CS7643 Spring '24: Assignment 1

Tyler Burki
tburki3@gatech.edu

February 9, 2024

1 Theory Problem Set

1.1 Softmax gradient derivation

Let s($\mathbf{z}$) be the function whose output for some input vector $\mathbf{z}$ is a vector such that the i^{th} element, s_i, is equivalent to

$$\frac{e^{z_i}}{\sum_k e^{z_k}} \tag{1}$$

where k is the number of elements in $\mathbf{z}$.

We will derive $\frac{ds}{dz}$ by first using the quotient rule (Quotient rule, 2020) to derive $\frac{ds_i}{dz_j}$ from the equation above:

$$\frac{ds_i}{dz_j} = \frac{d}{dz_j}\frac{e^{z_i}}{\sum_k e^{z_k}} = \frac{(\frac{d}{dz_j}e^{z_i})\sum_k e^{z_k} - e^{z_i}(\frac{d}{dz_j}\sum_k e^{z_k})}{(\sum_k e^{z_k})^2} \tag{2}$$

First we consider the case where $i = j$ and simplify the equation:

$$\frac{(\frac{d}{dz_j}e^{z_i})\sum_k e^{z_k} - e^{z_i}(\frac{d}{dz_j}\sum_k e^{z_k})}{(\sum_k e^{z_k})^2} = \frac{e^{z_i}\cdot\sum_k e^{z_k} - e^{z_i}\cdot e^{z_i}}{(\sum_k e^{z_k})^2} \tag{3}$$

Next we can factor out e^{z_i} from the numerator and split the denominator by turning the problem into a multiplication of fractions:

$$\frac{e^{z_i}\cdot\sum_k e^{z_k} - e^{z_i}\cdot e^{z_i}}{(\sum_k e^{z_k})^2} = \frac{e^{z_i}(\sum_k e^{z_k} - e^{z_i})}{(\sum_k e^{z_k})^2} = \frac{e^{z_i}}{\sum_k e^{z_k}}\cdot\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}} \tag{4}$$

The term on the left is just s_i, and the term on the right can be simplified as follows:

$$\frac{e^{z_i}}{\sum_k e^{z_k}}\cdot\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}} = (s_i)(\frac{\sum_k e^{z_k}}{\sum_k e^{z_k}} - \frac{e^{z_i}}{\sum_k e^{z_k}}) = (s_i)(1 - s_i) \tag{5}$$

Next we consider the case where $i \neq j$ and follow a similar procedure with respect to z_j:

$$\frac{(\frac{d}{dz_j}e^{z_i})\sum_k e^{z_k} - e^{z_i}(\frac{d}{dz_j}\sum_k e^{z_k})}{(\sum_k e^{z_k})^2} = \frac{0 \cdot \sum_k e^{z_k} - e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} = \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} \tag{6}$$

We can then simplify in the same manner as the previous case:

$$\frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} = (\frac{-e^{z_i}}{\sum_k e^{z_k}})(\frac{e^{z_j}}{\sum_k e^{z_k}}) = (-s_i)(s_j) \tag{7}$$

With these two cases, we can now construct the Jacobian J for s($\mathbf{z}$) where i, j are the row and column indices, respectively:

$$J(s(z)) = \begin{bmatrix} s_1(1-s_1) & \cdots & -s_1s_j & \cdots & -s_1s_k \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ -s_is_1 & \cdots & s_i(1-s_j) & \cdots & -s_is_k \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ -s_ks_1 & \cdots & -s_ks_j & \cdots & s_k(1-s_k) \end{bmatrix} \tag{8}$$

We see that J ϵ $\mathbb{R}^{k \times k}$ and notice that when i = j (the diagonal) the value of J(i, j) = $s_i(1-s_j)$ while all other entries are of the form $-s_is_j$. It follows that the Jacobian can be calculated for any i, j pair as such:

$$J(i,\, j) = \begin{cases} s_i(1-s_j) & \text{i = j} \\ -s_is_j & \text{otherwise} \end{cases} \tag{9}$$

1.2 AND, OR with single linear threshold neuron

To find a suitable linear threshold neuron to represent the **AND** and **OR** functions we must determine a vector **w** and scalar b such that

$$f_{AND}(\mathbf{x}) = f_{OR}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T x + b \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

follows the respective function's truth table for all [x_1, x_2] ϵ {0, 1}.

AND Design We must find values for **w** and b that satisfy (10) and the truth table below.

x_1	x_2	$f_{AND}(x)$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table for AND

$\mathbf{w} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$ and b = -1 is one such solution. To validate, we can iterate through the inputs from the truth table and verify that the correct output is given for each:

$$[.5, .5] \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1 = -1 \tag{11}$$

$$[.5, .5] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 1 = -.5 \tag{12}$$

$$[.5, .5] \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 1 = -.5 \tag{13}$$

$$[.5, .5] \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 1 = 0 \tag{14}$$

Since our activation function returns 1 iff $\mathbf{w}^T\mathbf{x} + \text{b} \geq 0$, we see that only the final input, where $x_1 = x_2 = 1$, will cause the neuron to activate and therefore the solution has been confirmed.

OR Design The **OR** function possesses the same constraints as **AND** with respect to its own truth table.

x_1	x_2	$f_{OR}(x)$
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth table for OR

Let us verify the solution $\mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and b = -1 in the same manner as above:

$$[1, 1] \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1 = -1 \tag{15}$$

$$[1, 1] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 1 = 0 \tag{16}$$

$$[1, 1] \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 1 = 0 \tag{17}$$

$$[1,1] \cdot \begin{bmatrix}1\\1\end{bmatrix} - 1 = 1 \tag{18}$$

Here we see $\mathbf{w}^T\mathbf{x}$ + b < 0 only when $x_1 = x_2 = 0$, meaning that it is the only case where the neuron would not be activated. This result matches the truth table.

Solutions **AND**: $\mathbf{w} = \begin{bmatrix}.5\\.5\end{bmatrix}$, b = -1 **OR**: $\mathbf{w} = \begin{bmatrix}1\\1\end{bmatrix}$, b = -1

1.3 Proof XOR cannot be represented by 1.2

Let $\mathbf{w} = \begin{bmatrix}w_1\\w_2\end{bmatrix}$ and b be the weights and bias, respectively, of a linear model describing the XOR function such that

$$f_{XOR}(\mathbf{x}) = \begin{cases}1 & \mathbf{w}^T x + b \geq 0\\ 0 & \text{otherwise}\end{cases} \tag{19}$$

We will show that such a model is not possible by contradiction. *Table 3* is the truth table for the function.

x_1	x_2	$f_{XOR}(x)$
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Truth table for XOR

We begin by deriving constraints for **w** and b from the truth table:

$$\mathbf{w}^T \begin{bmatrix}0\\0\end{bmatrix} + b < 0 \Rightarrow b < 0 \tag{20}$$

$$\mathbf{w}^T \begin{bmatrix}0\\1\end{bmatrix} + b \geq 0 \Rightarrow w_2 \geq -b \tag{21}$$

$$\mathbf{w}^T \begin{bmatrix}1\\0\end{bmatrix} + b \geq 0 \Rightarrow w_1 \geq -b \tag{22}$$

$$\mathbf{w}^T \begin{bmatrix}1\\1\end{bmatrix} + b < 0 \Rightarrow w_1 + w_2 < -b \tag{23}$$

From these constraints we know that b must be negative, and that both w_1 and w_2 are positive numbers larger than $-b$. It follows that the last inequality, $w_1 + w_2 < -b$, does not hold. Since the minimum value w_1 and w_2 can attain is $-b$, we can substitute this value for both w_1 and w_2 into the final inequality to get $-b - b < -b \Rightarrow -2b < -b$. Because $b < 0$, $-2b$ and $-b$ will always produce positive numbers and the left side of the inequality will always have twice the magnitude of the right side and therefore cannot satisfy the constraint.

2 Paper Review

The following section will review *Understanding Deep Learning Requires Rethinking Generalization* by Zhang et al. from 2017.

2.1 Review

The authors assess contemporary methods for reduction of generalization error in deep neural networks and discover that many do not have a significant impact on a model's ability to generalize well. To show this, they perform a number of experiments that alter the training labels or modify the data in some fashion in order to measure the effect on generalization error. For example, one experiment randomizes labels with probability p while another randomizes all labels indiscriminately. Other experiments shift pixels in image data or replace the data altogether with Guassian noise.

In spite of these attempts to increase generalization error, the authors found that their models, with hyperparameters remaining constant between experiments, were able to achieve zero training error and similar generalization error to their counterparts. As highlighted in the paper, "Deep neural networks easily fit random labels" (Zhang et al., 2017, p.2). The question then becomes: what is truly responsible for reduction of generalization error in deep neural networks?

The authors find that regularization–explicit or implicit–is neither a necessary nor sufficient condition for low generalization error. The largest differentiating method for regularization appeared to be *early stopping* which "...could potentially improve generalization performance" (Zhang et al., 2017, p.7).

The models' performance on augmented data sets and random labels leads the authors to discuss the effective capacity and expressiveness of deep neural networks. The authors argue that traditional methods of evaluating model complexity–such as Rademacher complexity and VC-dimension– either produce trivial bounds or cannot be applied to the models they have developed. Furthermore, it was found that training on their corrupted data/labels did not significantly increase the complexity and training times of their models, and that the models were essentially able to "memorize" the training data in order to label it correctly. They conclude that further study of measures for model complexity are necessary.

2.2 Questions

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?

Let us frame the problem as if we were trying to score highly on and academic test. On one hand, we have the list of questions while on the other we have a benevolent agent telling us how "close" our answer is to the correct one. Assume that we are trying to map each question to one of four responses: a, b, c, d. In this scenario, the correct response letter has no association with the content of the question. This parallels the random assignment of labels seen in the paper.

We can begin by guessing a letter for each question, and have the agent tells us how "close" we got to the answer (e.g. if we answer "a" but the label is "c" we might be "two away" from the correct answer). It is not difficult to imagine how feedback from the agent could be leveraged to arrive at the correct answers. From there, the answers can be memorized and regurgitated on the test for a perfect score.

However, if new questions appear on the test we haven't explicitly trained for, our method quickly falls apart. Since we didn't actually learn how to solve the problems, we will (at best) be left randomly guessing the answer or (at worst) following a recipe for disaster (e.g. all questions with the word "the" are answer "c").

When there are patterns in the data, we can learn to represent and group data in a variety of ways and extract features. Compositions of these features can help us better identify ground truth, and because deep neural networks work off of distributed representations, they can help us better generalize to compositions we haven't yet seen. The composition of shapes is a good example of this. A network may learn what horizontal and vertical ellipses are and be able to generalize that a horizontal and vertical ellipse is actually a circle without needing to be explicitly trained on the shape of a circle. Without the features generated by the tranining examples, solving new problems erodes to nothing more than guesswork.

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

Since I started with machine learning only a year or so ago, I find the paper to be a little disheartening. I have come to understand that regularization is central to preventing overfitting in neural networks, but this paper suggests that might not be the case. Such techniques make sense because if weights grow unbounded the lines between features may become blurred and subject to over-representation of specific training examples.

However, I can also understand the case *against* regularization, as I imagine the ratios of the weights would still represent the training data in a similar way, even if they are unbounded. For example, if I have a concept $f(\mathbf{x}) = 5x_1 + x_2$ and my model converges to $\hat{f}(\mathbf{x}) = 5000x_1 + 1000x_2$ I still know that fluctuations in feature x_1 are five times as impactful to the result as the same flucuation in feature x_2.

The authors specifically identifying early stopping as a regularization method with potential benefit also makes sense because we intuitively have a better shot at generalizing well if we don't put too much stock into the training data. In other words, we can learn a little from the data without committing to (over)fitting it perfectly so we should be able to better handle examples we have not yet seen.

While many concepts in machine learning have be around for over half a century, the impression this paper leaves on me is that there is still much more to understand about the field.

3 Coding

We will be running a series of experiments on the two-layer network, observing the results and performing analysis of those results.

3.1 Learning Rates

Testing different learning rates

	$\eta = 1$	$\eta = e^{-1}$	$\eta = 5e^{-2}$	$\eta = e^{-2}$
Training Accuracy	0.9450	0.9431	0.9451	0.9285
Testing Accuracy	0.9471	0.9443	0.9461	0.9301

Table 4: Train and test performance on different learning rates

Learning curves

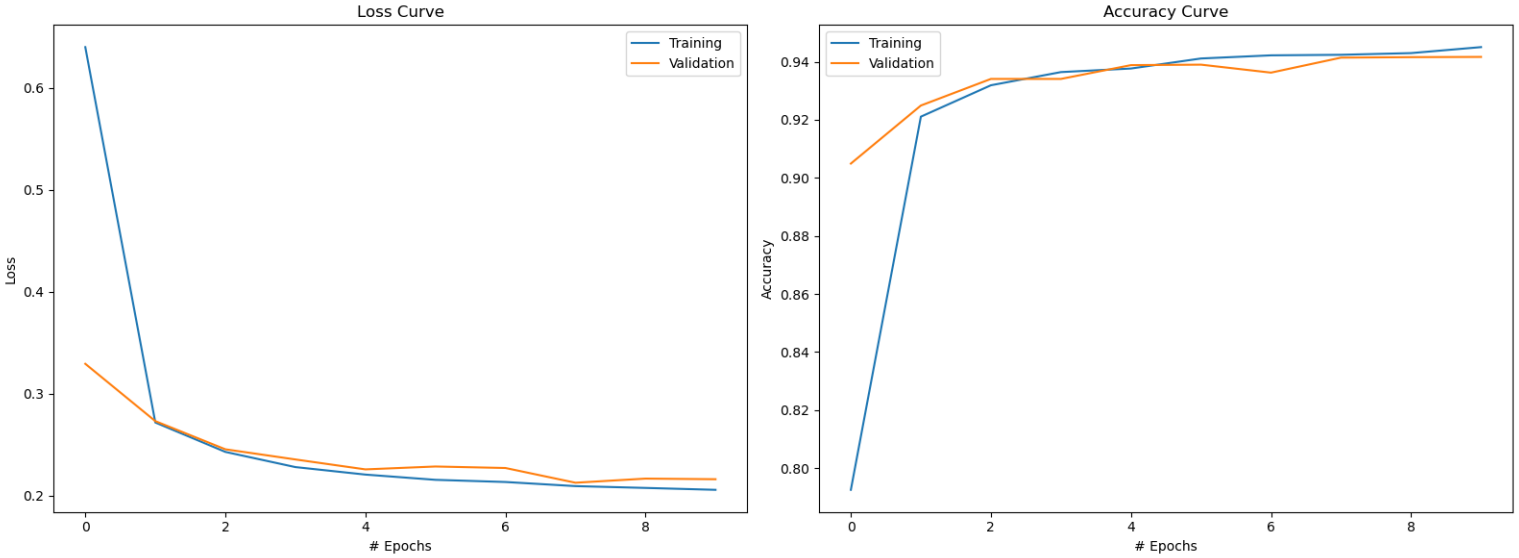

Figure 1: $\eta = 1$

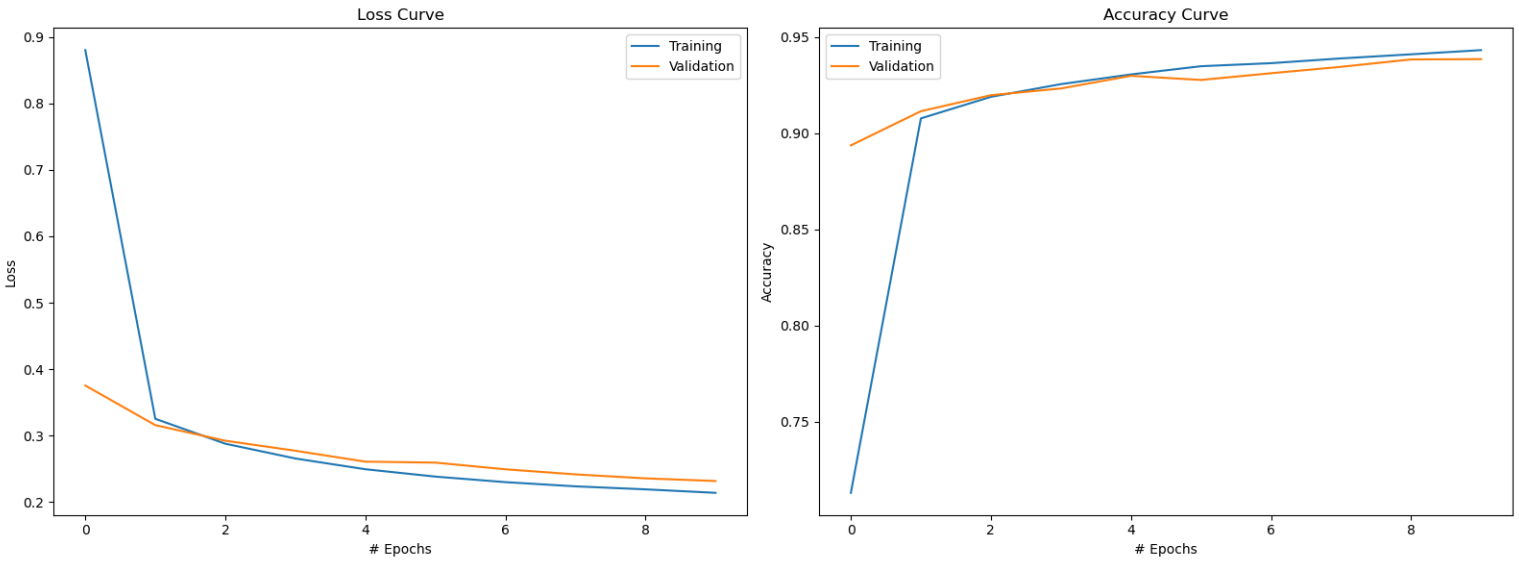

Figure 2: $\eta = e^{-1}$

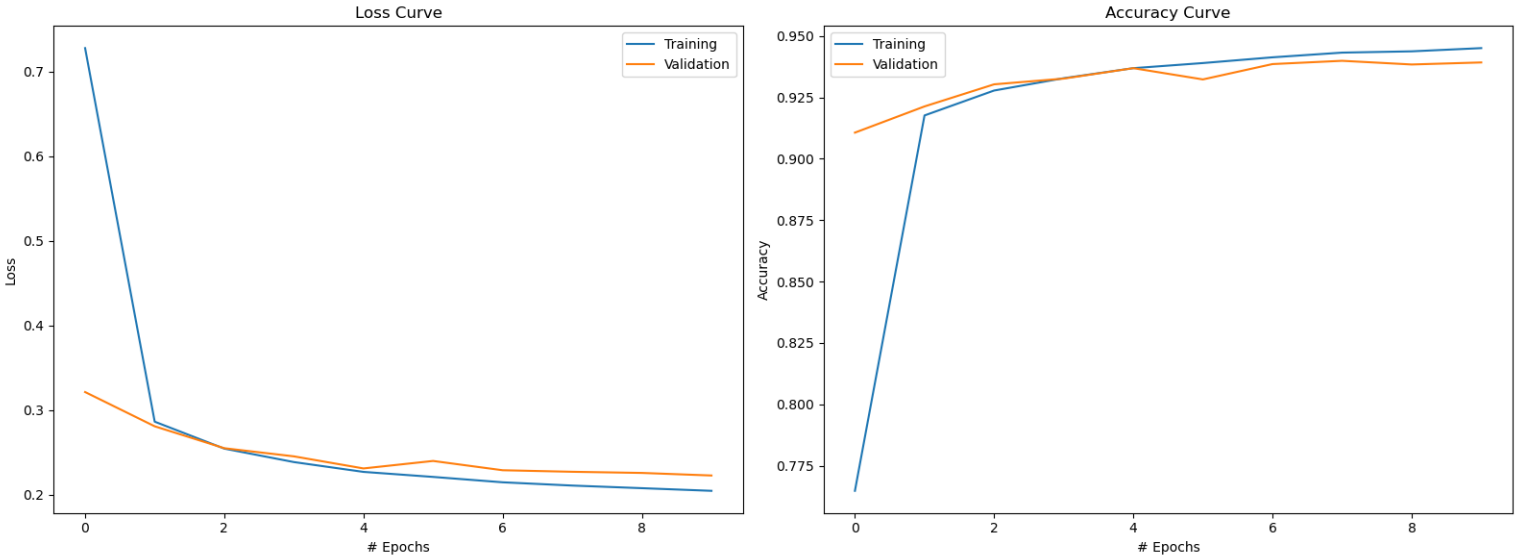

Figure 3: $\eta = 5e^{-2}$

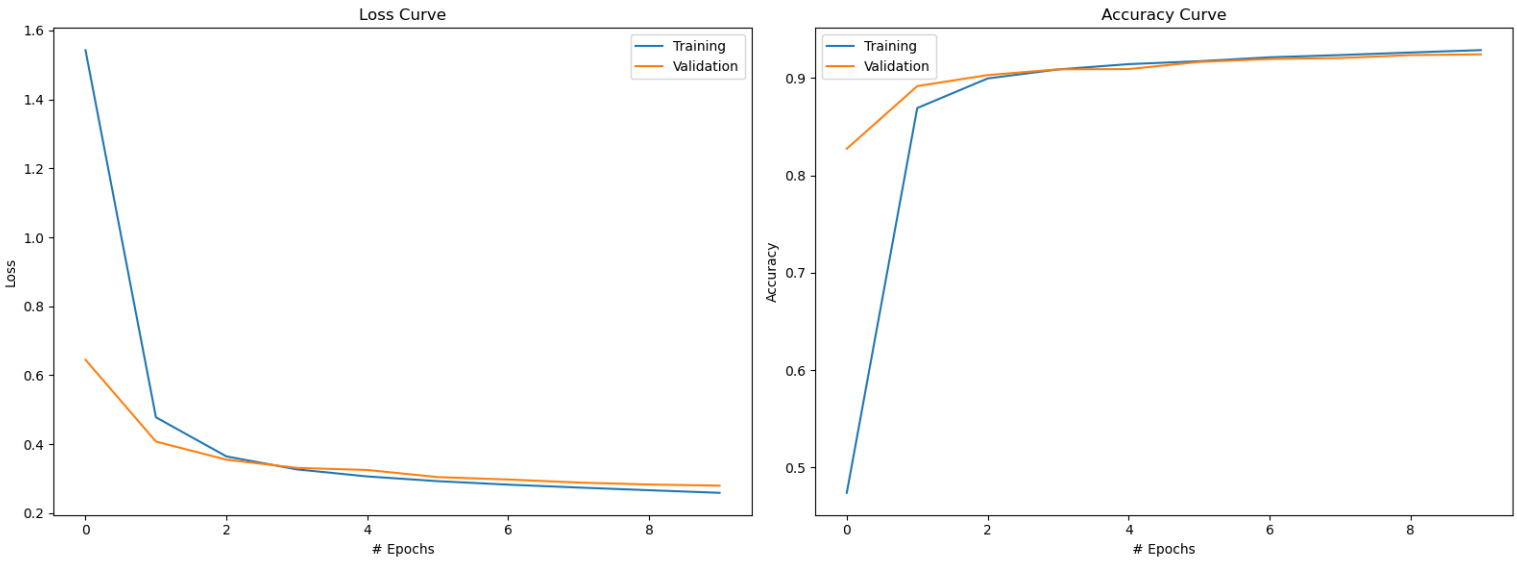

Figure 4: $\eta = e^{-2}$

Analysis

The learning rates of $\eta = 1$, e^{-1} and $5e^{-2}$ all had similar accuracy (within 0.2%) on both training and test sets. For $\eta = e^{-2}$, both sets were less accurate by roughly two percent. This is indicative of a local minimum in the loss function between $\eta = e^{-1}$ (≈ 0.3679) and $\eta = e^{-2}$ (≈ 0.1353). This likely means there is some point in the range between those values that is able to move gradient descent out of worse minimum into the better one that larger rates find.

The previous point can be reinforced by examining the average rates of change (RoC) in accuracy with respect to the differences in η:

	$\eta = 1$ - $\eta = e^{-2}$	$\eta = e^{-1}$ - $\eta = e^{-2}$	$\eta = 1 - \eta = e^{-1}$
Testing Accuracy RoC	0.012	0.061	0.004

Table 5: Rate of change (RoC) of testing accuracy with respect to η

The first value in *Table 5* represents the average rate of change in accuracy over the the entire range of η. The second value is the RoC between the two values where there is evidence that a local minimum exists. This value is almost six times the average RoC. When combined with the last value that represents the RoC between the three highest learning rates, it is clear that smallest learning rate $\eta = e^{-2}$ is getting stuck in a local minimum.

The absence of a dip in accuracy as the learning rate goes to one indicates that the larger steps taken during gradient descent do not cause the optimizer to leap past the minimum found into a worse one. There is, however, some evidence of oscillation around the minimum for sufficiently

large learning rates. For example, in *Figure 3* the training and validation curves are overlapping at epoch four and converge at epoch five before diverging slightly again after epoch six. The curves of the smallest learning rate, seen in *Figure 4*, converge much more smoothly and therefore waste fewer cycles finding the minimum once it is reached.

This experiment indicates that a learning rate greater than $\eta = e^{-2}$ should be selected, but also that some care should be taken to select a small enough learning rate that does not oscillate around the minimum and allows for timely convergence.

3.2 Regularization

Testing different regularization coefficients

	$\lambda = 1$	$\lambda = e^{-1}$	$\lambda = e^{-2}$	$\lambda = e^{-3}$	$\lambda = e^{-4}$
Training Accuracy	0.1019	0.1022	0.1508	0.7069	0.8636
Validation Accuracy	0.1104	0.1068	0.1068	0.7354	0.8616
Testing Accuracy	0.1135	0.1135	0.1135	0.7368	0.8745

Table 6: Train, validation and test performance on different regularization coefficients

Learning curves

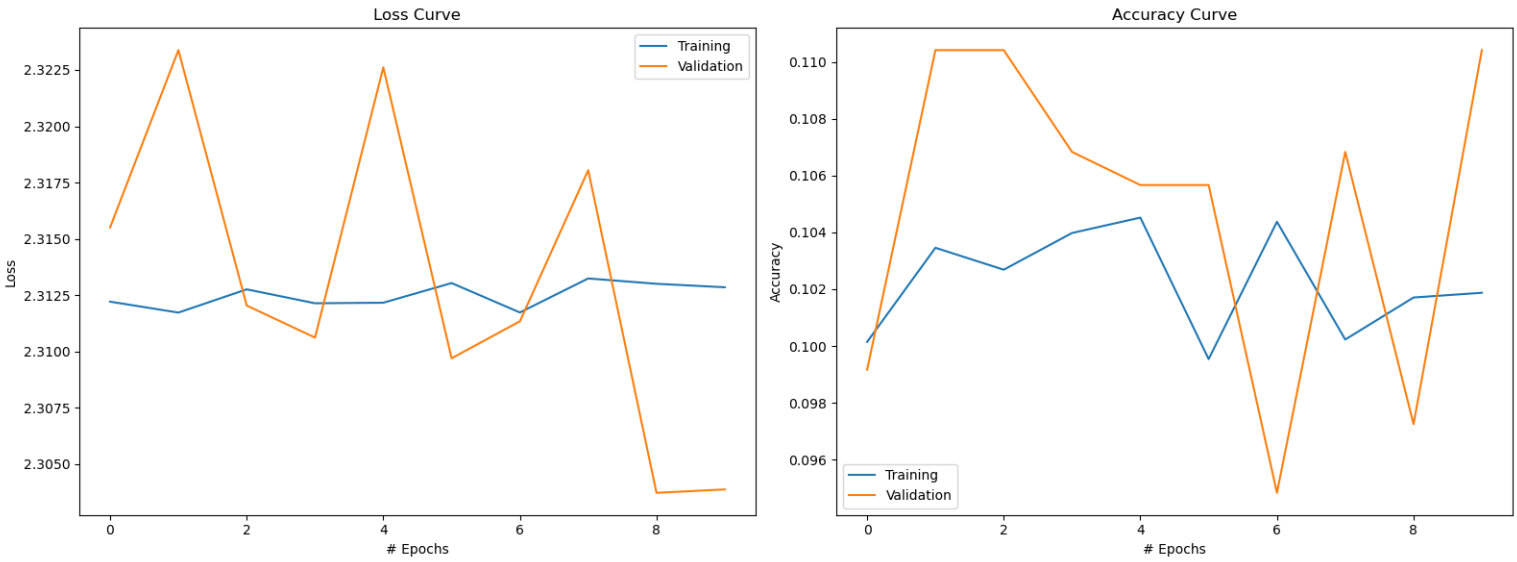

Figure 5: $\lambda = 1$

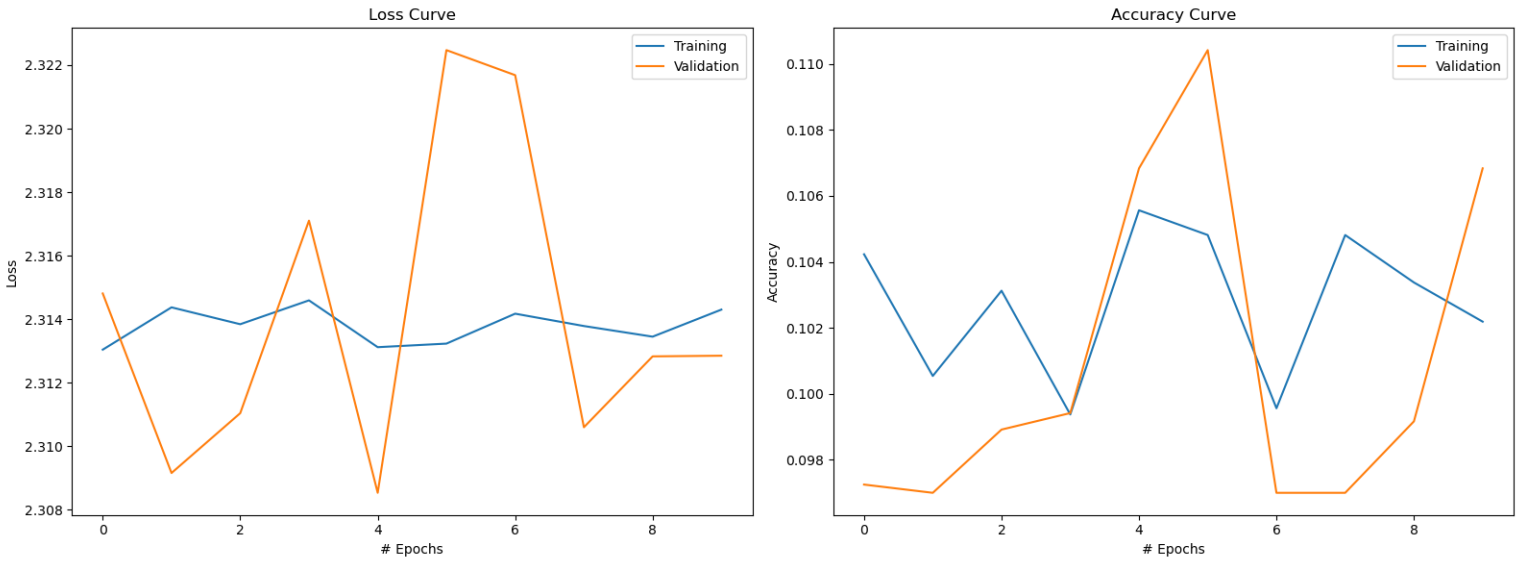

Figure 6: $\lambda = e^{-1}$

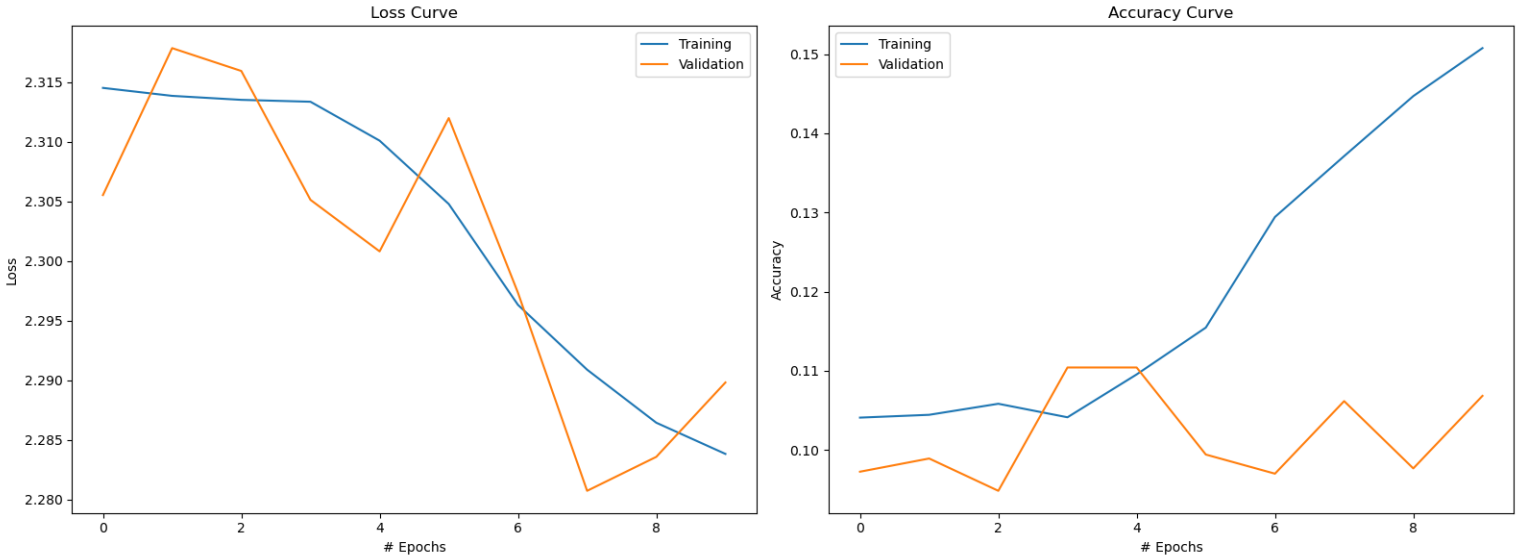

Figure 7: $\lambda = e^{-2}$

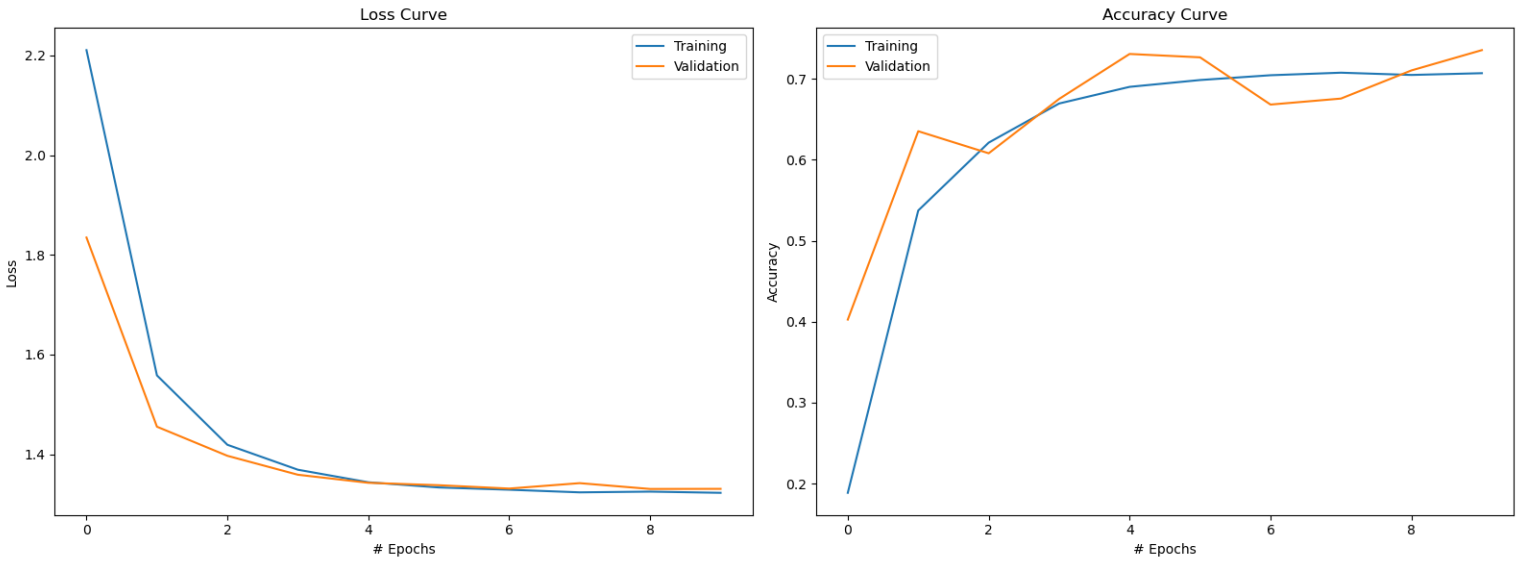

Figure 8: $\lambda = e^{-3}$

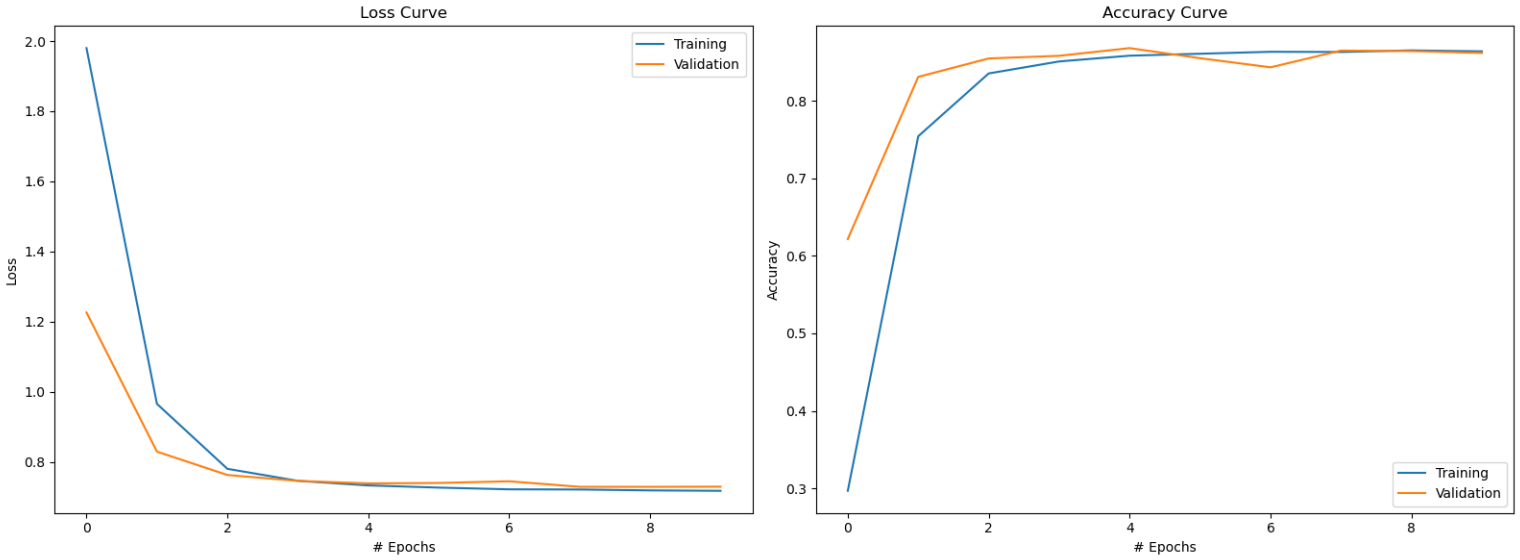

Figure 9: $\lambda = e^{-4}$

Analysis

It is clear from *Table 6* that smaller values of λ have a marked improvement on all forms of accu-

racy. This improvement is most apparent between values of $\lambda = e^{-3}$ and $\lambda = e^{-4}$ where accuracy increases by over 0.6 for all phases. This is indicative of tougher (higher) regularization restrictions preventing the model from sufficiently representing the complexity of the target function. In fact, all values tested in this experiment are too large to represent this complexity since in *3.1 Learning Rates* higher accuracies were achieved with the smaller, default regularization coefficient and similar learning rates.

This is further evidenced by the learning curves generated for each coefficient in *Figures 5-9*. Only the smallest coefficient, $\lambda = e^{-4}$, has a curve that could be referred to as "smooth". The larger coefficients cause the training and validation curves to oscillate wildly about each other, indicating that the model has learned almost nothing about the target concept.

An issue that can occur when decreasing λ is overfitting. This is due to the regularization coefficient imposing less authority towards keeping weights small and therefore increasing the complexity of the model. If the model becomes too complex, it can vary too much with the training data and be unable to generalize well. However, there is no evidence of overfitting in this experiment as there is no divergence of the training and validation accuracies as λ decreases.

Smaller regularization coefficients can also decrease time to convergence, because weights can be updated more quickly (larger updates). This can be visualized in *Figure 9* where the training and validation curves are closely positioned after two epochs. Larger coefficients struggle to converge as quickly or, as seen in the largest coefficients tested, fail to come close at the end of the ten epochs run.

To achieve minimal loss, the regularization coefficient should be lowered beyond the values tested here, but attention should be paid during selection toward the possibility of overfitting.

3.3 Hyper-parameter tuning

In order to improve the accuracy of the model beyond the first two experiments, the hyper-parameters to the model can be modified. This section will examine the effect and determination of optimal hyper-parameters for the **learning rate**, **regularization coefficient** and **batch size**. To more easily compare the outcomes of tuning with previous experiments, the number of epochs used for training has been left at the default 10.

Best model hyper-parameters

	η	λ	Batch size	Training accuracy	Validation accuracy	Testing accuracy
Best	0.6	0.0001	16	0.9810	0.9717	0.9747
Default	0.1	0.001	64	0.9234	0.9197	0.9262

Table 7: The best hyper-parameters and resulting accuracies compared to default

Learning rate

Since accuracy improves only marginally with learning rates higher than $\eta = e^{-1}$ per *Table 5*,

determination of the optimal rate shifts towards a focus on convergence. If the learning rate is set too high, the model could diverge by overshooting the minimum of the loss function or oscillating around the minimum as a result of taking too large of steps. A learning rate that is set too close to the lower bound could converge too slowly and risk not capturing the complexity of the model in the time allocated for training.

Regularization coefficient

Analysis from *3.2 Regularization* indicates that the concept is of considerable complexity. As a result, smaller regularization coefficients should produce the best results. However, as the regularization coefficient shrinks, the danger of overfitting increases. Therefore, optimization of this parameter requires identifying the boundary where the complexity of the model has increased too much and begins to overfit the training data and not generalize well.

Batch size

Batch size can have an effect on both stability and convergence as well as the model's ability to generalize. For small batch sizes, gradients are averaged over fewer samples and are updated more frequently. Frequent updates introduce more noise into the model via stochastic gradient descent (sgd), which can help the optimizer overcome local minima but increase convergence time.

Considerations

At first, the path to optimal parameters appears clear by following the logic of the above sections: find the smallest viable learning rate above the lower bound identified that sufficiently learns the complexity of the target, pair it with the smallest regularization coefficient that does not overfit, and run in batches that are small enough to avoid local minima without destabilizing training.

However, this does not take into account the dependencies these parameters impose on each other as they relate to overall model accuracy. For example, if the model is overfitting, it may be prudent to increase the regularization coefficient to combat it. If the learning rate and batch size are unmodified, the model is now learning a worse representation at the same speed and level of stochasticity. Therefore, it may be necessary to decrease the learning rate and/or increase the batch size to follow gradients more gradually and not accept poor gradients as easily in order to preserve accuracy.

Method

The considerations above were leveraged to narrow the search space for each parameter to a small range of viable values. Next, a grid search was performed on those ranges to further shrink the space. The search parameters can be found below.

η	λ	Batch size
[.2, .4, .6, .8]	[0.001, .0005, .0001]	[16, 32, 64, 128]

Table 8: Grid search parameters

Based on the results from the search, the best candidates were selected and attempted to be modified further in context of the considerations listed above.

Observations

The parameters found provide the highest accuracy through all phases. The validation and testing accuracies are closely tied with this model, but training is showing signs of overfitting. This fitment to the training data can be reduced by increasing the regularization coefficient, but at the cost of significantly reducing the accuracy. The trade-off did not end up being worth it, so instead consideration should be paid to the amount of training performed. It would likely be unwise to increase the number of epochs in this scenario, as further training may cause the learning curves to diverge and increase overfitting. Early stopping is likely necessary to preserve generalization ability with this model.

Increasing the batch size was attempted to reduce the stochasticity of training in hopes of promoting smoother convergence, but accuracy was once again decreased significantly as the optimizer was not able to escape local minima as easily. Reducing the learning rate further was prone to the same issues.

Conclusion

While the parameters found increase the probability of overfitment as training continues, they were selected for their dominance in the model's accuracy. Utilizing additional regularization methods such as early stopping can help alleviate the issues with the model's parameters while providing the best possible accuracy.

References

[Quotient rule] Quotient rule. (2020, November 5). Retrieved from Wikipedia website: https://en.wikipedia.org/wiki/Quotient_rule

[Zhang et al.] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding Deep Learning Requires Rethinking Generalization. *ICLR, 2017.*