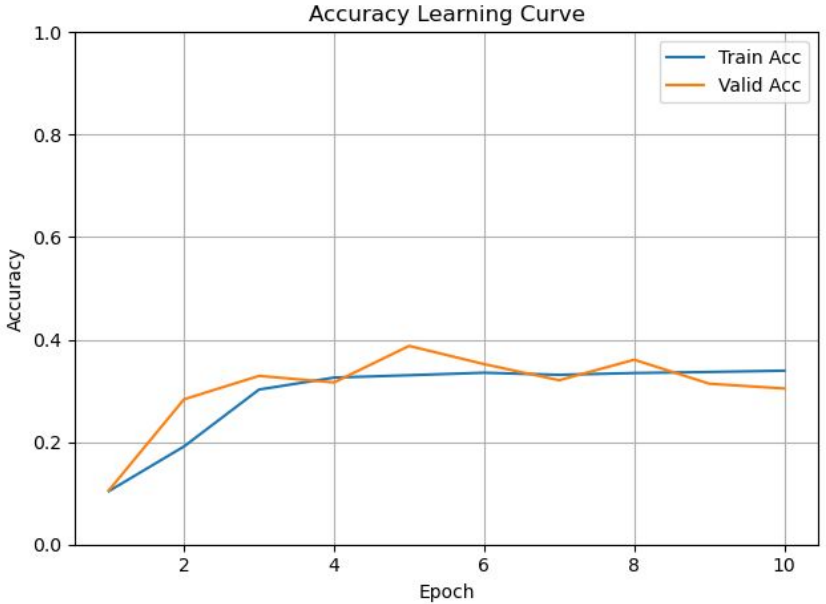
2. Regularization (Alpha 1.0)



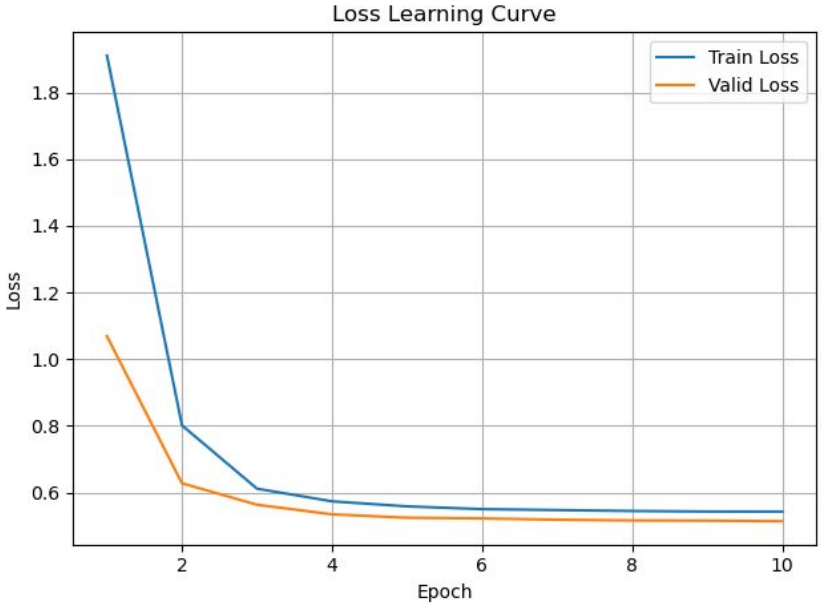


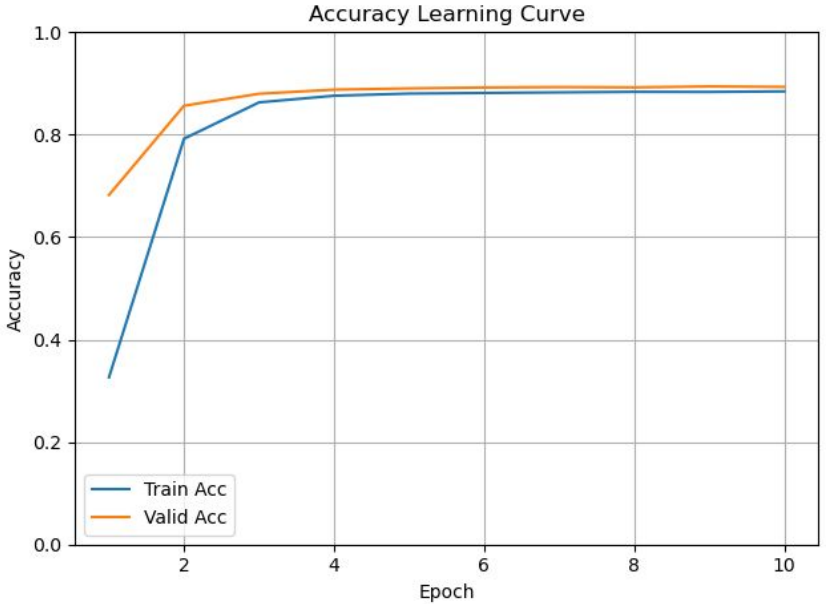
2. Regularization (Alpha 1e-1)



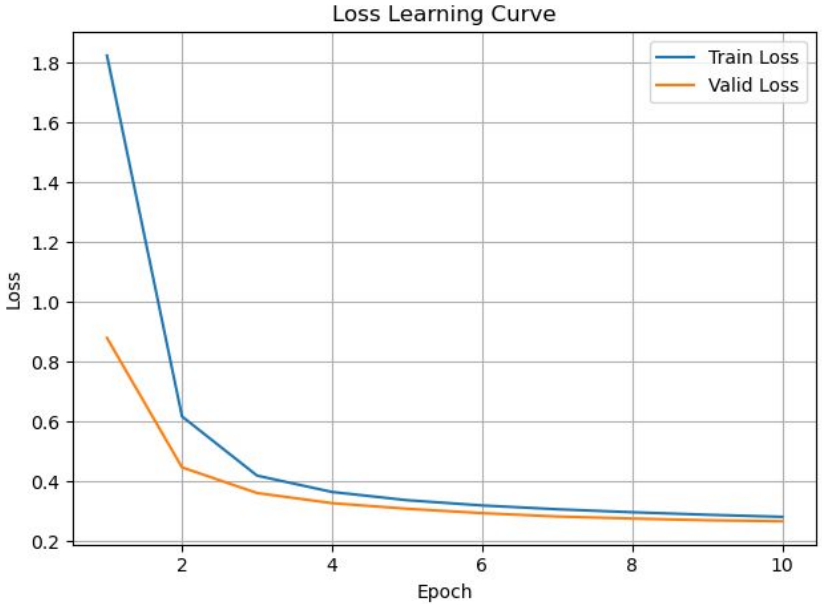


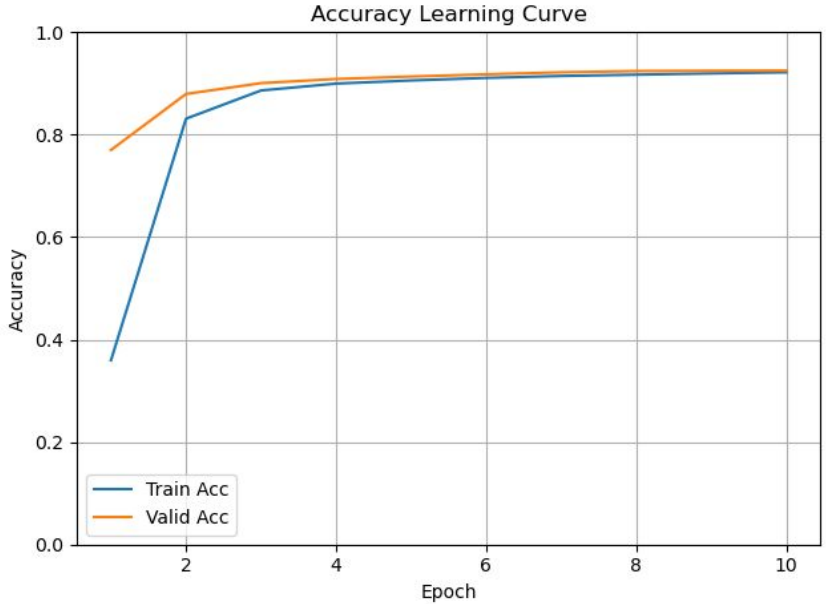
2. Regularization (Alpha 1e-2)



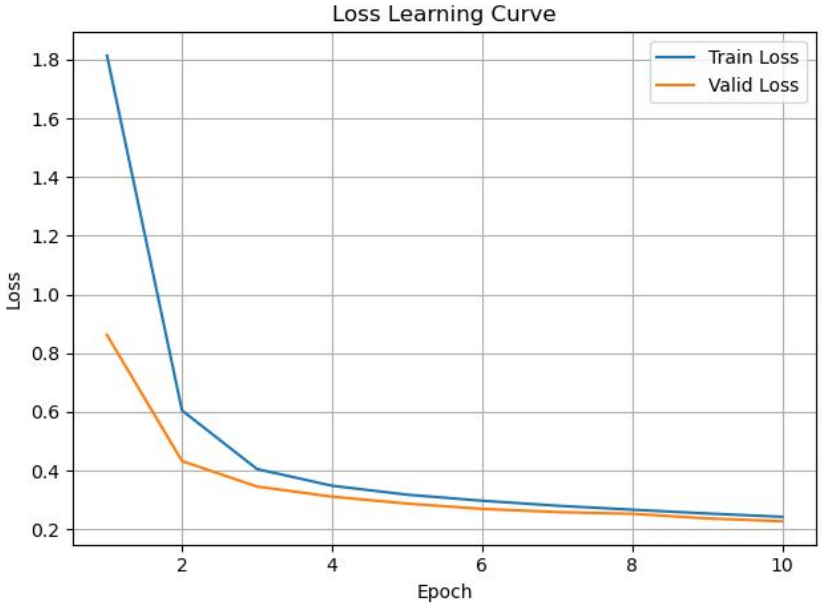


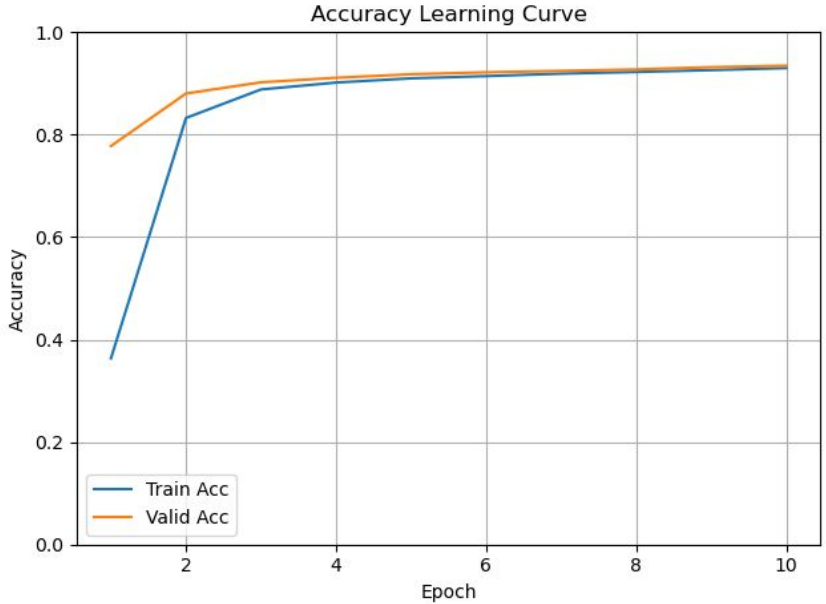
2. Regularization (Alpha 1e-3)





2. Regularization (Alpha 1e-4)





2. Regularization

The regularization experiments reveal a striking inverse relationship between regularization strength and model performance, demonstrating the fundamental tension between preventing overfitting and allowing sufficient model flexibility. As the regularization coefficient (alpha) decreases from 1.0 to 1e-4, we observe a dramatic improvement in all performance metrics. At the highest regularization level (alpha=1.0), the model achieves only around 10% accuracy across training, validation, and test sets, which is essentially random performance for what appears to be a 10-class classification problem. This occurs because excessive regularization constrains the model weights so severely that the network cannot learn meaningful patterns from the data. The L2 penalty term dominates the loss function, forcing weights toward zero and preventing the model from developing the complex representations necessary for effective classification. The learning curves show that both training and validation losses plateau at high values with minimal improvement, indicating the model lacks the capacity to fit even the training data properly.

As we reduce the regularization strength to alpha=1e-1, performance improves dramatically but remains suboptimal at around 38% accuracy. The model can now learn some patterns, but the regularization is still too aggressive for this particular task and dataset. The sweet spot emerges around alpha=1e-2 to alpha=1e-4, where we see excellent performance exceeding 90% accuracy on all metrics. This range represents the optimal balance where regularization provides just enough constraint to prevent overfitting while allowing the model sufficient flexibility to learn complex decision boundaries. The learning curves in this range show healthy convergence with both training and validation metrics improving steadily and converging to similar values, indicating good generalization without overfitting. The key insight here relates to the bias-variance tradeoff in regularization. Strong regularization introduces high bias by severely limiting model complexity, preventing the discovery of optimal solutions even when sufficient data exists. Weak regularization allows the model to explore the full parameter space and find representations that capture the underlying data distribution effectively, leading to superior generalization performance when the model architecture and dataset are well-matched.

3. Hyper-parameter Tuning

	Default	**Exp1:** lr=1.0 reg=0.0001	**Exp2:** +hidden_size=256	**Exp3:** +batch_size=32	**Exp4:** +momentum=0.95	**Exp5:** +epochs=15
Training Accuracy	92.11%	97.97%	97.54%	97.92%	98.03%	98.42%
Validation Accuracy	92.54%	97.14%	96.88%	97.02%	97.16%	97.58%
Test Accuracy	92.39%	97.19%	97.08%	97.18%	97.32%	97.50%
Improvements	+0.00%	+4.80%	+4.69%	+4.79%	+4.93%	+5.11%

3. Hyper-parameter Tuning (Explanation)

The dramatic improvement from 92.39% to 97.50% test accuracy stems from addressing fundamental bottlenecks in the optimization process that were limiting the baseline model's performance. The aggressive learning rate of 1.0 represents the most critical change, enabling the model to traverse the loss landscape efficiently rather than getting trapped in suboptimal regions. MNIST's relatively smooth optimization surface can handle this high learning rate without the instability typically associated with such aggressive settings in more complex tasks. The combination with minimal regularization (0.0001) creates an environment where the model can fully exploit the large, clean MNIST dataset without artificial constraints that were previously preventing it from reaching its representational potential.

The supporting hyperparameters work synergistically to maintain stability while maximizing learning efficiency. The smaller batch size of 32 introduces beneficial noise into the gradient estimates, which paradoxically improves generalization by preventing overfitting to specific batch compositions and helping the optimizer explore different regions of the parameter space. This stochastic effect is particularly valuable when paired with the high learning rate, as it prevents the model from converging too quickly to the first reasonable solution it encounters. The increased momentum of 0.95 acts as a stabilizing force, smoothing out the potentially erratic updates from the high learning rate and small batch size combination. This creates a virtuous cycle where rapid learning is maintained without sacrificing convergence stability. The extended training period of 15 epochs allows this optimized configuration to fully converge, extracting maximum performance from the architecture without hitting overfitting issues that might occur with more complex datasets. The 5.11% improvement demonstrates that the baseline configuration was significantly underutilizing the model's capacity through overly conservative hyperparameter choices.