CS 4644/7643: Deep Learning
Spring 2024
Problem Set 1

Instructor: Zsolt Kira
TAs: Aditya Akula, Avinash Prabhu, Bowen Zuo,
Katie Stevo, Krishanu Agarwal, Manav Agrawal,
Pranay Mathur, Vikranth Keerthipati, William Held
Discussions: https://piazza.com/gatech/spring2024/cs4644acs7643a

Due: Friday, February 2, 11:59pm

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   - For the **HW1 Writeup** component on Gradescope, you could upload one single PDF containing the answers to all the theory questions and the report for coding problems. **However, the solution to each problem/subproblem must be on a separate page. When submitting to Gradescope, please make sure to mark the page(s) corresponding to each problem/sub-problem.** Likewise, the pages of the report must also be marked to their corresponding subproblems.
   - For the **HW1 Code** component on Gradescope, please use the `collect_submission.sh` script provided and upload the resulting **hw1_code.zip** on Gradescope. Please make sure you have saved the most recent version of your codes.
   - Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.

2. LaTeX'd solutions are strongly encouraged (solution template available in the zip file in HW1 under the Assignments tab on Canvas), but scanned handwritten copies are acceptable. Hard copies are **not** accepted.

3. We generally encourage you to collaborate with other students.

   You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and *not* as a group activity. Please list the students you collaborated with.

1 Collaborators [0.5 points]

Ellias Cho

2 Optimization

1. We are given various information in the problem. We know that we have some arbitrary curve r(t) that lies within the level surface passing through x_0. We also know that $r(t) \in L_{f(x_0)} \forall t$ and that all values of r(t) are on the same curve of the value c. This is important, as it shows that the level curve is not affected by the value of t. Since the parameter t doesn't affect f_0, that means that the derivative of f_0 with respect to t is 0.

   The $\frac{\partial f_0}{\partial r}$ is given as the gradient vector in the problem. If we multiply the $\frac{\partial f_0}{\partial r}$ by $\frac{\partial r}{\partial t}$(the tangent to the curve), we get the $\frac{\partial f_0}{\partial t}$, which we determined was 0. This means that taking the dot product of the gradient vector and the tangent of r(t) at t, we show that the two values are orthogonal to each other.

   This orthogonal property is crucial for gradient descent to work, as the gradient will always be pointing to another level surface and never run parallel to the r(t). Additionally, when making calculations in a deep learning model, we will always optimize our movement and won't get stuck until we reach a local minimum. However, blindly following this direction can be bad when the models are dealing with high dimensionality. This direction could be pointing towards a saddle points, which would slow down the algorithm and possible cause it to get stuck depending on the learning rate of the algorithm.

2. Proof: if g has a local minimum at some w^t, then the gradient at $w^t = 0$ (converse doesn't hold true).

   We make the assumption that g has a local min. at w^t. This indicates that there must exist some $\gamma > 0$, such that this holds true for all points w with a given distance $||w^t$ - w$||_2 < \gamma$, g(w^t) $\leq$ g(w). [Given Information].

   Then, we observe the limit as vector x approaches 0 (where the vector is of size γ). We know that we have a distance of γ where $g(w) \geq g(w^t)$, so this inequality holds true:

$$\lim_{x \to 0} \frac{g(w^t + x) - g(w^t)}{||x||_2} \geq 0$$

   We take the gradient of w^t which becomes $\nabla g(w^t) = \lim_{x \to 0} \frac{g(w^t+x)-g(w^t)}{||x||_2}$. In order for the value of w^t to be 0, the gradient of the point must be less than or equal to 0. For the condition of the limit inequality above and the w^t to be 0, $\nabla g(w^t) = 0$.

   However, the converse is not necessarily true. Let's use the example where the function of g(x) is equal to x^5. The first derivative is $4x^5$ showing us that there is a critical point at x = 0. But, when we take the second derivative and set it equal to 0, we don't get a positive or

negative value, we just get 0 again. This means that the gradient is equal to 0 at this x = 0. However, this isn't a local minimum because g(x) > 0 on the right side and g(x) < 0 on the left side of the origin. Just because the gradient is 0 at a given point does not mean that there exists a local minimum (as demonstrated with things like saddle points and some critical points).

3. Prove that if a differentiable function g: $\mathbb{R}^n \rightarrow \mathbb{R}$ is convex and the gradient at some w^* is 0, then w^* is the global minimum of g.

   Proof: We have a function that is convex, has the correct set mapping $\mathbb{R}^n \rightarrow \mathbb{R}$, and for some point w*, $\nabla g(w^*) = 0$. [Given Information]. We will show that w* is the global minimum of the function g.

   Since g is convex, this property holds true for $x, y \in domain(g)$:

   $$g(k\mathbf{x} + (1-k)\mathbf{y}) \leq kg(\mathbf{x}) + (1-k)g(\mathbf{y}) \ \forall k \in [0,1]$$

   We can apply the first-order Taylor expansion of the convex function g, around the point w*. Taylor Expansion: $f(a) \geq f(x) + f'(x)(a-x)$ where x is the specific point in function f and a is any point in the domain f.

   So, we can show that $g(u) \geq g(w^*) + \nabla g(w^*)(u - w^*)$

   Since ∇ g(w*) = 0, we can show that the inequality becomes $g(u) \geq g(w^*)$. This means that for any $u \in domain(g)$, the value of g will be greater than w*. This shows that w* is the global minimum for the function g, given the conditions that g is differentiable and convex, and there exists a point where the gradient is 0.

   Therefore, the statement that when give a differentiable function g: $\mathbb{R}^n \rightarrow \mathbb{R}$ that is convex and a gradient for some point w* equal to 0, then w* is the global minimum.

4. We will derive the gradient of s with respect to the logits, given the softmax function:

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

   To derive $\frac{\partial s_i}{\partial z_j}$ when i = j, we must use the quotient rule:

   $$\text{Quotient Rule: } \frac{f(x)}{g(x)} = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$$

$$\text{SO } \frac{\partial s_i}{\partial z_i} = \frac{\frac{\partial e^{z_i}}{\partial z_i} \times \sum_k e^{z_k}) - (e^{z_i} \times \frac{\partial \sum_k e^{z_k}}{\partial z_i})}{(\sum_k e^{z_k})^2}$$

We can simplify the equation by solving the partials and substituting, resulting in the following equation:

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

After doing some factoring, we arrive at the following calculations:

$$\frac{\partial s_i}{\partial z_i} = \frac{e^{z_i}(\sum_k e^{z_k} - e^{z_i})}{(\sum_k e^{z_k})^2} = \frac{e^{z_i}}{\sum_k e^{z_k}} \times \frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}} = \frac{e^{z_i}}{\sum_k e^{z_k}} \times (1 - \frac{e^{z_i}}{\sum_k e^{z_k}}) = s_i(1 - s_i)$$

So, what happens when i ! = j ? We have to perform and simply the quotient rule with respect to $\frac{\partial s_i}{\partial z_j}$.

$$\frac{\partial s_i}{\partial z_j} = \frac{(\frac{\partial e^{z_i}}{\partial z_j} \times \sum_k e^{z_k}) - (\frac{\partial \sum_k e^{z_k}}{\partial z_j} \times e^{z_i})}{(\sum_k e^{z_k})^2}$$

After plugging in the derivatives and simplifying the formula we arrive at:

$$\frac{\partial s_i}{\partial z_j} = \frac{(0 \times \sum_k e^{z_k}) - (e^{z_j} \times e^{z_i})}{(\sum_k e^{z_k})^2} = \frac{-e^{z_j} e^{z_i}}{(\sum_k e^{z_k})^2} = -1 \times \frac{e^{z_i}}{\sum_k e^{z_k}} \times \frac{e^{z_j}}{\sum_k e^{z_k}} = -s_i s_j$$

This results in the following :

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

3 Directed Acyclic Graphs (DAG)

6. Prove that if the graph G is a DAG, then G has a topological ordering. We can prove this using proof by induction. Due to the face that G is a DAG, there must exist (there may be multiple, choose one at random) a vertex v such that 0 edges are going into the vertex (a source vertex). We made assumption that there is more than one node in the graph G, so we remove that vertex v from the graph. The graph G will still be a DAG, as removing a node with no edges entering it will not cause a cycle to form. Then, we remove all the edges from that vertex and add the vertex v into a list for the topological ordering. This is the inductive step. Thus, the inductive hypothesis shows us that G must have a topological ordering, as we can iteratively continue this process until a topological ordering is created for G.

7. Prove that if the graph G has a topological ordering, then G is a DAG. In order to do this, we will complete a proof by contradiction. Let's assume that graph G has a cycle. Let v_i be the first vertex in the topological ordering such that there is a cycle containing v_i . Since v_i is the first vertex in the cycle, there must be a path going out of v_i to a vertex v_j

such that the last edge of vertex v_j goes into vertex v_i. This causes a contradiction because for a topological ordering all edges go from Left to the Right of the list. If there is an edge going from v_j to v_i such that j > i, it violated the property of the topological ordering. Thus, the assumption made in the beginning that graph G contains a cycle leads to a contradiction. Thus graph G is acyclic, meaning that if a graph G has a topological ordering, then G is a DAG.

4 Paper Review

8.) This paper details a new approach towards Machine Learning involving Weight-Agnostic Neural Networks. These NN are capable of performing tasks without explicit weight training; instead of relying on learning through weights, they aim to optimize the perfect architecture for the given task. The proposed network involves utilizing topology search in order to find neural net architectures that rely on a single, shared weight parameter. The search algorithm involves multi-objective optimization, considering factors such as performance, complexity, and dominance relations. Strengths of this paper involve developing a rather novel concept that follows the structure of the brain closer. This paper details this process and demonstrates the out-of-box thinking that leads to new and innovative solutions. Moreover, the approach simplifies the weights of the neural network to a single parameter, increasing the computational efficiency and allowing a range of weights to be tested. I think that a majority of the weaknesses from the paper involve simplification, evaluation limitations, and application. From the lack of research in the application of this method (as most ML algorithms are case-specific) to the inability to access performance, this approach to Neural Networks needs to be further investigated for real-world use.

9.) I believe the paper was very interesting, as it challenges the notion that we need to rely on weights for machine learning algorithms. This might shift our attention away from finding methods of better optimization of weights to finding ways to improve the creation/architecture of neural nets. I remember in class we discussed that theoretically we can use three layers for neural networks to represent many possible machine learning problems such as classification, super-solution, style transfer, etc. I found that this idea relates heavily to the paper, as they focus more on how the neural networks as opposed to how it is trained. I would be interested in looking into the research regarding the intersection of these two ideals. It's exciting how new and innovative solutions are created in research labs, and I believe that many things can be learned about the approaches to intelligence and discovery that the authors of this paper pioneered.

5 Implement and train a network on MNIST

```
#Cell 1
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
#Cell 2
%cd /content/drive/MyDrive/DLHW1/data/

!sh get_data.sh
```

```
    /content/drive/MyDrive/DLHW1/data
    --2024-02-04 07:34:24--  https://pjreddie.com/media/files/mnist_train.csv
    Resolving pjreddie.com (pjreddie.com)... 162.0.215.52
    Connecting to pjreddie.com (pjreddie.com)|162.0.215.52|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 109575994 (104M) [text/csv]
    Saving to: ‘mnist_train.csv.14’

    mnist_train.csv.14  100%[===================>] 104.50M  12.4MB/s    in 13s

    2024-02-04 07:34:37 (8.16 MB/s) - ‘mnist_train.csv.14’ saved [109575994/109

    --2024-02-04 07:34:37--  https://pjreddie.com/media/files/mnist_test.csv
    Resolving pjreddie.com (pjreddie.com)... 162.0.215.52
    Connecting to pjreddie.com (pjreddie.com)|162.0.215.52|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 18289443 (17M) [text/csv]
    Saving to: ‘mnist_test.csv.13’

    mnist_test.csv.13   100%[===================>]  17.44M  5.11MB/s    in 3.4s

    2024-02-04 07:34:41 (5.11 MB/s) - ‘mnist_test.csv.13’ saved [18289443/18289
```

```
# Cell 3
# Run all local tests in this block
# If you get an error saying test not found, add an __init__.py file in the
# tests directory
%cd /content/drive/MyDrive/DLHW1/
! python -m unittest tests.test_training
```

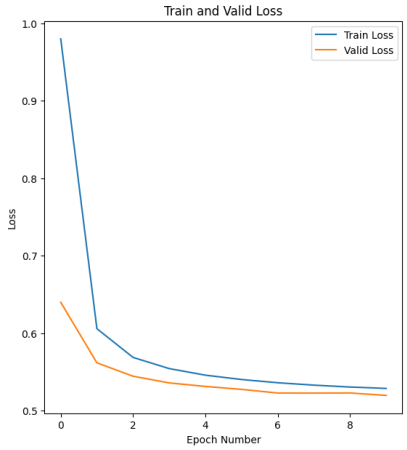
```
    /content/drive/MyDrive/DLHW1
    Loading training data...
    Traceback (most recent call last):
      File "/usr/lib/python3.10/runpy.py", line 196, in _run_module_as_main
        return _run_code(code, main_globals, None,
      File "/usr/lib/python3.10/runpy.py", line 86, in _run_code
        exec(code, run_globals)
      File "/usr/lib/python3.10/unittest/__main__.py", line 18, in <module>
        main(module=None)
      File "/usr/lib/python3.10/unittest/main.py", line 101, in __init__
        self.runTests()
      File "/usr/lib/python3.10/unittest/main.py", line 271, in runTests
        self.result = testRunner.run(self.test)
      File "/usr/lib/python3.10/unittest/runner.py", line 184, in run
        test(result)
      File "/usr/lib/python3.10/unittest/suite.py", line 84, in __call__
        return self.run(*args, **kwds)
      File "/usr/lib/python3.10/unittest/suite.py", line 122, in run
        test(result)
      File "/usr/lib/python3.10/unittest/suite.py", line 84, in __call__
        return self.run(*args, **kwds)
      File "/usr/lib/python3.10/unittest/suite.py", line 122, in run
        test(result)
      File "/usr/lib/python3.10/unittest/suite.py", line 84, in __call__
        return self.run(*args, **kwds)
      File "/usr/lib/python3.10/unittest/suite.py", line 122, in run
        test(result)
      File "/usr/lib/python3.10/unittest/case.py", line 650, in __call__
        return self.run(*args, **kwds)
      File "/usr/lib/python3.10/unittest/case.py", line 591, in run
        self._callTestMethod(testMethod)
      File "/usr/lib/python3.10/unittest/case.py", line 549, in _callTestMethod
        method()
      File "/content/drive/MyDrive/DLHW1/tests/test_training.py", line 48, in t
        train_data, train_label, _, _ = load_mnist_trainval()
      File "/content/drive/MyDrive/DLHW1/utils.py", line 44, in load_mnist_trai
        data, label = load_csv('./data/mnist_train.csv')
```

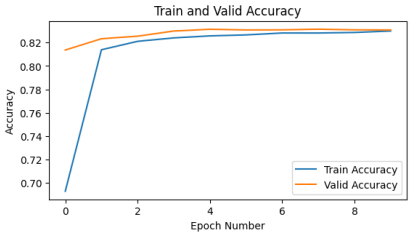
config_exp.yaml

```
Train:
  batch_size: 62
  learning_rate: 0.15
  reg: 0.0001
  epochs: 20
  momentum: 0.9
  debug: True

Model:
  type: TwoLayerNet # SoftmaxRegression or Tw
  hidden_size: 128 # only applicable for Twol

```

```
      File "/content/drive/MyDrive/DLHW1/utils.py", line 27, in load_csv
        x = [int(px) / 255 for px in x]
      File "/content/drive/MyDrive/DLHW1/utils.py", line 27, in <listcomp>
        x = [int(px) / 255 for px in x]
    KeyboardInterrupt
    ^C


#Cell 4
import yaml
import copy
%cd /content/drive/MyDrive/DLHW1


from models import TwoLayerNet, SoftmaxRegression
from optimizer import SGD
from utils import load_mnist_trainval, load_mnist_test, generate_batched_data,

    /content/drive/MyDrive/DLHW1


# Cell 5
%matplotlib inline
def train_model(yaml_config_file):
  args = {}
  with open(yaml_config_file) as f:
      config = yaml.full_load(f)

  for key in config:
      for k, v in config[key].items():
          args[k] = v

  # Prepare MNIST data
  train_data, train_label, val_data, val_label = load_mnist_trainval()
  test_data, test_label = load_mnist_test()

  # Prepare model and optimizer
  if args["type"] == 'SoftmaxRegression':
      model = SoftmaxRegression()
  elif args["type"] == 'TwoLayerNet':
      model = TwoLayerNet(hidden_size=args["hidden_size"])
  optimizer = SGD(learning_rate=args["learning_rate"], reg=args["reg"])

  # Training Code
  train_loss_history = []
  train_acc_history = []
  valid_loss_history = []
  valid_acc_history = []
  best_acc = 0.0
  best_model = None
  for epoch in range(args["epochs"]):
      batched_train_data, batched_train_label = generate_batched_data(train_data
      epoch_loss, epoch_acc = train(epoch, batched_train_data, batched_train_lab

      train_loss_history.append(epoch_loss)
      train_acc_history.append(epoch_acc)
      # evaluate on test data
      batched_test_data, batched_test_label = generate_batched_data(val_data, va
      valid_loss, valid_acc = evaluate(batched_test_data, batched_test_label, mo
      if args["debug"]:
          print("* Validation Accuracy: {accuracy:.4f}".format(accuracy=valid_ac

      valid_loss_history.append(valid_loss)
      valid_acc_history.append(valid_acc)

      if valid_acc > best_acc:
          best_acc = valid_acc
          best_model = copy.deepcopy(model)

  #Testing Code
  batched_test_data, batched_test_label = generate_batched_data(test_data, test_
  _, test_acc = evaluate(batched_test_data, batched_test_label, best_model) # te
  if args["debug"]:
      print("Final Accuracy on Test Data: {accuracy:.4f}".format(accuracy=test_a

  return train_loss_history, train_acc_history, valid_loss_history, valid_acc_hi
```

```
# Cell 6
# train softmax model
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
     Epoch: [9][300/750]     Batch Time 0.001        Batch Loss 0.5260       Tra
     Epoch: [9][310/750]     Batch Time 0.001        Batch Loss 0.5351       Tra
     Epoch: [9][320/750]     Batch Time 0.001        Batch Loss 0.4656       Tra
     Epoch: [9][330/750]     Batch Time 0.001        Batch Loss 0.4512       Tra
     Epoch: [9][340/750]     Batch Time 0.001        Batch Loss 0.6849       Tra
     Epoch: [9][350/750]     Batch Time 0.001        Batch Loss 0.5645       Tra
     Epoch: [9][360/750]     Batch Time 0.001        Batch Loss 0.3507       Tra
     Epoch: [9][370/750]     Batch Time 0.001        Batch Loss 0.4832       Tra
     Epoch: [9][380/750]     Batch Time 0.001        Batch Loss 0.5418       Tra
     Epoch: [9][390/750]     Batch Time 0.001        Batch Loss 0.6418       Tra
     Epoch: [9][400/750]     Batch Time 0.001        Batch Loss 0.4753       Tra
     Epoch: [9][410/750]     Batch Time 0.001        Batch Loss 0.4591       Tra
     Epoch: [9][420/750]     Batch Time 0.001        Batch Loss 0.4200       Tra
     Epoch: [9][430/750]     Batch Time 0.001        Batch Loss 0.6319       Tra
     Epoch: [9][440/750]     Batch Time 0.001        Batch Loss 0.4269       Tra
     Epoch: [9][450/750]     Batch Time 0.001        Batch Loss 0.4988       Tra
     Epoch: [9][460/750]     Batch Time 0.001        Batch Loss 0.4887       Tra
     Epoch: [9][470/750]     Batch Time 0.001        Batch Loss 0.6019       Tra
     Epoch: [9][480/750]     Batch Time 0.001        Batch Loss 0.7691       Tra
     Epoch: [9][490/750]     Batch Time 0.001        Batch Loss 0.4468       Tra
     Epoch: [9][500/750]     Batch Time 0.001        Batch Loss 0.5376       Tra
     Epoch: [9][510/750]     Batch Time 0.001        Batch Loss 0.8341       Tra
     Epoch: [9][520/750]     Batch Time 0.001        Batch Loss 0.5015       Tra
     Epoch: [9][530/750]     Batch Time 0.001        Batch Loss 0.7515       Tra
     Epoch: [9][540/750]     Batch Time 0.001        Batch Loss 0.4510       Tra
     Epoch: [9][550/750]     Batch Time 0.001        Batch Loss 0.4467       Tra
     Epoch: [9][560/750]     Batch Time 0.001        Batch Loss 0.5459       Tra
     Epoch: [9][570/750]     Batch Time 0.001        Batch Loss 0.5357       Tra
     Epoch: [9][580/750]     Batch Time 0.001        Batch Loss 0.2222       Tra
     Epoch: [9][590/750]     Batch Time 0.001        Batch Loss 0.4025       Tra
     Epoch: [9][600/750]     Batch Time 0.001        Batch Loss 0.6422       Tra
     Epoch: [9][610/750]     Batch Time 0.001        Batch Loss 0.4586       Tra
     Epoch: [9][620/750]     Batch Time 0.001        Batch Loss 0.6901       Tra
     Epoch: [9][630/750]     Batch Time 0.001        Batch Loss 0.6486       Tra
     Epoch: [9][640/750]     Batch Time 0.001        Batch Loss 0.6308       Tra
     Epoch: [9][650/750]     Batch Time 0.001        Batch Loss 0.4211       Tra
     Epoch: [9][660/750]     Batch Time 0.001        Batch Loss 0.8335       Tra
     Epoch: [9][670/750]     Batch Time 0.001        Batch Loss 0.4629       Tra
     Epoch: [9][680/750]     Batch Time 0.001        Batch Loss 0.4252       Tra
     Epoch: [9][690/750]     Batch Time 0.003        Batch Loss 0.5258       Tra
     Epoch: [9][700/750]     Batch Time 0.001        Batch Loss 0.5944       Tra
     Epoch: [9][710/750]     Batch Time 0.001        Batch Loss 0.4809       Tra
     Epoch: [9][720/750]     Batch Time 0.001        Batch Loss 0.3237       Tra
     Epoch: [9][730/750]     Batch Time 0.001        Batch Loss 0.3405       Tra
     Epoch: [9][740/750]     Batch Time 0.001        Batch Loss 0.5868       Tra
     * Average Accuracy of Epoch 9 is: 0.8300
     Evaluate: [0/188]       Batch Accuracy 0.9062
     Evaluate: [1/188]       Batch Accuracy 0.8750
     Evaluate: [2/188]       Batch Accuracy 0.8906
     Evaluate: [3/188]       Batch Accuracy 0.9219
     Evaluate: [4/188]       Batch Accuracy 0.8281
     Evaluate: [5/188]       Batch Accuracy 0.7188
     Evaluate: [6/188]       Batch Accuracy 0.8281
     Evaluate: [7/188]       Batch Accuracy 0.8750
     Evaluate: [8/188]       Batch Accuracy 0.8438
     Evaluate: [9/188]       Batch Accuracy 0.8594
     Evaluate: [10/188]      Batch Accuracy 0.8125
     Evaluate: [11/188]      Batch Accuracy 0.8438
     Evaluate: [12/188]      Batch Accuracy 0.8281

# Cell 7
# plot results for softmax model
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

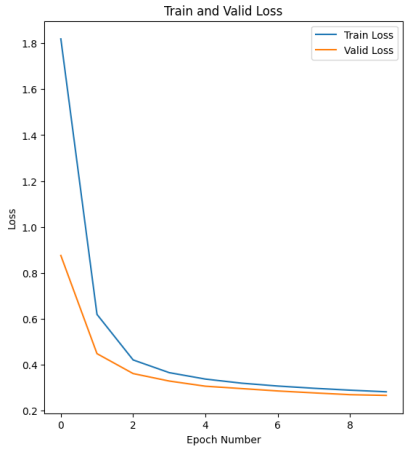


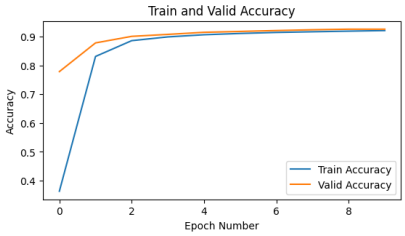



```
# Cell 8
# train two layer neural network
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history = t
```

```
Evaluate: [145/157]     Batch Accuracy 0.9688
Evaluate: [146/157]     Batch Accuracy 1.0000
Evaluate: [147/157]     Batch Accuracy 0.9219
Evaluate: [148/157]     Batch Accuracy 0.9688
Evaluate: [149/157]     Batch Accuracy 0.9375
Evaluate: [150/157]     Batch Accuracy 0.9062
Evaluate: [151/157]     Batch Accuracy 0.9219
Evaluate: [152/157]     Batch Accuracy 0.8125
Evaluate: [153/157]     Batch Accuracy 0.9375
Evaluate: [154/157]     Batch Accuracy 0.8438
Evaluate: [155/157]     Batch Accuracy 0.9062
Evaluate: [156/157]     Batch Accuracy 0.9375
Final Accuracy on Test Data: 0.9247
```

```
# Cell 9
# plot two layer neural network
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```





Assignment 1 Writeup

- Name: Jadon
- GT Email: jco9@gatech.edu
- GT ID: 903725118

Two Layer Neural Network

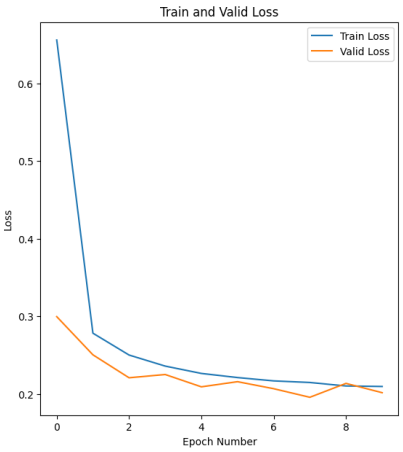
Learning Rates

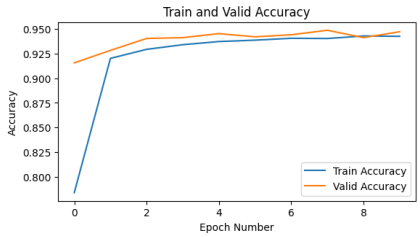
- Tune the Two Layer Neural Network with various learning rates (while keeping all other hyperparameters constant) by changing the config file.
    - lr = 1
    - lr = 1e-1
    - lr = 1e-2
    - lr = 5e-2

```
# Cell 10
# Change lr to 1 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history = t
```

```
    Evaluate: [100/157]     Batch Accuracy 0.9375
    Evaluate: [101/157]     Batch Accuracy 0.9688
    Evaluate: [102/157]     Batch Accuracy 0.8906
    Evaluate: [103/157]     Batch Accuracy 0.8906
    Evaluate: [104/157]     Batch Accuracy 1.0000
    Evaluate: [105/157]     Batch Accuracy 0.9062
    Evaluate: [106/157]     Batch Accuracy 0.9375
    Evaluate: [107/157]     Batch Accuracy 0.9844
    Evaluate: [108/157]     Batch Accuracy 0.9844
    Evaluate: [109/157]     Batch Accuracy 1.0000
    Evaluate: [110/157]     Batch Accuracy 0.9688
    Evaluate: [111/157]     Batch Accuracy 0.9688
    Evaluate: [112/157]     Batch Accuracy 0.9844
    Evaluate: [113/157]     Batch Accuracy 1.0000
    Evaluate: [114/157]     Batch Accuracy 1.0000
    Evaluate: [115/157]     Batch Accuracy 1.0000
    Evaluate: [116/157]     Batch Accuracy 0.9375
    Evaluate: [117/157]     Batch Accuracy 0.9844
    Evaluate: [118/157]     Batch Accuracy 0.9844
    Evaluate: [119/157]     Batch Accuracy 1.0000
    Evaluate: [120/157]     Batch Accuracy 1.0000
    Evaluate: [121/157]     Batch Accuracy 0.9688
    Evaluate: [122/157]     Batch Accuracy 0.9688
    Evaluate: [123/157]     Batch Accuracy 0.9688
    Evaluate: [124/157]     Batch Accuracy 0.9844
    Evaluate: [125/157]     Batch Accuracy 0.9688
    Evaluate: [126/157]     Batch Accuracy 0.9531
    Evaluate: [127/157]     Batch Accuracy 0.9688
    Evaluate: [128/157]     Batch Accuracy 1.0000
    Evaluate: [129/157]     Batch Accuracy 0.9688
    Evaluate: [130/157]     Batch Accuracy 0.9844
    Evaluate: [131/157]     Batch Accuracy 0.9688
    Evaluate: [132/157]     Batch Accuracy 1.0000
    Evaluate: [133/157]     Batch Accuracy 0.9531
    Evaluate: [134/157]     Batch Accuracy 0.9844
    Evaluate: [135/157]     Batch Accuracy 1.0000
    Evaluate: [136/157]     Batch Accuracy 1.0000
    Evaluate: [137/157]     Batch Accuracy 1.0000
    Evaluate: [138/157]     Batch Accuracy 0.9844
    Evaluate: [139/157]     Batch Accuracy 1.0000
    Evaluate: [140/157]     Batch Accuracy 0.9219
    Evaluate: [141/157]     Batch Accuracy 0.9219
    Evaluate: [142/157]     Batch Accuracy 1.0000
    Evaluate: [143/157]     Batch Accuracy 0.9688
    Evaluate: [144/157]     Batch Accuracy 1.0000
    Evaluate: [145/157]     Batch Accuracy 0.9688
    Evaluate: [146/157]     Batch Accuracy 1.0000
    Evaluate: [147/157]     Batch Accuracy 0.9844
    Evaluate: [148/157]     Batch Accuracy 0.9688
    Evaluate: [149/157]     Batch Accuracy 0.9844
    Evaluate: [150/157]     Batch Accuracy 0.9688
    Evaluate: [151/157]     Batch Accuracy 0.9219
    Evaluate: [152/157]     Batch Accuracy 0.8281
    Evaluate: [153/157]     Batch Accuracy 0.9531
    Evaluate: [154/157]     Batch Accuracy 0.8438
    Evaluate: [155/157]     Batch Accuracy 0.9062
    Evaluate: [156/157]     Batch Accuracy 1.0000
    Final Accuracy on Test Data: 0.9485
```

```
# Cell 11
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_a
```

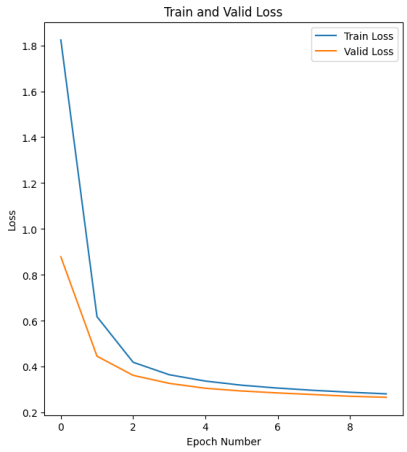


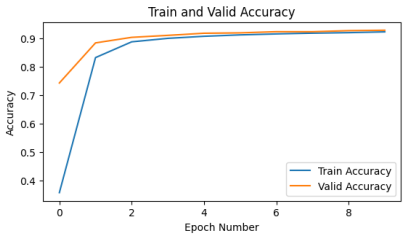



```
# Cell 12
# Change lr to 1e-1 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [145/157]     Batch Accuracy 0.9688
Evaluate: [146/157]     Batch Accuracy 1.0000
Evaluate: [147/157]     Batch Accuracy 0.9219
Evaluate: [148/157]     Batch Accuracy 0.9688
Evaluate: [149/157]     Batch Accuracy 0.9375
Evaluate: [150/157]     Batch Accuracy 0.9062
Evaluate: [151/157]     Batch Accuracy 0.9062
Evaluate: [152/157]     Batch Accuracy 0.7969
Evaluate: [153/157]     Batch Accuracy 0.9062
Evaluate: [154/157]     Batch Accuracy 0.8281
Evaluate: [155/157]     Batch Accuracy 0.8906
Evaluate: [156/157]     Batch Accuracy 0.9375
Final Accuracy on Test Data: 0.9256
```

```
# Cell 13
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

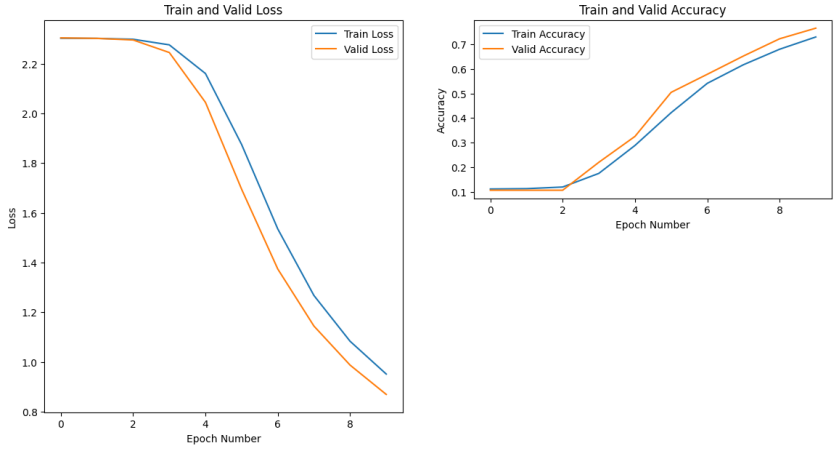





```
# Cell 14
# Change lr to 1e-2 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [128/157]     Batch Accuracy 0.7188
Evaluate: [129/157]     Batch Accuracy 0.7656
Evaluate: [130/157]     Batch Accuracy 0.7500
Evaluate: [131/157]     Batch Accuracy 0.7812
Evaluate: [132/157]     Batch Accuracy 0.7969
Evaluate: [133/157]     Batch Accuracy 0.7969
Evaluate: [134/157]     Batch Accuracy 0.7969
Evaluate: [135/157]     Batch Accuracy 0.9062
Evaluate: [136/157]     Batch Accuracy 0.9375
Evaluate: [137/157]     Batch Accuracy 0.9531
Evaluate: [138/157]     Batch Accuracy 0.8906
Evaluate: [139/157]     Batch Accuracy 0.9219
Evaluate: [140/157]     Batch Accuracy 0.8125
Evaluate: [141/157]     Batch Accuracy 0.7656
Evaluate: [142/157]     Batch Accuracy 0.9219
Evaluate: [143/157]     Batch Accuracy 0.8281
Evaluate: [144/157]     Batch Accuracy 0.8438
Evaluate: [145/157]     Batch Accuracy 0.7500
Evaluate: [146/157]     Batch Accuracy 0.8281
Evaluate: [147/157]     Batch Accuracy 0.7656
Evaluate: [148/157]     Batch Accuracy 0.7812
Evaluate: [149/157]     Batch Accuracy 0.7812
Evaluate: [150/157]     Batch Accuracy 0.7969
Evaluate: [151/157]     Batch Accuracy 0.7656
Evaluate: [152/157]     Batch Accuracy 0.6875
Evaluate: [153/157]     Batch Accuracy 0.7969
Evaluate: [154/157]     Batch Accuracy 0.7031
Evaluate: [155/157]     Batch Accuracy 0.7031
Evaluate: [156/157]     Batch Accuracy 0.6875
Final Accuracy on Test Data: 0.7564
```

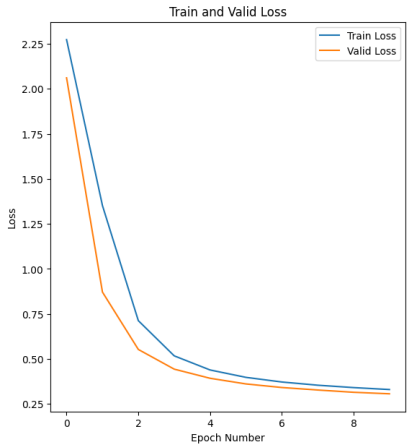
```
# Cell 15
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

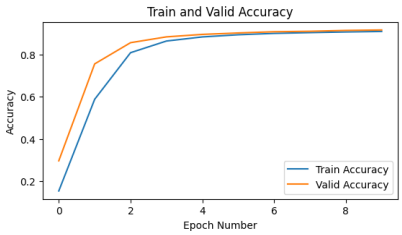



```
# Cell 16
# Change lr to 5e-2 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [111/157]     Batch Accuracy 0.9531
Evaluate: [112/157]     Batch Accuracy 0.9375
Evaluate: [113/157]     Batch Accuracy 0.9375
Evaluate: [114/157]     Batch Accuracy 0.9688
Evaluate: [115/157]     Batch Accuracy 0.9844
Evaluate: [116/157]     Batch Accuracy 0.9375
Evaluate: [117/157]     Batch Accuracy 0.9062
Evaluate: [118/157]     Batch Accuracy 0.9688
Evaluate: [119/157]     Batch Accuracy 1.0000
Evaluate: [120/157]     Batch Accuracy 0.9688
Evaluate: [121/157]     Batch Accuracy 0.9375
Evaluate: [122/157]     Batch Accuracy 0.7656
Evaluate: [123/157]     Batch Accuracy 0.8438
Evaluate: [124/157]     Batch Accuracy 0.9531
Evaluate: [125/157]     Batch Accuracy 0.9688
Evaluate: [126/157]     Batch Accuracy 0.9219
Evaluate: [127/157]     Batch Accuracy 0.9844
Evaluate: [128/157]     Batch Accuracy 0.9688
Evaluate: [129/157]     Batch Accuracy 0.9375
Evaluate: [130/157]     Batch Accuracy 0.9531
Evaluate: [131/157]     Batch Accuracy 0.9375
Evaluate: [132/157]     Batch Accuracy 0.9844
Evaluate: [133/157]     Batch Accuracy 0.9375
Evaluate: [134/157]     Batch Accuracy 0.9844
Evaluate: [135/157]     Batch Accuracy 1.0000
Evaluate: [136/157]     Batch Accuracy 1.0000
Evaluate: [137/157]     Batch Accuracy 1.0000
Evaluate: [138/157]     Batch Accuracy 0.9844
Evaluate: [139/157]     Batch Accuracy 1.0000
Evaluate: [140/157]     Batch Accuracy 0.8750
Evaluate: [141/157]     Batch Accuracy 0.9219
Evaluate: [142/157]     Batch Accuracy 0.9844
Evaluate: [143/157]     Batch Accuracy 0.9688
Evaluate: [144/157]     Batch Accuracy 0.9844
Evaluate: [145/157]     Batch Accuracy 0.9531
Evaluate: [146/157]     Batch Accuracy 0.9844
Evaluate: [147/157]     Batch Accuracy 0.9062
Evaluate: [148/157]     Batch Accuracy 0.9531
Evaluate: [149/157]     Batch Accuracy 0.9375
Evaluate: [150/157]     Batch Accuracy 0.9062
Evaluate: [151/157]     Batch Accuracy 0.9062
Evaluate: [152/157]     Batch Accuracy 0.7969
Evaluate: [153/157]     Batch Accuracy 0.9062
Evaluate: [154/157]     Batch Accuracy 0.8125
Evaluate: [155/157]     Batch Accuracy 0.8750
Evaluate: [156/157]     Batch Accuracy 0.9375
Final Accuracy on Test Data: 0.9143
```

```
# Cell 17
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```





Describe and explain your findings here:

After running the training model code with various learning rates (1, 0.1, 0.001, 0.005), the experiment found interesting results as demonstrated by the graph plots.

The learning rate of 1 was relatively high, meaning that it made adjustments to the parameters fairly quickly. Even though the model ran relatively fast, there were times when the accuracy and loss were overshot (causing a lack of convergence). In some cases, this can lead to instability which will cause divergence or models failing to find the most optimal solution.

The learning rate of 0.1 was a great starting point. This rate leads to a great balance of speed (with reduced complexity) and stability (never overshooting the accuracy). The convergence took more time than its higher counterpart but led to relatively little inaccuracies. This learning rate should be adjusted and altered in order to optimize the model as best as possible.

The learning rate of 0.001 was relatively small in this case, as the graph indicated that it didn't converge in 10 epochs. A smaller learning rate means that it takes more iterations to find the optimal solution (slower models) and it can can get stuck in local minimas (not returning the most optimal solution).

The learning rate of 0.05 was similar to the 0.1 learning rate. However, a key distinction is that this rate aimed to balance the pros of faster convergence with stability.

Overall, the somewhere around 0.1 was the optimal learning rate for this experiment. Other values could be tested but this interval seems to be the best for this model.

Regularization

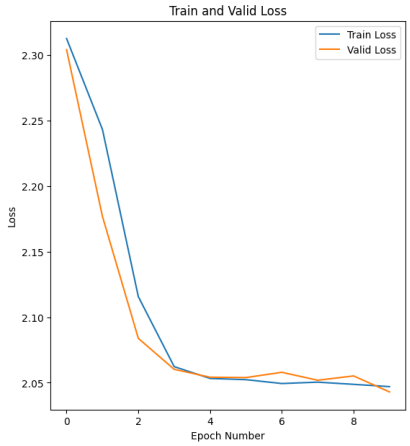
- Tune the Two Layer Neural Network with various regularization coefficients (while keeping all other hyperparameters constant) by changing the config file.
    - reg = 1e-1
    - reg = 1e-2
    - reg = 1e-3
    - reg = 1e-4

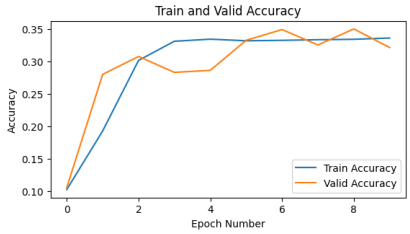
- reg = 1

```
# Cell 18
# Change reg to 1e-1 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
    Evaluate: [100/157]     Batch Accuracy 0.4531
    Evaluate: [101/157]     Batch Accuracy 0.4219
    Evaluate: [102/157]     Batch Accuracy 0.3281
    Evaluate: [103/157]     Batch Accuracy 0.3594
    Evaluate: [104/157]     Batch Accuracy 0.3750
    Evaluate: [105/157]     Batch Accuracy 0.3594
    Evaluate: [106/157]     Batch Accuracy 0.3594
    Evaluate: [107/157]     Batch Accuracy 0.2969
    Evaluate: [108/157]     Batch Accuracy 0.3906
    Evaluate: [109/157]     Batch Accuracy 0.4062
    Evaluate: [110/157]     Batch Accuracy 0.2969
    Evaluate: [111/157]     Batch Accuracy 0.3281
    Evaluate: [112/157]     Batch Accuracy 0.3750
    Evaluate: [113/157]     Batch Accuracy 0.4219
    Evaluate: [114/157]     Batch Accuracy 0.4219
    Evaluate: [115/157]     Batch Accuracy 0.3438
    Evaluate: [116/157]     Batch Accuracy 0.3281
    Evaluate: [117/157]     Batch Accuracy 0.2969
    Evaluate: [118/157]     Batch Accuracy 0.4375
    Evaluate: [119/157]     Batch Accuracy 0.3125
    Evaluate: [120/157]     Batch Accuracy 0.3906
    Evaluate: [121/157]     Batch Accuracy 0.3281
    Evaluate: [122/157]     Batch Accuracy 0.4375
    Evaluate: [123/157]     Batch Accuracy 0.4219
    Evaluate: [124/157]     Batch Accuracy 0.3281
    Evaluate: [125/157]     Batch Accuracy 0.4219
    Evaluate: [126/157]     Batch Accuracy 0.4531
    Evaluate: [127/157]     Batch Accuracy 0.3594
    Evaluate: [128/157]     Batch Accuracy 0.3906
    Evaluate: [129/157]     Batch Accuracy 0.3281
    Evaluate: [130/157]     Batch Accuracy 0.4531
    Evaluate: [131/157]     Batch Accuracy 0.3281
    Evaluate: [132/157]     Batch Accuracy 0.4375
    Evaluate: [133/157]     Batch Accuracy 0.3906
    Evaluate: [134/157]     Batch Accuracy 0.3906
    Evaluate: [135/157]     Batch Accuracy 0.3594
    Evaluate: [136/157]     Batch Accuracy 0.4375
    Evaluate: [137/157]     Batch Accuracy 0.3594
    Evaluate: [138/157]     Batch Accuracy 0.3750
    Evaluate: [139/157]     Batch Accuracy 0.4062
    Evaluate: [140/157]     Batch Accuracy 0.4375
    Evaluate: [141/157]     Batch Accuracy 0.3906
    Evaluate: [142/157]     Batch Accuracy 0.3750
    Evaluate: [143/157]     Batch Accuracy 0.3906
    Evaluate: [144/157]     Batch Accuracy 0.4375
    Evaluate: [145/157]     Batch Accuracy 0.3594
    Evaluate: [146/157]     Batch Accuracy 0.4219
    Evaluate: [147/157]     Batch Accuracy 0.3594
    Evaluate: [148/157]     Batch Accuracy 0.3906
    Evaluate: [149/157]     Batch Accuracy 0.3125
    Evaluate: [150/157]     Batch Accuracy 0.4062
    Evaluate: [151/157]     Batch Accuracy 0.2812
    Evaluate: [152/157]     Batch Accuracy 0.2500
    Evaluate: [153/157]     Batch Accuracy 0.4531
    Evaluate: [154/157]     Batch Accuracy 0.3125
    Evaluate: [155/157]     Batch Accuracy 0.3594
    Evaluate: [156/157]     Batch Accuracy 0.3750
    Final Accuracy on Test Data: 0.3613
```

```
# Cell 19
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

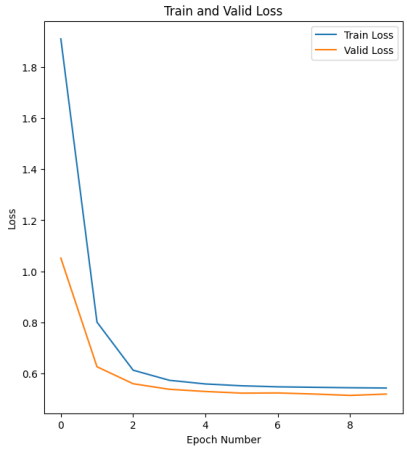


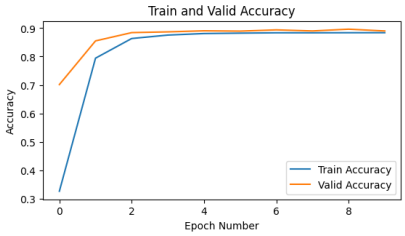



```
# Cell 20
# Change reg to 1e-2 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [145/157]     Batch Accuracy 0.9531
Evaluate: [146/157]     Batch Accuracy 0.9844
Evaluate: [147/157]     Batch Accuracy 0.8906
Evaluate: [148/157]     Batch Accuracy 0.9375
Evaluate: [149/157]     Batch Accuracy 0.8906
Evaluate: [150/157]     Batch Accuracy 0.8906
Evaluate: [151/157]     Batch Accuracy 0.8750
Evaluate: [152/157]     Batch Accuracy 0.7344
Evaluate: [153/157]     Batch Accuracy 0.8750
Evaluate: [154/157]     Batch Accuracy 0.8281
Evaluate: [155/157]     Batch Accuracy 0.8281
Evaluate: [156/157]     Batch Accuracy 1.0000
Final Accuracy on Test Data: 0.8935
```

```
# Cell 21
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

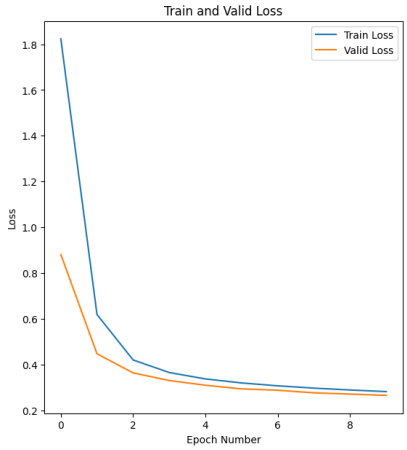


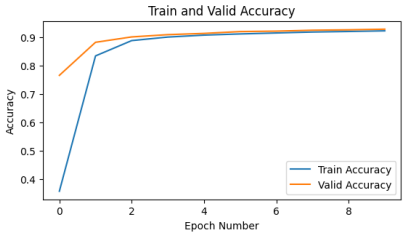



```
# Cell 22
# Change reg to 1e-3 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [128/157]     Batch Accuracy 0.9688
Evaluate: [129/157]     Batch Accuracy 0.9375
Evaluate: [130/157]     Batch Accuracy 0.9688
Evaluate: [131/157]     Batch Accuracy 0.9531
Evaluate: [132/157]     Batch Accuracy 0.9688
Evaluate: [133/157]     Batch Accuracy 0.9531
Evaluate: [134/157]     Batch Accuracy 0.9844
Evaluate: [135/157]     Batch Accuracy 1.0000
Evaluate: [136/157]     Batch Accuracy 1.0000
Evaluate: [137/157]     Batch Accuracy 1.0000
Evaluate: [138/157]     Batch Accuracy 0.9844
Evaluate: [139/157]     Batch Accuracy 1.0000
Evaluate: [140/157]     Batch Accuracy 0.8906
Evaluate: [141/157]     Batch Accuracy 0.9375
Evaluate: [142/157]     Batch Accuracy 1.0000
Evaluate: [143/157]     Batch Accuracy 0.9531
Evaluate: [144/157]     Batch Accuracy 1.0000
Evaluate: [145/157]     Batch Accuracy 0.9688
Evaluate: [146/157]     Batch Accuracy 1.0000
Evaluate: [147/157]     Batch Accuracy 0.9219
Evaluate: [148/157]     Batch Accuracy 0.9688
Evaluate: [149/157]     Batch Accuracy 0.9375
Evaluate: [150/157]     Batch Accuracy 0.9219
Evaluate: [151/157]     Batch Accuracy 0.9062
Evaluate: [152/157]     Batch Accuracy 0.8125
Evaluate: [153/157]     Batch Accuracy 0.9375
Evaluate: [154/157]     Batch Accuracy 0.8438
Evaluate: [155/157]     Batch Accuracy 0.8750
Evaluate: [156/157]     Batch Accuracy 0.9375
Final Accuracy on Test Data: 0.9262
```

```
# Cell 23
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

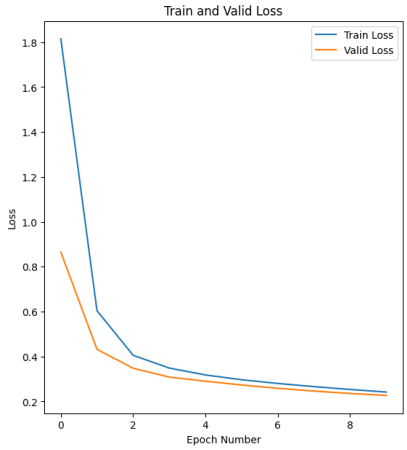


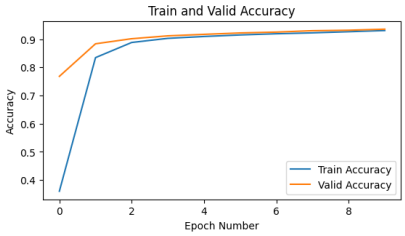



```
# Cell 24
# Change reg to 1e-4 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [111/157]     Batch Accuracy 0.9688
Evaluate: [112/157]     Batch Accuracy 0.9688
Evaluate: [113/157]     Batch Accuracy 1.0000
Evaluate: [114/157]     Batch Accuracy 0.9844
Evaluate: [115/157]     Batch Accuracy 1.0000
Evaluate: [116/157]     Batch Accuracy 0.9375
Evaluate: [117/157]     Batch Accuracy 0.9688
Evaluate: [118/157]     Batch Accuracy 1.0000
Evaluate: [119/157]     Batch Accuracy 1.0000
Evaluate: [120/157]     Batch Accuracy 1.0000
Evaluate: [121/157]     Batch Accuracy 0.9688
Evaluate: [122/157]     Batch Accuracy 0.8281
Evaluate: [123/157]     Batch Accuracy 0.8594
Evaluate: [124/157]     Batch Accuracy 0.9688
Evaluate: [125/157]     Batch Accuracy 0.9688
Evaluate: [126/157]     Batch Accuracy 0.9219
Evaluate: [127/157]     Batch Accuracy 0.9844
Evaluate: [128/157]     Batch Accuracy 0.9844
Evaluate: [129/157]     Batch Accuracy 0.9219
Evaluate: [130/157]     Batch Accuracy 0.9844
Evaluate: [131/157]     Batch Accuracy 0.9531
Evaluate: [132/157]     Batch Accuracy 0.9688
Evaluate: [133/157]     Batch Accuracy 0.9531
Evaluate: [134/157]     Batch Accuracy 0.9844
Evaluate: [135/157]     Batch Accuracy 1.0000
Evaluate: [136/157]     Batch Accuracy 1.0000
Evaluate: [137/157]     Batch Accuracy 1.0000
Evaluate: [138/157]     Batch Accuracy 0.9844
Evaluate: [139/157]     Batch Accuracy 1.0000
Evaluate: [140/157]     Batch Accuracy 0.8906
Evaluate: [141/157]     Batch Accuracy 0.9375
Evaluate: [142/157]     Batch Accuracy 0.9844
Evaluate: [143/157]     Batch Accuracy 0.9688
Evaluate: [144/157]     Batch Accuracy 1.0000
Evaluate: [145/157]     Batch Accuracy 0.9688
Evaluate: [146/157]     Batch Accuracy 1.0000
Evaluate: [147/157]     Batch Accuracy 0.9375
Evaluate: [148/157]     Batch Accuracy 0.9688
Evaluate: [149/157]     Batch Accuracy 0.9375
Evaluate: [150/157]     Batch Accuracy 0.9219
Evaluate: [151/157]     Batch Accuracy 0.9219
Evaluate: [152/157]     Batch Accuracy 0.7812
Evaluate: [153/157]     Batch Accuracy 0.9375
Evaluate: [154/157]     Batch Accuracy 0.8281
Evaluate: [155/157]     Batch Accuracy 0.9062
Evaluate: [156/157]     Batch Accuracy 0.9375
Final Accuracy on Test Data: 0.9343
```

```
# Cell 25
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```

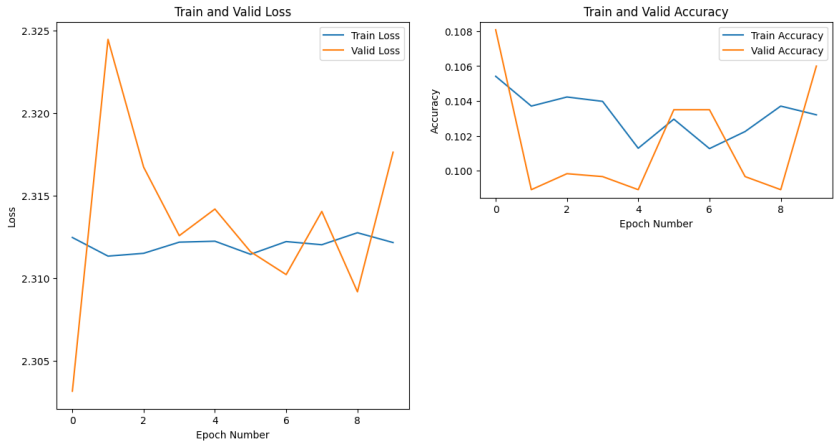





```
# Cell 26
# Change reg to 1 in the config file and run this code block
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
Evaluate: [145/157]     Batch Accuracy 0.1562
Evaluate: [146/157]     Batch Accuracy 0.0625
Evaluate: [147/157]     Batch Accuracy 0.1250
Evaluate: [148/157]     Batch Accuracy 0.0938
Evaluate: [149/157]     Batch Accuracy 0.1250
Evaluate: [150/157]     Batch Accuracy 0.1406
Evaluate: [151/157]     Batch Accuracy 0.0625
Evaluate: [152/157]     Batch Accuracy 0.1406
Evaluate: [153/157]     Batch Accuracy 0.0938
Evaluate: [154/157]     Batch Accuracy 0.1719
Evaluate: [155/157]     Batch Accuracy 0.1094
Evaluate: [156/157]     Batch Accuracy 0.0625
Final Accuracy on Test Data: 0.1028
```

```
# Cell 27
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```



Describe and explain your findings here:

After running the previous cells, I found that the regularization coefficient plays a huge role in the loss and accuracy of the model. Similar to learning rate, there was a particular range for the coefficient that caused the most optimal results from the model.

With a relatively high reg. coefficient of 0.1, this means that the regularization function will have a strong penalty for large weights for training (based on our implementation). This may prevent overfitting in some cases, but too much of this penalty caused some underfitting and the model was relatively inaccurate as well (mostly happens because complex patterns cannot be recorded with a high reg. coefficient).

With a reg. coefficient of 0.01, I thought the difference would be huge. However, this coefficient still seemed to be pretty high. There was no overfitting and the model was able to converge after some batches.

With a reg. coefficient of 0.001, initially I perceived this value as being too low. After experimenting, however, I found that this is probably the best starting point for our interval, as this value is able to balance both regularizing the data and capturing complex patterns that might occur.

With a reg. coefficient of 0.0001, the value has low regularization strength as the penalty was almost non-existent. While this means that the model has more freedom to learn from the training data, this graph most likely will suffer from overfitting when fed other data sets.

With a reg. coefficient of 1, it was super high and lead to severely strong regularization. This was most likely too restrictive for the data, causing tons of variability and the graph never converged on a single point.

Now, when performing hypertuning on the regular coefficient parameter, I will most likely consider an interval with 0.001 at the center, as this worked the best from all the values. I will also consider values like 0.0001 but will make sure that the model doesn't overfit. Most of these hyperparameters are case-dependent, as for this experiment I will have to continue trying values until I can achieve the most optimal accuracy on the dataset.

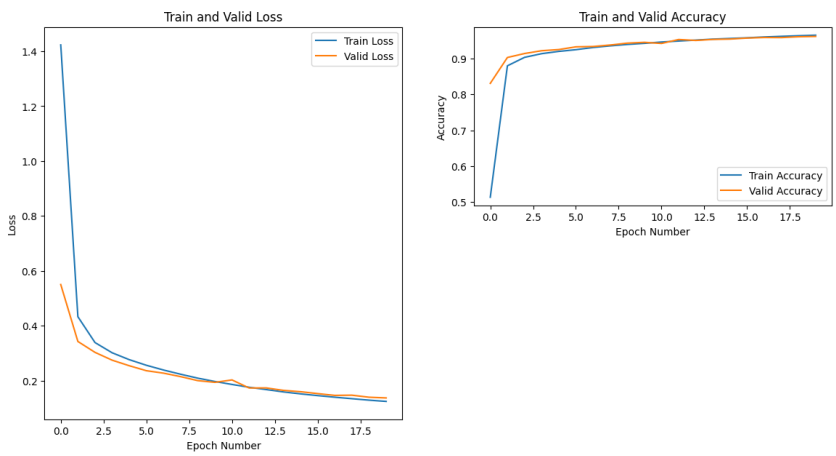
Hyper-parameter Tuning

You are now free to tune any hyperparameters for better accuracy. In this block type the configuration of your best model and provide a brief explanation of why it works.

```
# Cell 28
# hyperparameter tuning
train_loss_history, train_acc_history, valid_loss_history, valid_acc_history =
```

```
    Evaluate: [105/162]     Batch Accuracy 0.9194
    Evaluate: [106/162]     Batch Accuracy 0.9032
    Evaluate: [107/162]     Batch Accuracy 0.9839
    Evaluate: [108/162]     Batch Accuracy 1.0000
    Evaluate: [109/162]     Batch Accuracy 0.9355
    Evaluate: [110/162]     Batch Accuracy 0.9839
    Evaluate: [111/162]     Batch Accuracy 0.9839
    Evaluate: [112/162]     Batch Accuracy 1.0000
    Evaluate: [113/162]     Batch Accuracy 1.0000
    Evaluate: [114/162]     Batch Accuracy 1.0000
    Evaluate: [115/162]     Batch Accuracy 1.0000
    Evaluate: [116/162]     Batch Accuracy 0.9839
    Evaluate: [117/162]     Batch Accuracy 1.0000
    Evaluate: [118/162]     Batch Accuracy 1.0000
    Evaluate: [119/162]     Batch Accuracy 0.9677
    Evaluate: [120/162]     Batch Accuracy 0.9677
    Evaluate: [121/162]     Batch Accuracy 1.0000
    Evaluate: [122/162]     Batch Accuracy 1.0000
    Evaluate: [123/162]     Batch Accuracy 1.0000
    Evaluate: [124/162]     Batch Accuracy 1.0000
    Evaluate: [125/162]     Batch Accuracy 0.9839
    Evaluate: [126/162]     Batch Accuracy 0.9839
    Evaluate: [127/162]     Batch Accuracy 0.9516
    Evaluate: [128/162]     Batch Accuracy 0.9839
    Evaluate: [129/162]     Batch Accuracy 0.9839
    Evaluate: [130/162]     Batch Accuracy 0.9516
    Evaluate: [131/162]     Batch Accuracy 0.9839
    Evaluate: [132/162]     Batch Accuracy 1.0000
    Evaluate: [133/162]     Batch Accuracy 0.9839
    Evaluate: [134/162]     Batch Accuracy 0.9677
    Evaluate: [135/162]     Batch Accuracy 0.9677
    Evaluate: [136/162]     Batch Accuracy 1.0000
    Evaluate: [137/162]     Batch Accuracy 0.9677
    Evaluate: [138/162]     Batch Accuracy 1.0000
    Evaluate: [139/162]     Batch Accuracy 1.0000
    Evaluate: [140/162]     Batch Accuracy 1.0000
    Evaluate: [141/162]     Batch Accuracy 1.0000
    Evaluate: [142/162]     Batch Accuracy 0.9839
    Evaluate: [143/162]     Batch Accuracy 1.0000
    Evaluate: [144/162]     Batch Accuracy 1.0000
    Evaluate: [145/162]     Batch Accuracy 0.8710
    Evaluate: [146/162]     Batch Accuracy 0.9839
    Evaluate: [147/162]     Batch Accuracy 1.0000
    Evaluate: [148/162]     Batch Accuracy 1.0000
    Evaluate: [149/162]     Batch Accuracy 0.9839
    Evaluate: [150/162]     Batch Accuracy 1.0000
    Evaluate: [151/162]     Batch Accuracy 1.0000
    Evaluate: [152/162]     Batch Accuracy 0.9839
    Evaluate: [153/162]     Batch Accuracy 1.0000
    Evaluate: [154/162]     Batch Accuracy 0.9839
    Evaluate: [155/162]     Batch Accuracy 0.9516
    Evaluate: [156/162]     Batch Accuracy 0.8871
    Evaluate: [157/162]     Batch Accuracy 0.8871
    Evaluate: [158/162]     Batch Accuracy 0.9516
    Evaluate: [159/162]     Batch Accuracy 0.9355
    Evaluate: [160/162]     Batch Accuracy 0.9677
    Evaluate: [161/162]     Batch Accuracy 0.9444
    Final Accuracy on Test Data: 0.9627
```

```
%matplotlib inline
plot_curves(train_loss_history, train_acc_history, valid_loss_history, valid_acc
```




```
# Cell 30
#To collect submission
!sh collect_submission.sh
```

Describe and explain your findings here:

What I found from hours of testing and changing the hyperparameters is that hyper tuning to find the right values for each parameter is excruciating difficult and time-consuming. Every time you change one, you need to alter the others slightly to get a better accuracy or improvement on the model.

Some observations I made is that when I tried to increase the learning rate more or tried to decrease the regularization coefficient, I found that the accuracy for the validation dataset would stay the same or be less than the training dataset. This was most likely due to overfitting so I had a very small margin in which I could alter the hyperparameters. Additionally, I changed the epoch size and played with the batch size in order to figure out how that would impact the model. From my experimenting I found that a epoch number of 20 helped train the dataset more. I'm sure whether or not I was allowed to alter this term