Brandon Withington
CS434 HW3
5 / 17/ 2021

Q1.)

▶ **Q1 Prove Bernoulli Naïve Bayes has a linear decision boundary [4pts].** Prove the Bernoulli Naïve Bayes model described above has a linear decision boundary. Specifically, you'll need to show that class 1 will be predicted only if $b + \mathbf{w}^T\mathbf{x} > 1$ for some parameters b and $\mathbf{w}$. To do so, show that:

$$\frac{P(y=1|x_1,...,x_d)}{P(y=0|x_1,...,x_d)} > 1 \implies b + \sum_{i=1}^{d} w_i x_i > 0 \tag{4}$$

As part of your report, explicitly write expressions for the bias b and each weight w_i.

Hints: Expand Eq. (3) by substituting the posterior expressions from Eq. (1) & (2) into Eq. (3). Take the log of that expression and combine like-terms.

Posterior of our classifier without the normalized constant :

$$P(y=1 \mid x_1, x_2, \ldots.. x_n) = \frac{P(y=1)\prod_{i=1}^{n} P(xi \mid y=1)}{P(x_1, x_2, \ldots.. x_n)}$$

$$P(y=1 \mid x_1, x_2, \ldots.. x_n) = \theta_1 \prod_{i=1}^{n} \theta_1^{xi}(1 - \theta i_1)\, 1 - xi$$ this is our first equation

$$P(y=0 \mid x_1, x_2, \ldots.. x_n) = \frac{P(y=0)\prod_{i=1}^{n} P(xi \mid y=0)}{P(x_1, x_2, \ldots.. x_n)}$$

$$P(y=0 \mid x_1, x_2, \ldots.. x_n) = \theta_0 \prod_{i=1}^{n} \theta_0^{xi}(1 - \theta i_0)\, 1 - xi$$ is our second equation

$$\frac{P(y=1 \mid x_1, x_2, \ldots.. x_n)}{P(y=0 \mid x_1, x_2, \ldots.. x_n)} > 1 \Rightarrow b + \sum_{i=1}^{n} w_i x_i > 0$$

We first consider :

$$\frac{P\,(\,y=1\mid x_1, x_2, \ldots.. x_n)}{P\,(\,y=0\mid x_1, x_2, \ldots.. x_n)} > 1 \quad \text{(equation three)}$$

Plugging in our first two equations into our third, our result is :

$$\frac{\theta_1 \prod_{i=1}^{n} {\theta_1}^{xi}(1-\theta i_1)\,1-xi}{\theta_0 \prod_{i=1}^{n} {\theta_0}^{xi}(1-\theta i_0)\,1-xi} > 1 \quad == \left(\frac{\theta_1}{\theta_0}\right) \frac{\prod_{i=1}^{n} {\theta_1}^{xi}(1-\theta i_1)\,1-xi}{\prod_{i=1}^{n} {\theta_0}^{xi}(1-\theta i_0)\,1-xi} > 1$$

$$\left(\frac{\theta_1}{\theta_0}\right) \frac{\prod_{i=1}^{n} {\theta_1}^{xi}(1-\theta i_1)\,1-xi}{\prod_{i=1}^{n} {\theta_0}^{xi}(1-\theta i_0)\,1-xi} > 1 == \left(\frac{\theta_1}{\theta_0}\right) \frac{\prod_{i=1}^{n} \left(\frac{\theta_1}{1-\theta i_1}\right)^{xi}(1-\theta i_1)}{\prod_{i=1}^{n} \left(\frac{\theta_0}{1-\theta i_0}\right)^{xi}(1-\theta i_0)} > 1$$

After this we take the log of : $\left(\frac{\theta_1}{\theta_0}\right) \frac{\prod_{i=1}^{n} \left(\frac{\theta_1}{1-\theta i_1}\right)^{xi}(1-\theta i_1)}{\prod_{i=1}^{n} \left(\frac{\theta_0}{1-\theta i_0}\right)^{xi}(1-\theta i_0)} > 1$

Outcome : $\log\left(\frac{\theta_1}{\theta_0}\right) + \frac{\prod_{i=1}^{n} log\left[\left(\frac{\theta_1}{1-\theta i_1}\right)^{xi}(1-\theta i_1)\right]}{\prod_{i=1}^{n} log\left[\left(\frac{\theta_0}{1-\theta i_0}\right)^{xi}(1-\theta i_0)\right]} > 0$

$$\log\left(\frac{\theta_1}{\theta_0}\right) +$$

$$\sum_{i=1}^{n} log\left[\left(\frac{\theta i_1}{1-\theta i_1}\right)^{xi}(1-\theta i_1)\right] - \sum_{i=1}^{n} log\left[\left(\frac{\theta i_0}{1-\theta i_0}\right)^{xi}(1-\theta i_0)\right] > 0$$

=

$\log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{n} log[\left(\frac{\theta i_1}{\frac{\theta i_0}{1-\theta i_0}}\right)^{xi}\left(\frac{1-\theta i_1}{1-\theta i_0}\right)] > 0$

=

$\log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{n} [log\left(\frac{\theta i_1(1-\theta i_{0)}}{\theta i_0(1-\theta i_1)}\right)^{xi} log\left(\frac{1-\theta i_1}{1-\theta i_0}\right)] > 0$

=

$\log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{n} xi * [log\left(\frac{\theta i_1(1-\theta i_{0)}}{\theta i_0(1-\theta i_1)}\right) + log\left(\frac{1-\theta i_1}{1-\theta i_0}\right)] > 0$ is our 4th equation

Comparing $\log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{n} xi * [log\left(\frac{\theta i_1(1-\theta i_{0)}}{\theta i_0(1-\theta i_1)}\right) log\left(\frac{1-\theta i_1}{1-\theta i_0}\right)] > 0$ to :

$$b + \sum_{i=1}^{d} w_i x_i > 0$$

b = $\log\left(\frac{\theta_1}{\theta_0}\right)$ **is our bias**

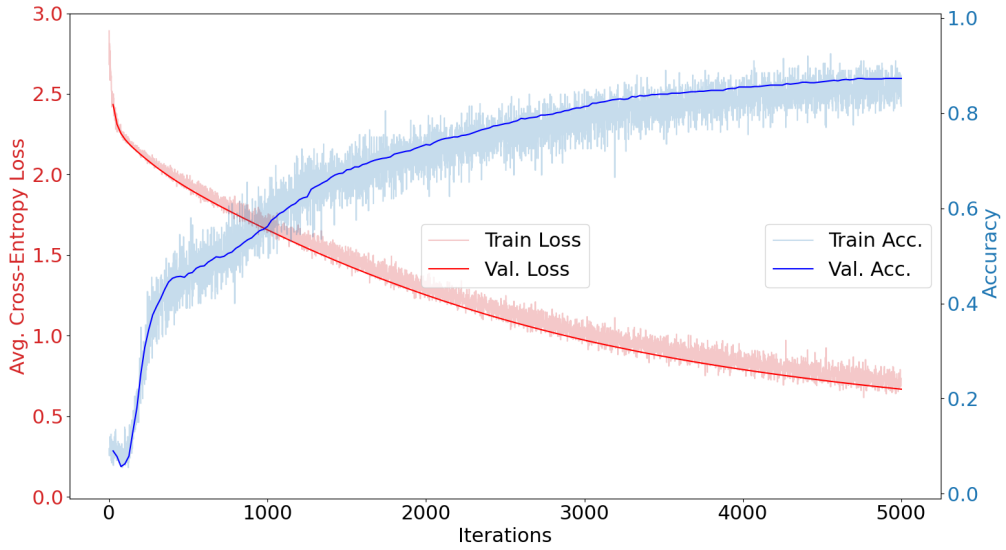
$w_i = log[\left(\frac{\theta i_1(1-\theta i_{0)}}{\theta i_0(1-\theta i_1)}\right) * \left(\frac{1-\theta i_1}{1-\theta i_0}\right)] \Rightarrow w_i = \log\left[\frac{\theta i_1}{\theta i_0}\right]$ **which is our weight.**

Q3.) Is code implementation for the backward pass

Code :

```
def backward(self, grad):
  self.grad_weights = self.input.T@grad
  self.grad_bias = np.sum(grad, axis=0)[np.newaxis,:]
  return grad@self.weights.T
```

Result from running skeleton code :



Q4.)

▶ Q4 Learning Rate [2pts]. The learning rate (or step size) in stochastic gradient descent controls how large of a step in the direction of the loss gradient we take our parameters at each iteration. The batch size determines how many data points we use to estimate the gradient. Modify the hyperparameters to run the following experiments:

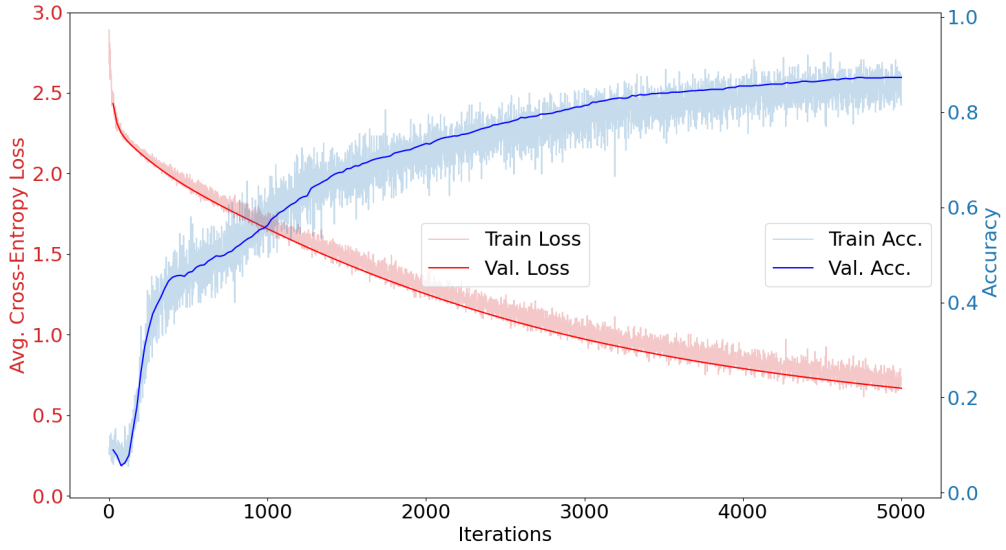
1. Step size of 0.0001 (leave default values for other hyperparameters)
2. Step size of 5 (leave default values for other hyperparameters)
3. Step size of 10 (leave default values for other hyperparameters)

Include these plot in your report and answer the following questions:

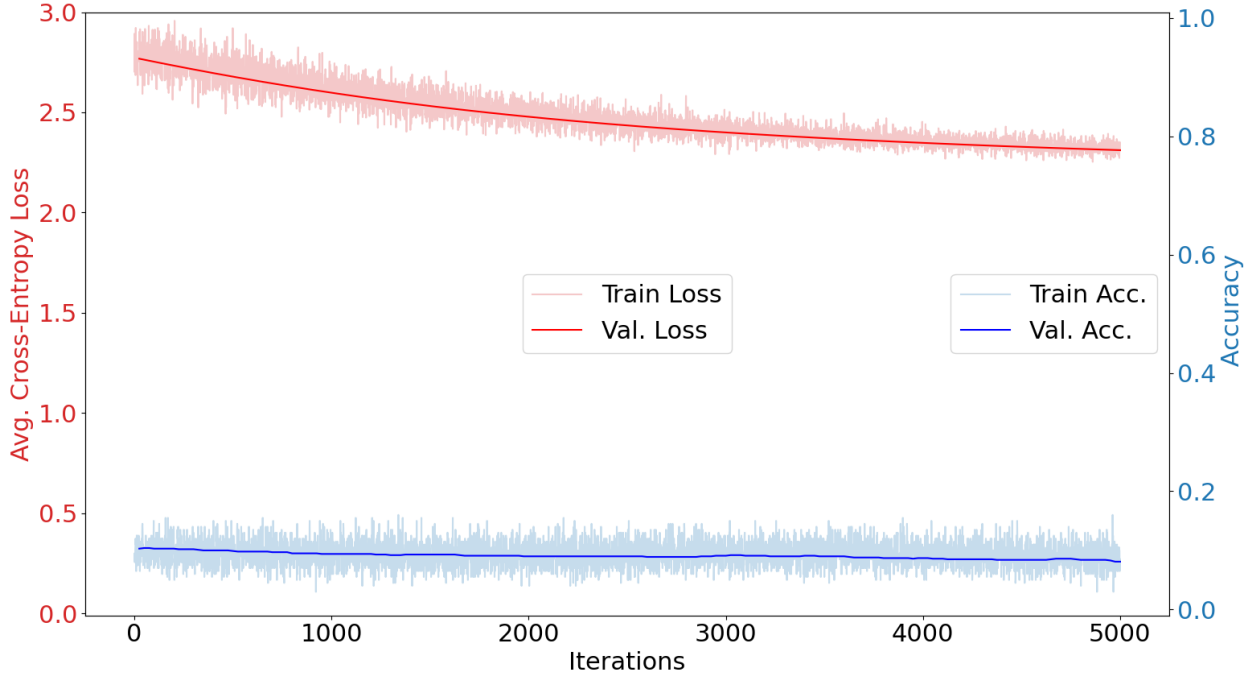
a) Compare and contrast the learning curves with your curve using the default parameters. What do you observe in terms of smoothness, shape, and what performance they reach?

b) For (a), what would you expect to happen if the max epochs were increased?

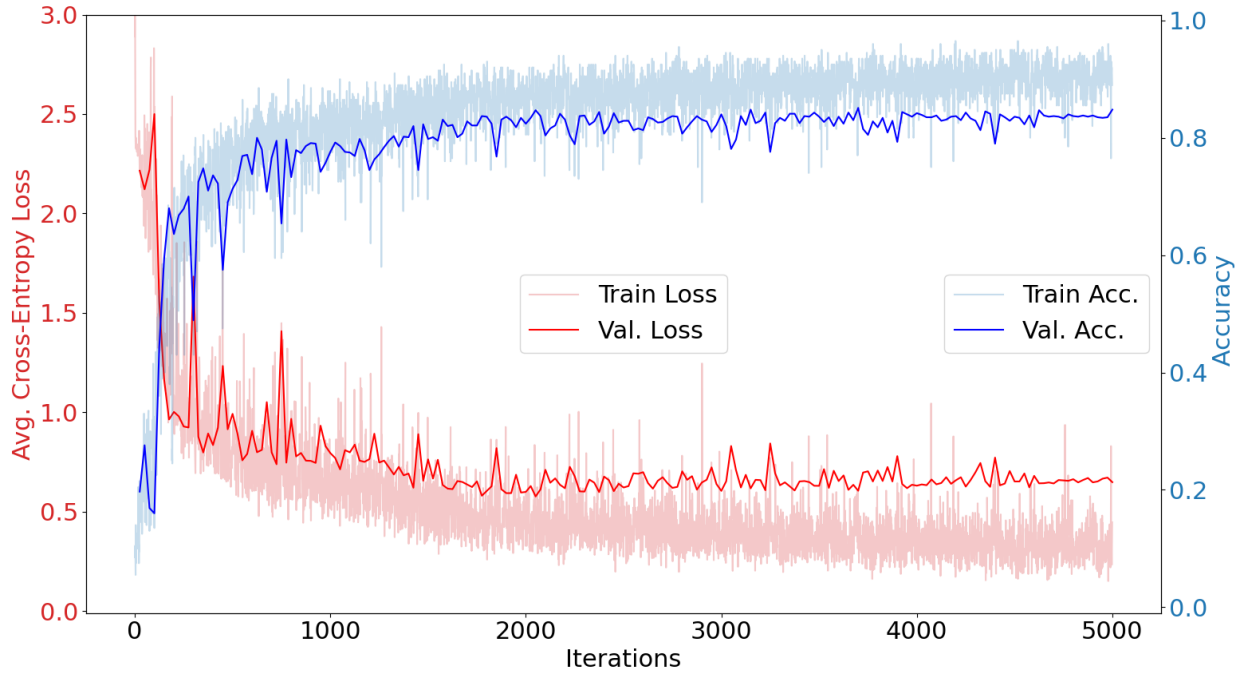
Pure default parameters :



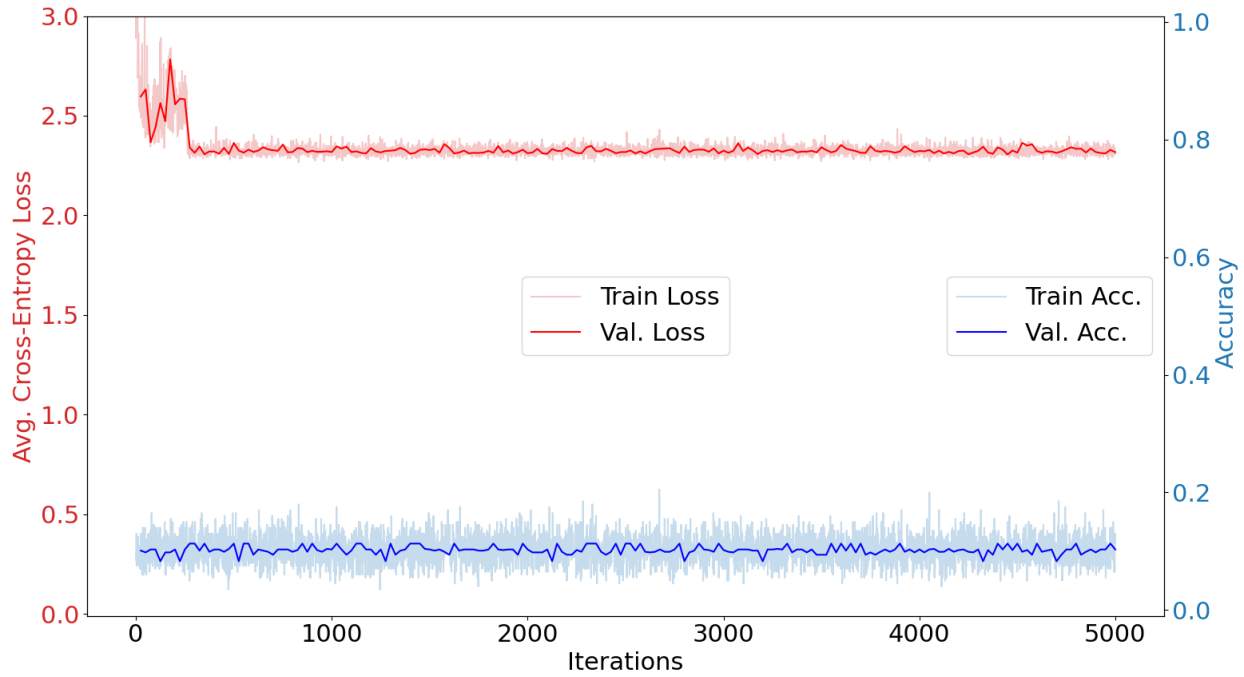
Step Size set at 0.0001 :



Step Size set at 5 :



Step Size set at 10 :



a.) Adding two more zeroes onto the default step size of 0.01 seems to have entirely separated the training loss line from the training accuracy line. The overall accuracy appears to have decreased. The same thing appears to have happened with step size set to 10, while those lines are more flat there are a couple ridges that boast higher accuracy than that of step size 0.0001. When I set the step size to 5 it was really interesting to see the accuracy and the fit of the line change in such a way that I did not expect.

b.) Training accuracy should go up as training loss goes down as the maximum number of epochs increases over time. The default configuration for our hyper parameters and step sizes set to 5 with all default parameters showcase this concept.

▶ Q5 ReLU's and Vanishing Gradients [3pts]. Modify the hyperparameters to run the following experiments:

1. 5-layer with Sigmoid Activation (leave default values for other hyperparameters)
2. 5-layer with Sigmoid Activation with 0.1 step size (leave default values for other hyperparameters)
3. 5-layer with ReLU Activation (leave default values for other hyperparameters)

Include these plot in your report and answer the following questions:

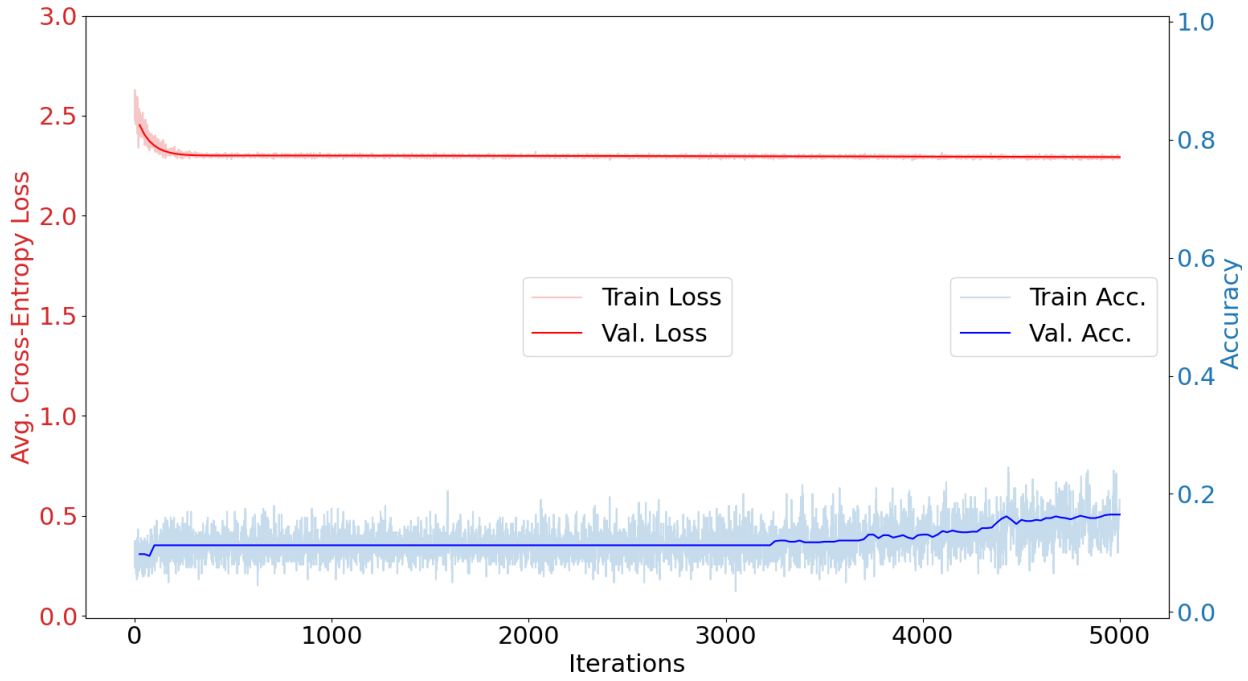
a) Compare and contrast the learning curves you observe and the curve for the default parameters in terms of smoothness, shape, and what performance they reach. Do you notice any differences in the relationship between the train and validation curves in each plot?

b) If you observed increasing the learning rate in (2) improves over (1), why might that be?

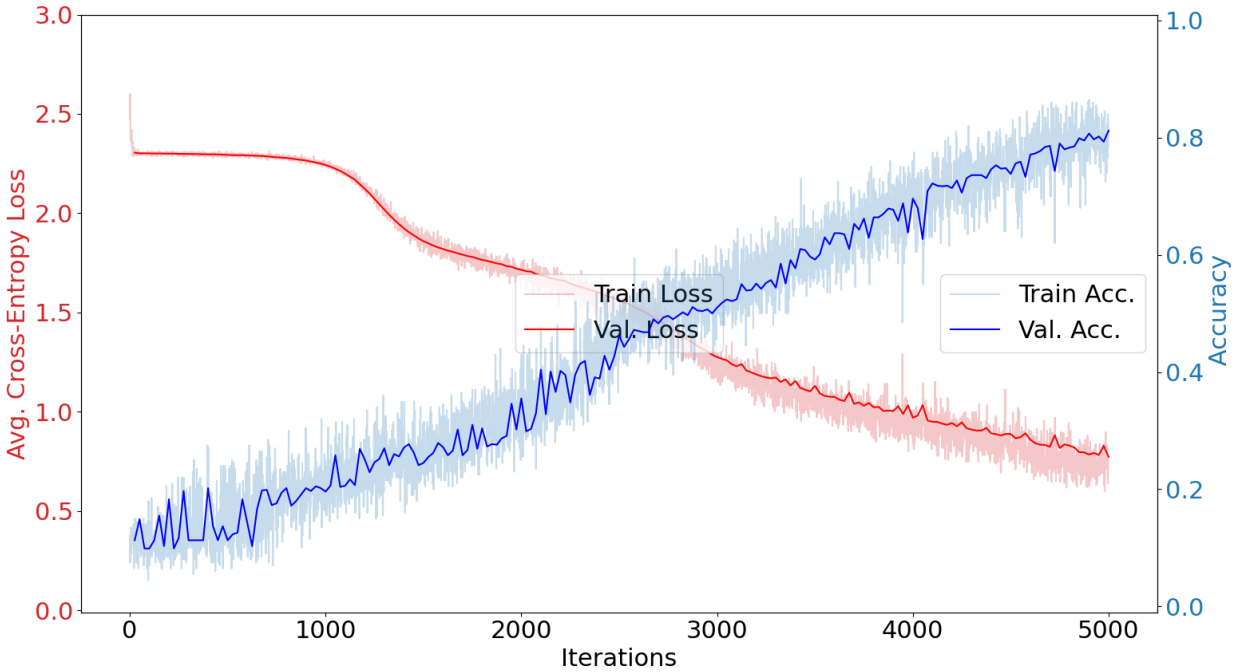
c) If (3) outperformed (1), why might that be? Consider the derivative of the sigmoid and ReLU functions.

Q5.)

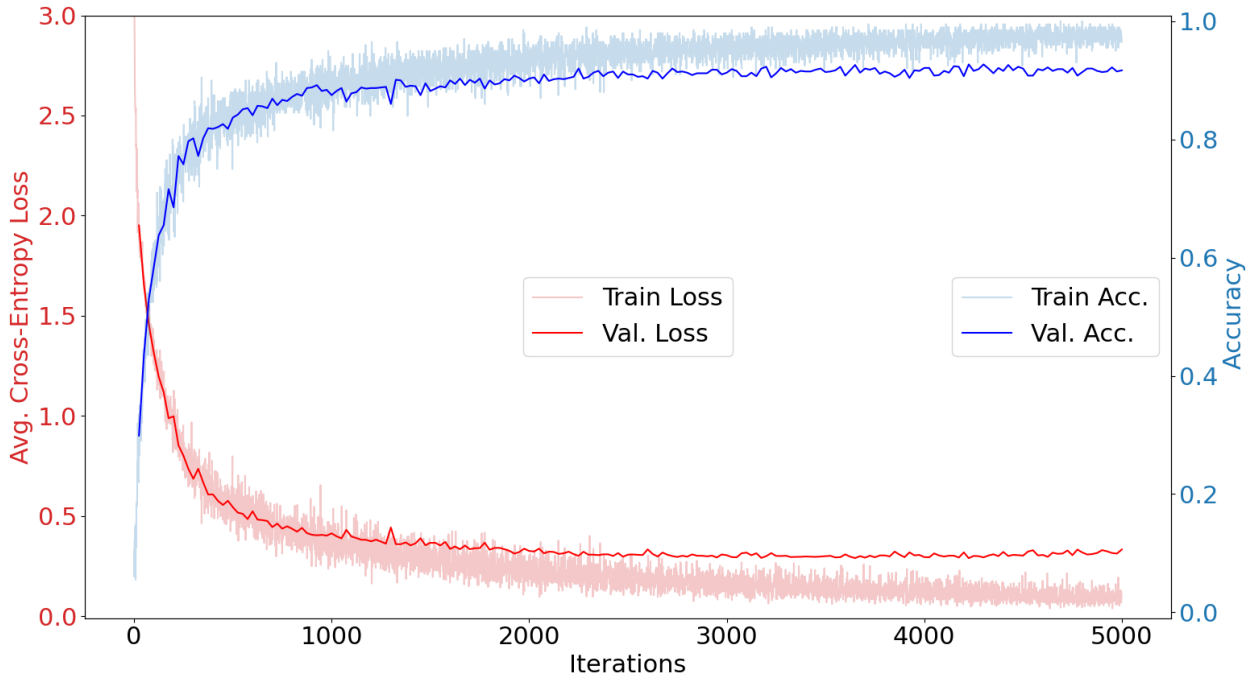
1. 5-layer with Sigmoid activation (Default values for other hyperparameters)



2. 5-layer with Sigmoid activation at step size 0.1



3. 5-layer with ReLU activation (Default values for all others)



a.) Comparing 1 and 2, the step size appears to dramatically impact the output of the sigmoid activation process, while 1 is smoother and is more akin to a straight line, 2 shows large amounts of ridges in the training accuracy. Interestingly the training loss curve does not show that many ridges. It is worth noting that training accuracy in 2 goes up gradually over the course of the running iterations. There was great differences between how 1 and 2 are related to each other, as with 1 there does not appear to be too much related to each other while 2 it is clear that as training loss decreases over time the training accuracy and value accuracy overall go up.

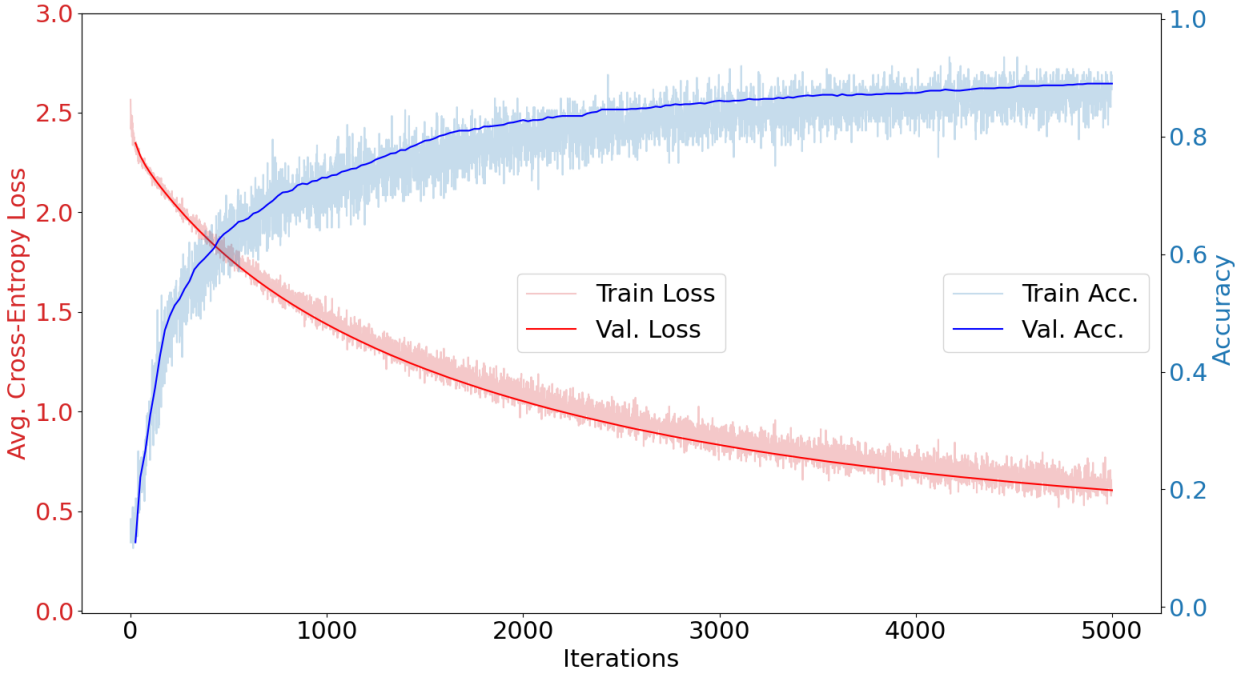
b.) Increasing the step size will increase the total range of the gradient of which it considers. Limiting the step size to a smaller number results in our program getting stuck in particular areas.

c.) 3 did outperform 1. Because looking at both the sigmoid and ReLU classes within our code we can see that one of the larger differences between them is how during the forward pass the sigmoid takes and returns : `1/(1+np.exp(-input))`
while the ReLU class takes `self.mask = (input > 0)`: and returns the value of input multiplied by this value. Which would be the max(0, input). The way both of these classes take their backward pass is different as well. Sigmoid computes a whole new gradient with respect to the input at self.act*(1-self.act) and multiplies itself by the loss gradient with respect to the output that which produces the loss gradient while ReLU makes masks out of the same elements from its forward.

▶ Q6 Measuring Randomness [1pt]. Using the default hyperparameters, set the random seed to 5 different values and report the validation accuracies you observe after training. What impact does this randomness have on the certainty of your conclusions in the previous questions?

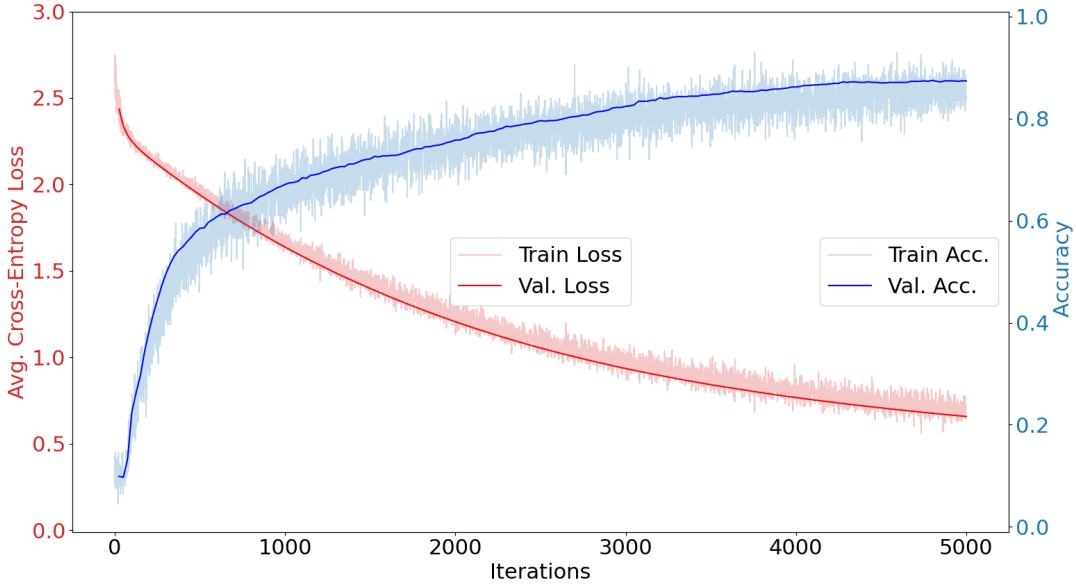
Q6.)

Random seed set to : 24 (all default parameters)



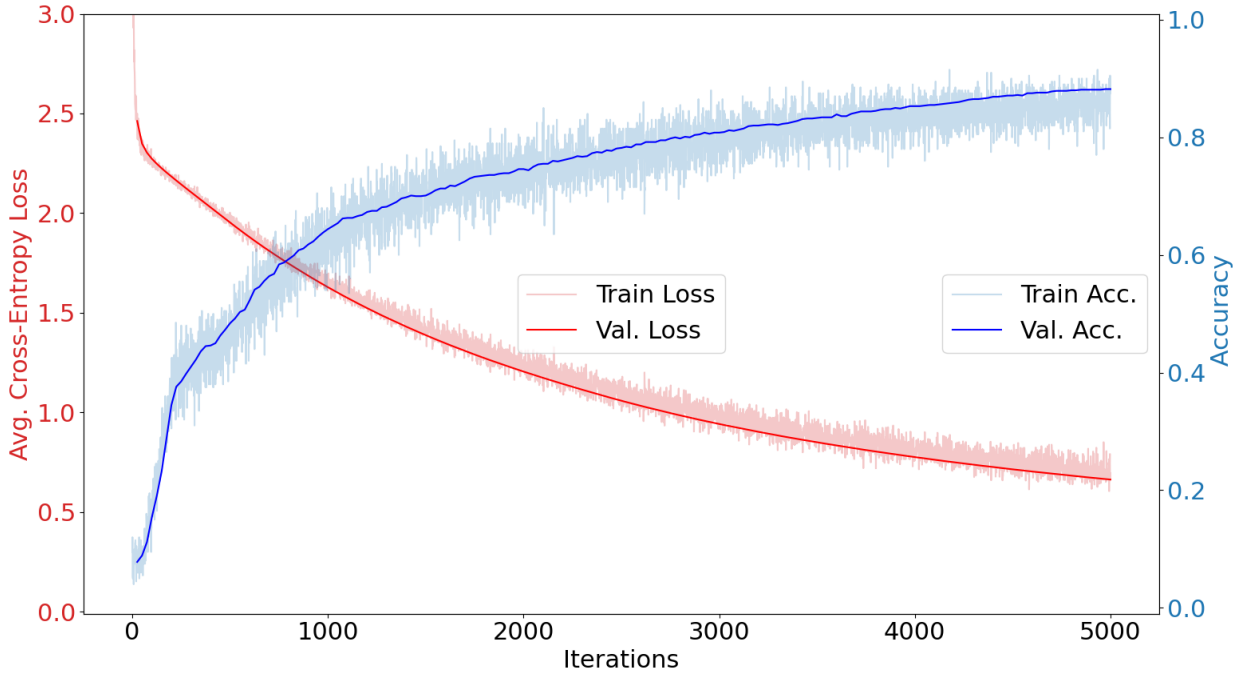
Validation accuracy : 89%

Random seed set to : 96 (all default parameters)



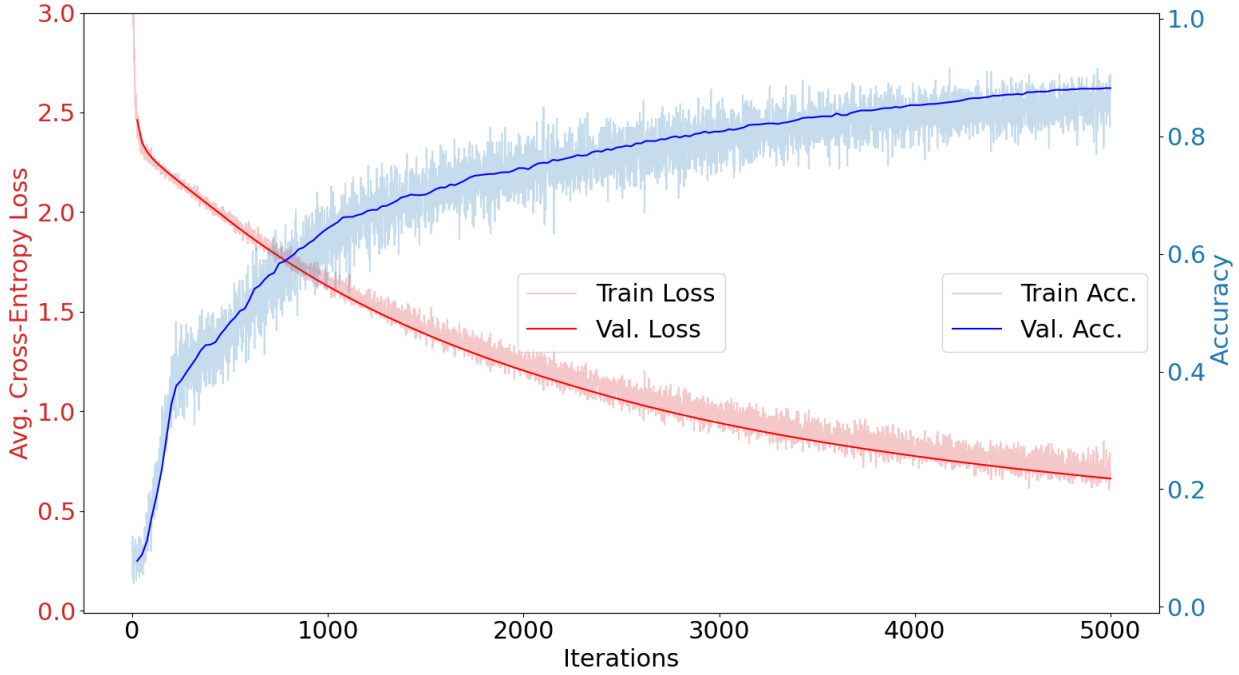
Validation accuracy : 87.4%

Random seed set to : 242(all default parameters)



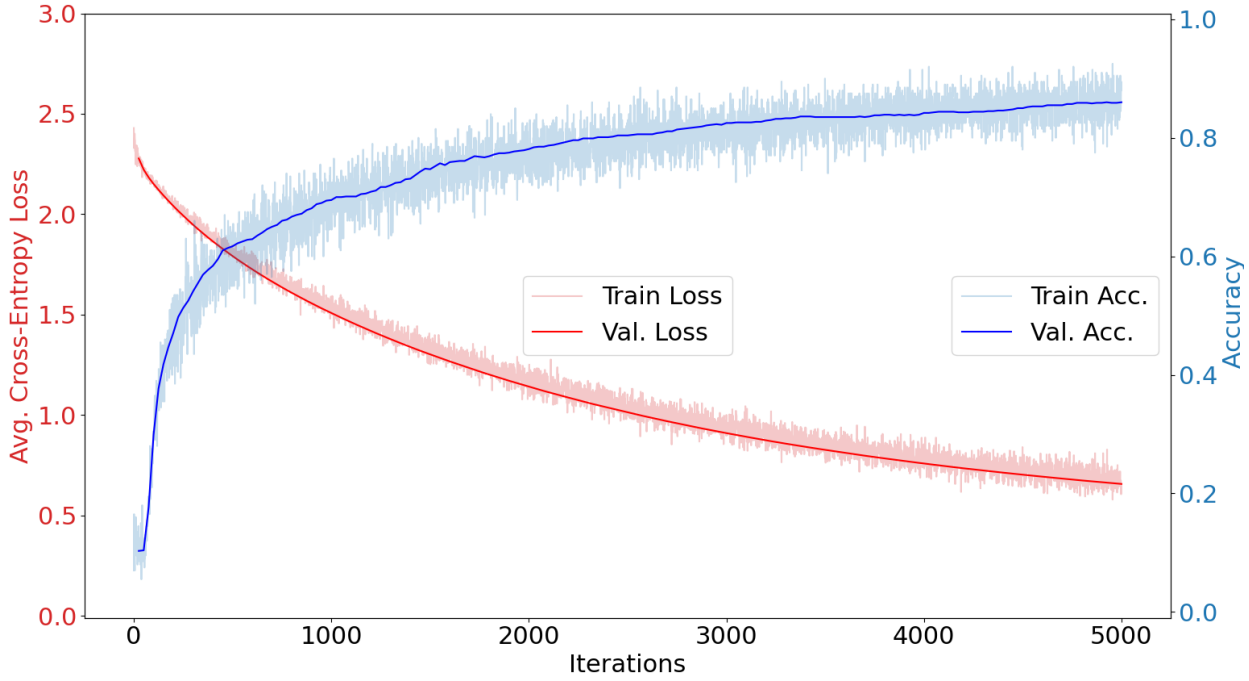
Validation accuracy : 88.2%

Random seed set to : 2467 (all default parameters)



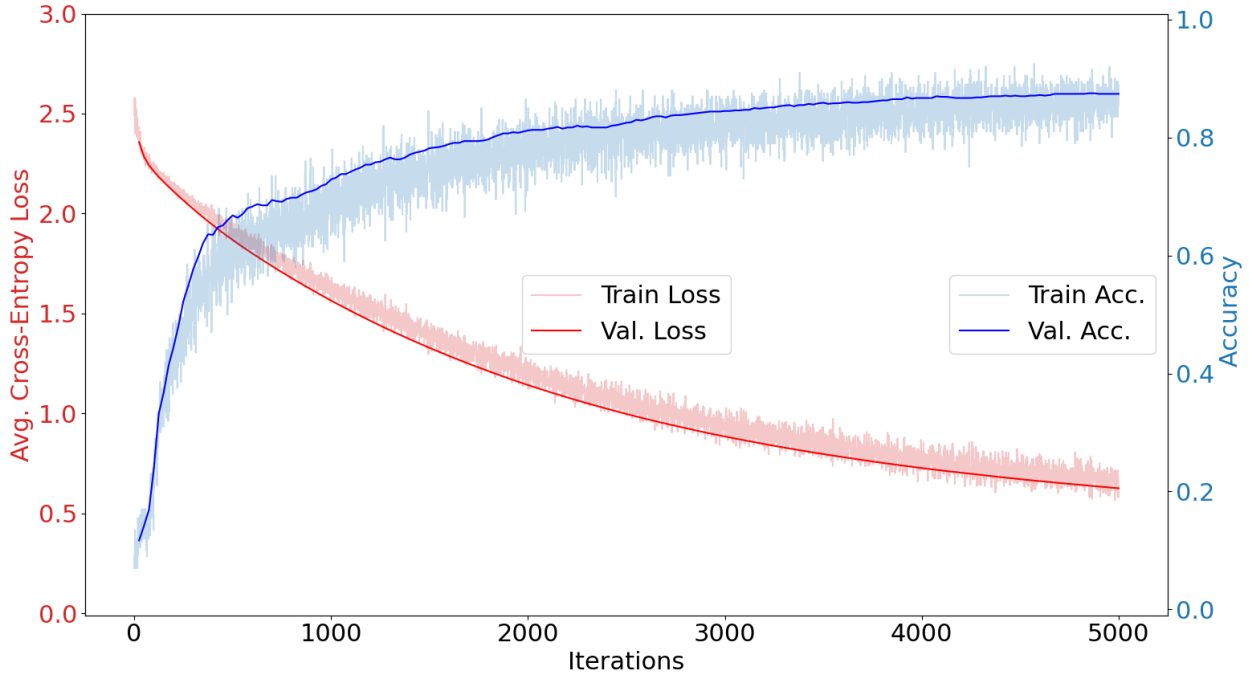
Validation accuracy :
87.1%

Random seed set to : 5432 (all default parameters)



Validation accuracy : 86%

Random seed set to : 9001 (all default parameters)



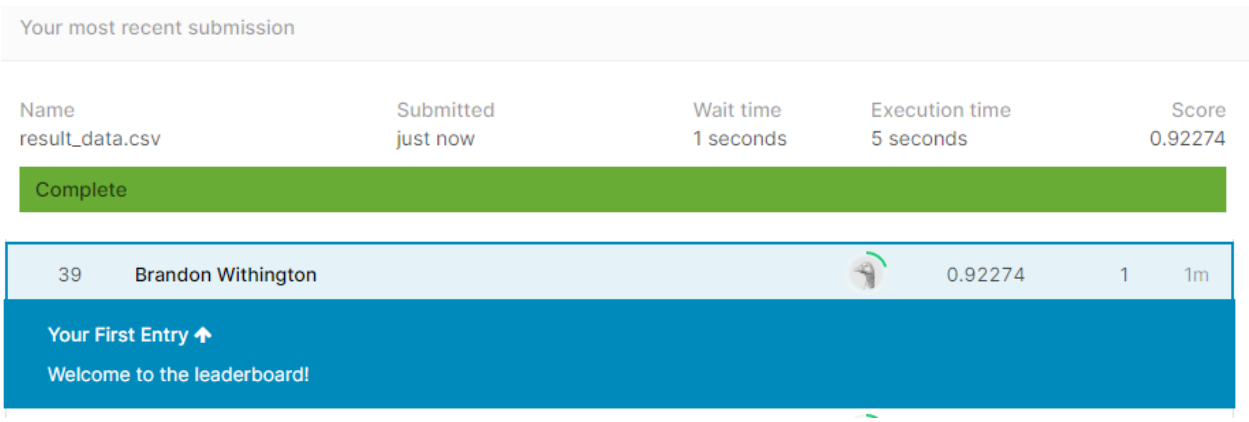
Validation accuracy : 87.4%

There does not appear to be any kind of correlation with the random seed number and the validation accuracy end number. The random seed does impact the training loss sometimes however, in the graphs the first few iterations sometimes will have a longer tail in the training loss gradient.

▶ Q7 Kaggle Submission [8pt]. Submit a set of predictions to Kaggle that outperforms the baseline on the public leaderboard. To make a valid submission, use the train set to train your neural network classifier and then apply it to the test instances in `mnist_small_test.csv` available from Kaggle's Data tab. Format your output as a two-column CSV as below:

```
id,digit
0,3
1,9
2,4
3,1
.
.
```

where the id is just the row index in `mnist_small_test.csv`. You may submit up to 10 times a day. In your report, tell us what modifications you made for your final submission.



Debrief :

1.) Approximately 16 hours.
2.) Moderate, the math at the beginning felt pretty long.
3.) Alone.
4.) I understand probably 60% of it deeply.
5.) Comparing and contrasting different hyperparameters is very interesting. I really like seeing the actual change in the graph as we change hyperparameters. That part is the most interesting to me!