a.d)

$$\frac{P(y=1|x_1 \dots x_d)}{P(y=0|x_1 \dots x_d)} > 1 \Rightarrow b + \sum_{i=1}^{d} w_i x_i > 0$$

$$\frac{P(y=1|x_1 \dots x_d)}{P(y=0|x_1 \dots x_d)} > 1 = \frac{P(y=1)\prod_{i=1}^{d} P(x_i|y=1) / P(x_1 \dots x_d)}{P(y=0)\prod_{i=1}^{d} P(x_i|y=0) / P(x_1 \dots x_d)} > 1 =$$

$$\frac{\phi_1 \prod_{i=1}^{d} \phi_{i1}^{x_i}(1-\phi_{i1})^{1-x_i}}{\phi_0 \prod_{i=1}^{d} \theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}} > 1 = \log\left(\frac{\phi_1 \prod_{i=1}^{d} \theta_{i1}^{x_i}(1-\theta_{i1})^{1-x_i}}{\phi_0 \prod_{i=1}^{d} \theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}}\right) > 0 =$$

$$\log(\phi_1) + \sum_{i=1}^{d} \log\left(\theta_{i1}^{x_i}(1-\theta_{i1})^{1-x_i}\right) - \log(\phi_0) + \sum_{i=1}^{d} \log\left(\theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}\right) > 0 =$$

$$\log(\phi_1) + \sum_{i=1}^{d} x_i \log(\theta_{i1}) + (1-x_i)\log(1-\theta_{i1}) - \log(\phi_0) + \sum_{i=1}^{d} x_i \log(\theta_{i0}) + (1-x_i)\log(1-\theta_{i0})$$

$$\log(\phi_1) - \log(\phi_0) + \sum_{i=1}^{d} x_i\left(\log(\theta_{i1}) - \log(\theta_{i0})\right) + (1-x_i)\left(\log(1-\theta_{i1}) - \log(1-\theta_{i0})\right) > 0 \rightarrow$$

$$\log(\phi_1) - \log(\phi_0) + d + \sum_{i=1}^{d} x_i\left(\log(\theta_{i1}) + \log(\theta_{i0}) - \log(1-\theta_{i1}) + \log(1-\theta_{i0})\right) > 0 \rightarrow$$

$$b = \log(\phi_1) - \log(\phi_0) + d, \quad w_i = \log(\theta_{i1}) - \log(\theta_{i0}) - \log(1-\theta_{i1}) + \log(1-\theta_{i0}) \rightarrow$$

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

Q2) Show $P(y=1|x_1=\alpha_1, x_2=\alpha_2) > P(y=1|x_1=\alpha_1)$

$P(y=1|x_1=\alpha_1) > P(y=0|x_1=\alpha_1)$ → using naive bayes assumption →

$P(y=1) \cdot P(x_1=\alpha_1|y=1) > P(y=0) \cdot P(x_1=\alpha_1|y=0)$ and because

$P(y=1) = P(y=0)$ → $P(y=1) \cdot P(x_1=\alpha_1|y=1) > P(y=0) \cdot P(x_1=\alpha_1|y=0)$ →

$P(x_1=\alpha_1|y=1) > P(x_1=\alpha_1|y=0)$ then $P(y=1|x_1=\alpha_1, x_2=\alpha_2) > P(y=1|x_1=\alpha_1)$ →

$P(y=1) \cdot P(x_1=\alpha_1|y=1) \cdot P(x_2=\alpha_2|y=1) > P(y=0) \cdot P(x_1=\alpha_1|y=1)$ →

$P(x_1=\alpha_1|y=1) \cdot P(x_2=\alpha_2|y=1) > P(x_1=\alpha_1|y=1)$ → because $P(x_1=\alpha_1|y=1) = P(x_2=\alpha_2|y=1)$ →

$2 \cdot P(x_1=\alpha_1|y=1) > P(x_1=\alpha_1|y=1)$ → and because

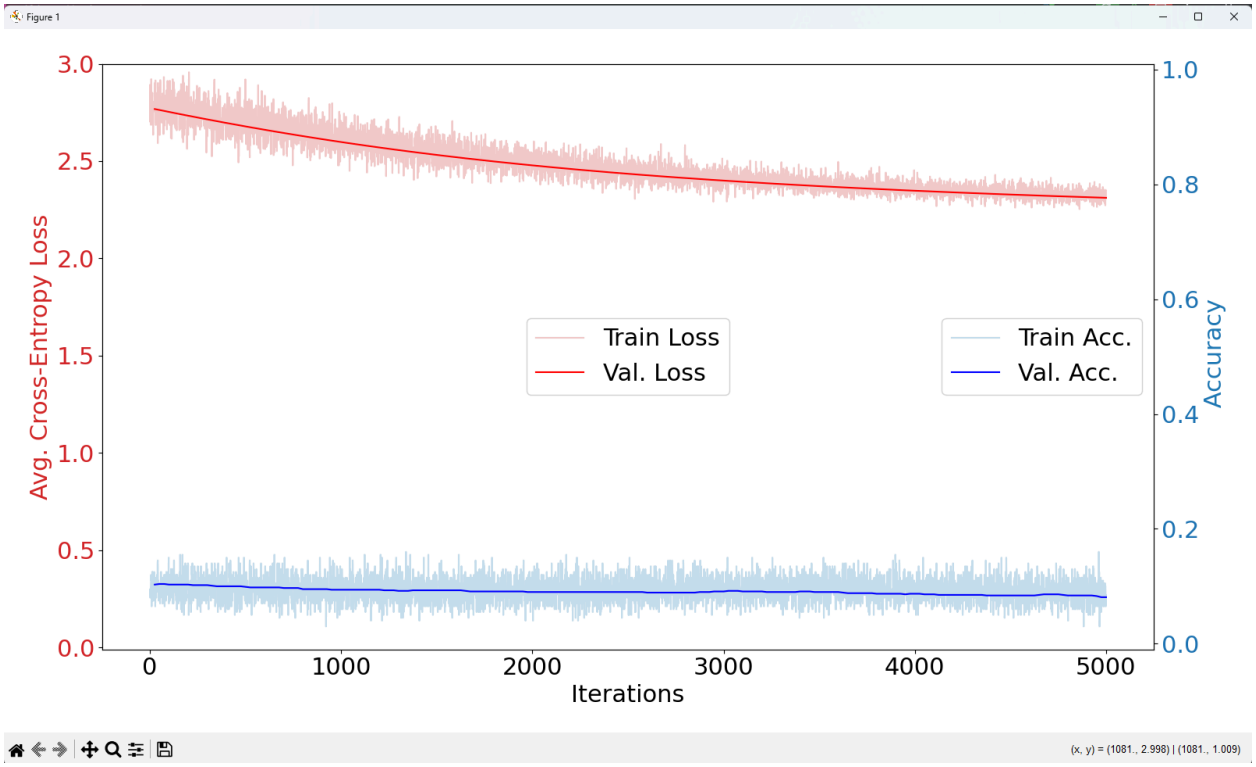
$P(x_1=\alpha_1|y=1) \geq 0$ $P(x_1=\alpha_1|y=1) > P(x_1=\alpha_1|y=0)$ thus

$P(x_1=\alpha_1|y=1) > 0$ and $2x > x$ where $x > 0$ thus

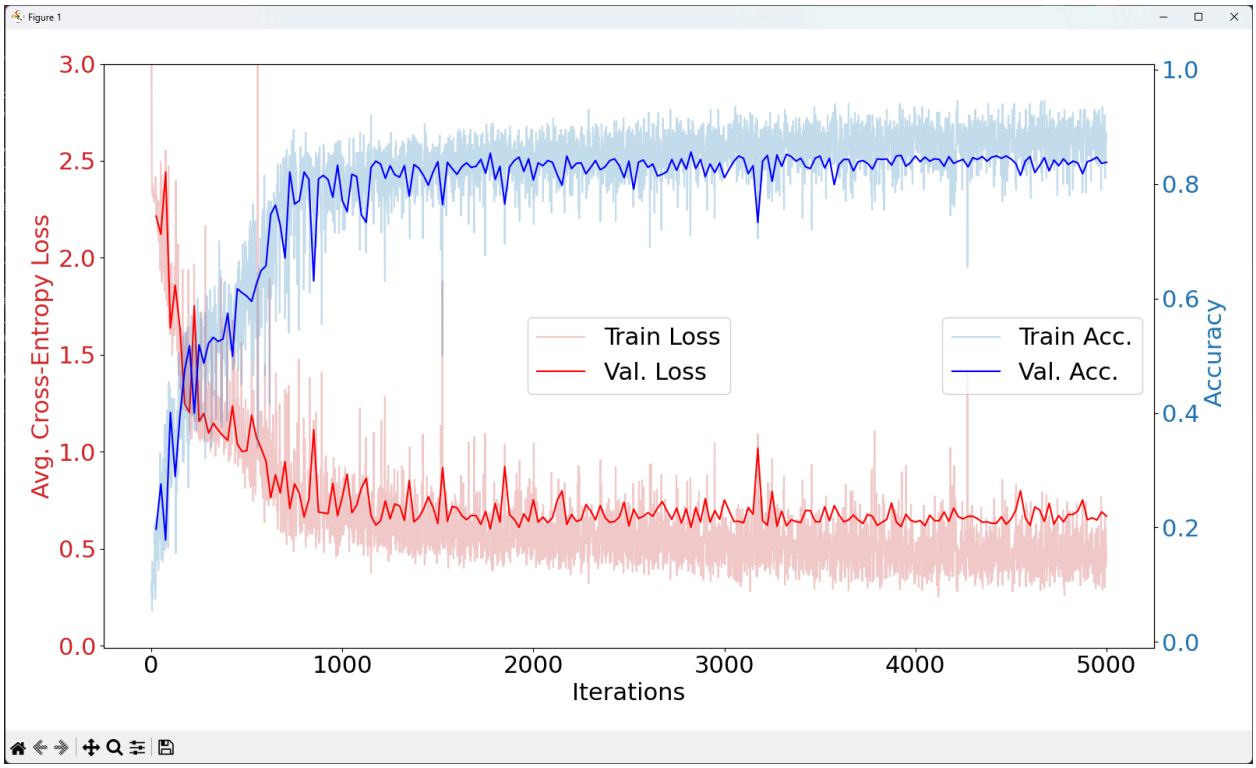
$2 \cdot P(x_1=\alpha_1|y=1) > P(x_1=\alpha_1|y=1) \Rightarrow P(y=1|x_1=\alpha_1, x_2=\alpha_2) > P(y=1|x_1=\alpha_1)$

Q4)

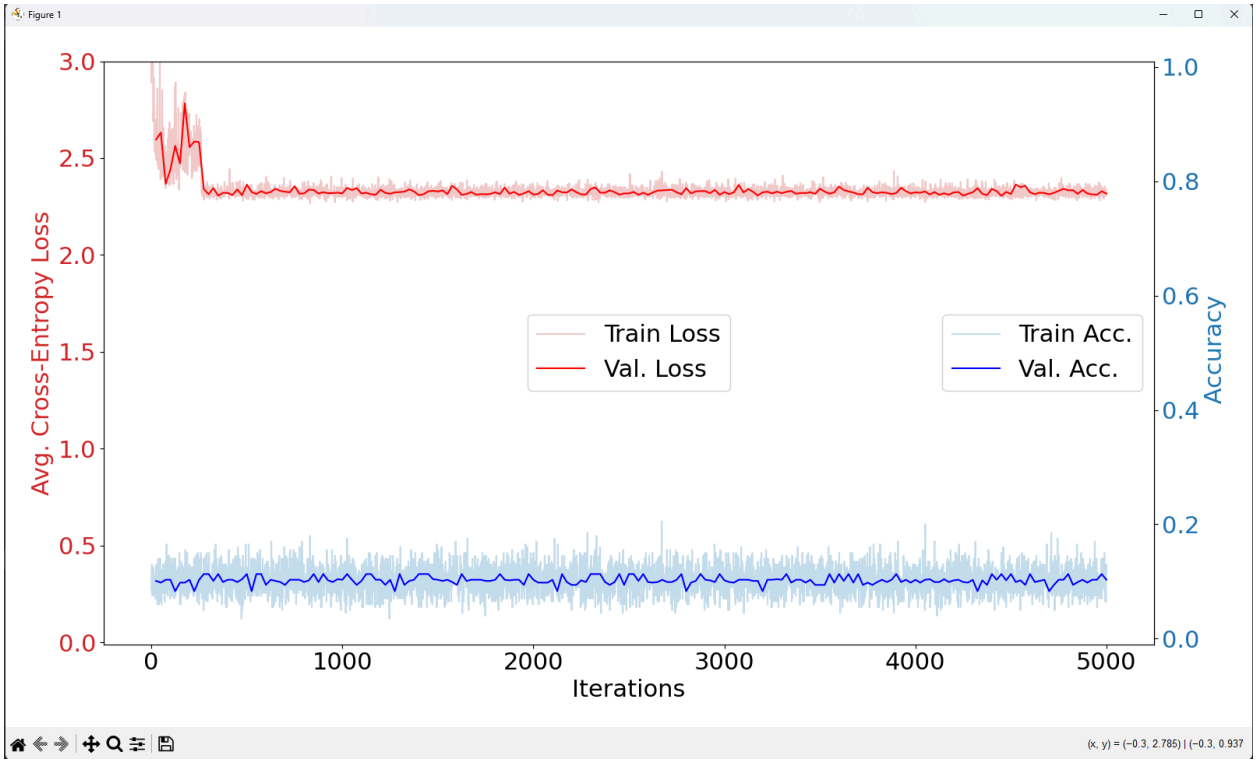
1. Step size of 0.0001 (leave default values for other hyperparameters)



2. Step size of 5 (leave default values for other hyperparameters)



3. Step size of 10 (leave default values for other hyperparameters)

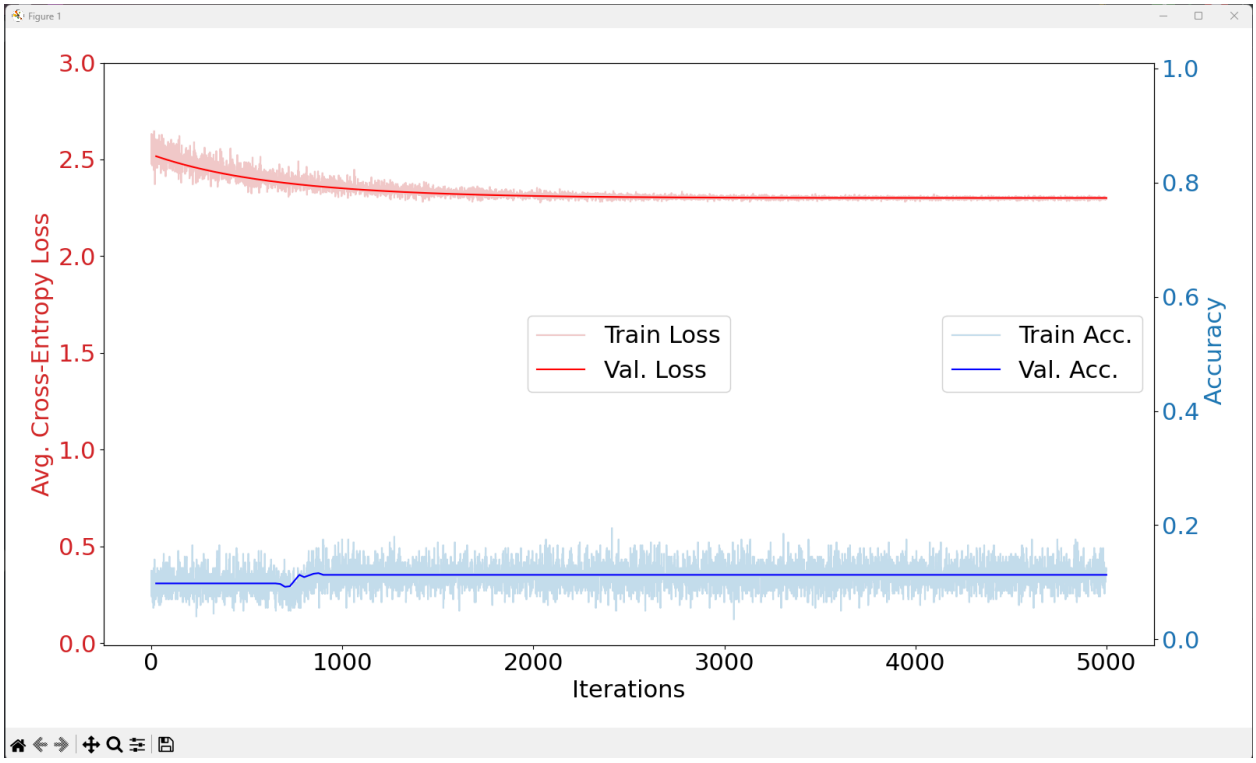


a) Compare and contrast the learning curves with your curve using the default parameters. What do you observe in terms of smoothness, shape, and what performance they reach?

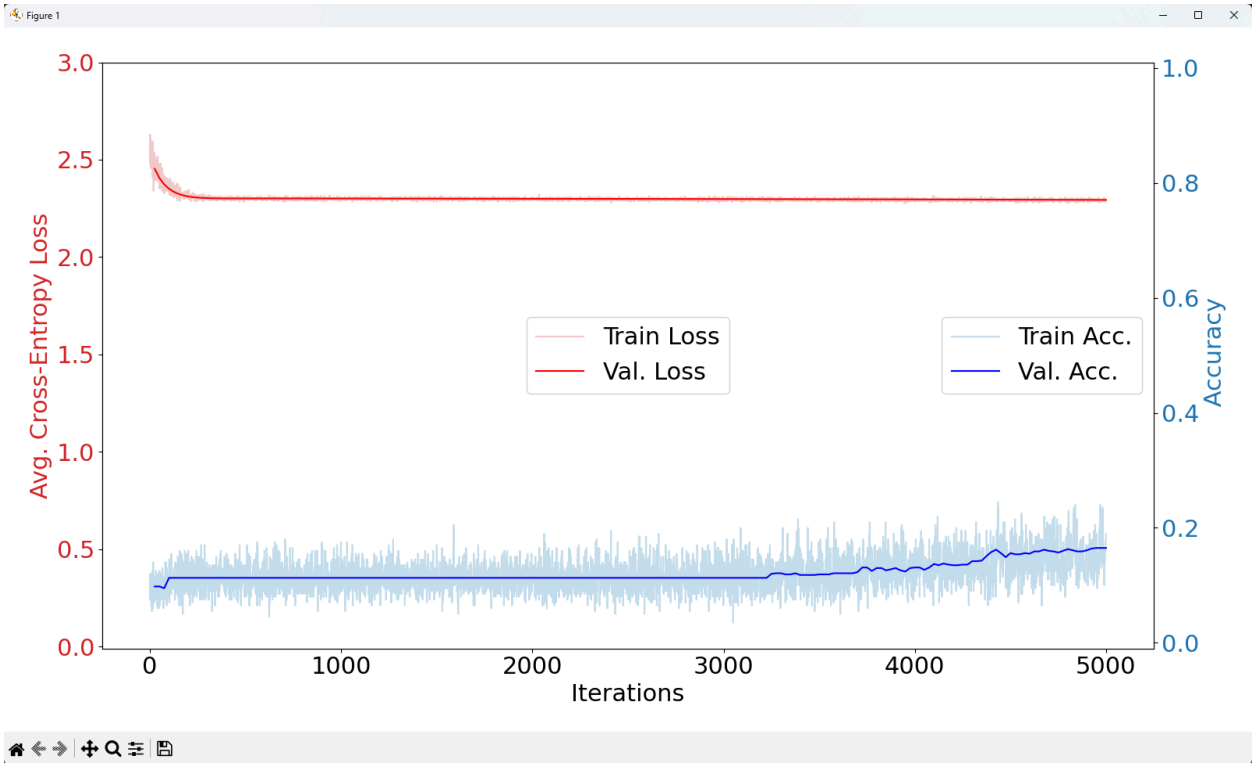
I notice that When the step size is too little or too much it does not produce results, this can be seen in the .00001 and the 10 where neither makes any notable progress. Comparatively though 5 worked decently well where it got to about 80% accuracy then it got sporadic and jumped between 82% and 70% roughly. The smoothness seemed to match as it decreased or it was smoother as the step size decreased. The shape was similar to the performance where .00001 and 10 were lines and 5 was similar to a log function.
b) For (a), what would you expect to happen if the max epochs were increased?
I would expect that .00001 would start to get better, and 5 and 10 to match how they performed in the original.

Q5)

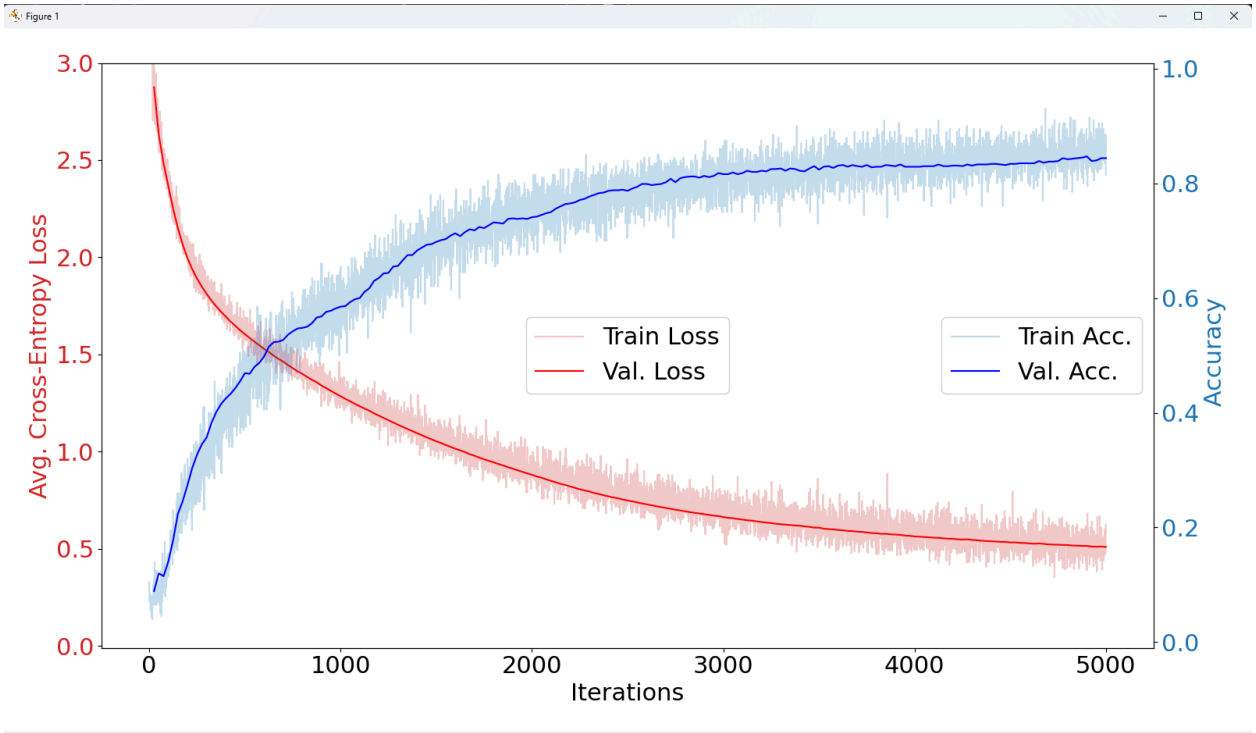
1. 5-layer with Sigmoid Activation (leave default values for other hyperparameters)



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?

Yes so weirdly both the sigmoid functions performed poorly having very little curve but remaining pretty smooth due to no/little change. They were both funny as the val vs train loss were about the same. The last one was the sigmoid this one performed well getting to the 84% and being somewhat smooth over the course of training. In this one there was a significant difference in val loss and range of training loss.

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

This might be because the sigmoid operates slower like in the demo we saw in class sigmoid is relatively slower and this could be because sigmoid caps the inputs between -1 and 1 where as relu doesnt.

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

Considering the derivative functions of both the relu and sigmoid, relu is 0 for anything negative and 1 else where as sigmoid is (sigmoid(x))(1-sigmoid(x) thus the sigmoid function is more complex and like stated before sigmoid caps the max output values at -1,1.

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?

1)Seed:50 ,step size:.001,batch size:200,max epochs:200,number of layers:5,width of layers:16
Loss: 0.5161, Train acc: 84.74%, va acc 86%
2)Seed:75 ,step size:.001,batch size:200,max epochs:200,number of layers:5,width of layers:16
Loss: 0.4939 Train Acc: 85.06% Val Acc: 84.1%
3)Seed:1,step size:.001,batch size:200,max epochs:200,number of layers:5,width of layers:16
Loss: 0.8045 Train Acc: 74.84% Val Acc: 74.0%
4)Seed:200,step size:.001,batch size:200,max epochs:200,number of layers:5,width of layers:16
Loss: 0.673 Train Acc: 79.34% Val Acc: 78.4%
5)Seed:150,step size:.001,batch size:200,max epochs:200,number of layers:5,width of layers:16
Loss: 0.5798 Train Acc: 83.9% Val Acc: 84.3%'

Randomness can have a big impact on this as we had a low of 74% val accuracy and a high of 86% meaning there was a range of 12% in our validation accuracy which is huge.

q7)
Ended up around .0001 step size, 7000 epochs, 20 layers, 28 width, I got here by just fiddling with numbers trying to get the validation as high as possible/reducing the difference between validation and training but only really got to 90ish.

1. Approximately how many hours did you spend on this assignment?
8-12?
2. Would you rate it as easy, moderate, or difficult?
Moderate, kaggle is hard, the initial assignment was easy once understood.
3. Did you work on it mostly alone or did you discuss the problems with others?
Alone
4. How deeply do you feel you understand the material it covers (0%–100%)?
80-100%
5. Any other comments?
none