Esmat Sahak
ECE324

Assignment 3: Multi-Layer Perceptron for Binary Classification

3.2: Understanding the Dataset

How many high income earners are there in the dataset? How many low income earners?
High income earners: 11687
Low income earners: 37155

Is the dataset balanced (i.e. does it have roughly the same number of each of the classes/labels - in this case high income vs. low income)? What are some possible problems with training on an unbalanced dataset?
The training set is unbalanced. This bias for the overrepresented class can lead to poor predictions, particularly for the classes with lower sample sizes. This then reduces the per-class accuracy.

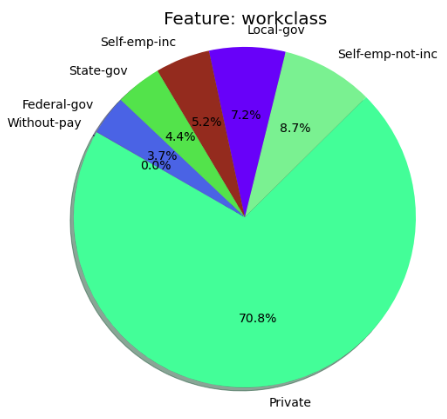
3.3: Cleaning

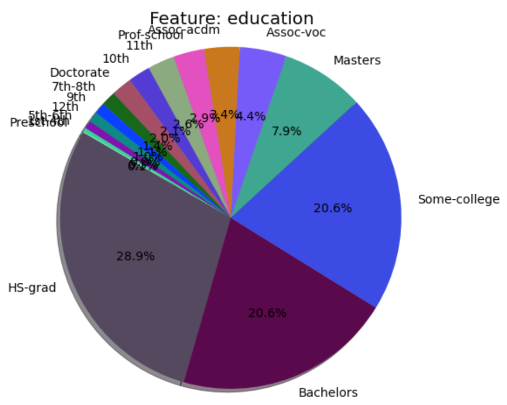
How many samples (rows) were removed by the above cleaning process? How many are left?
There are 45222 sample remaining. Hence, 48842 - 45222 = 3620 samples were removed.

Do you think this is a reasonable number of samples to throw out?
Of the original dataset, 7.41% of the samples were thrown out. This isn't a significant amount (< 10%) and there is still a large number of data samples to work with, so this is a reasonable number of samples to filter out.

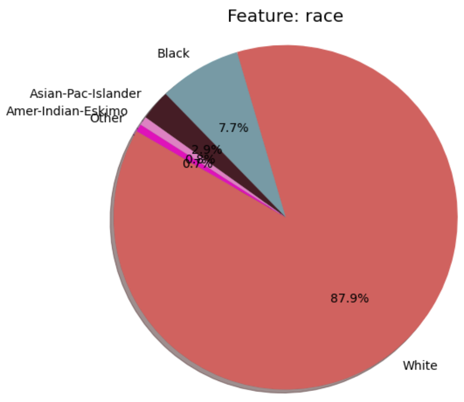
3.5: Visualization and Understanding

What is the minimum age of individuals and the minimum number of hours worked per week for the dataset?
Minimum age: 17
Minimum hours worked per week: 1





Esmat Sahak
ECE324

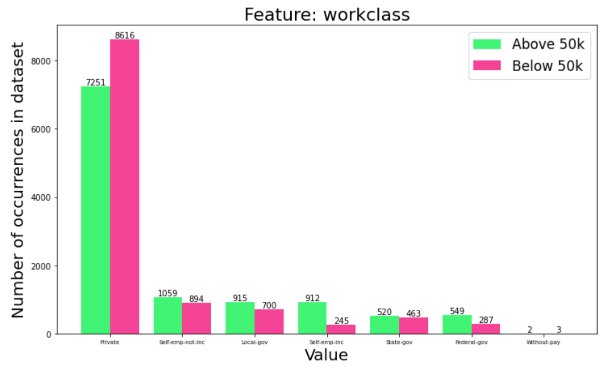


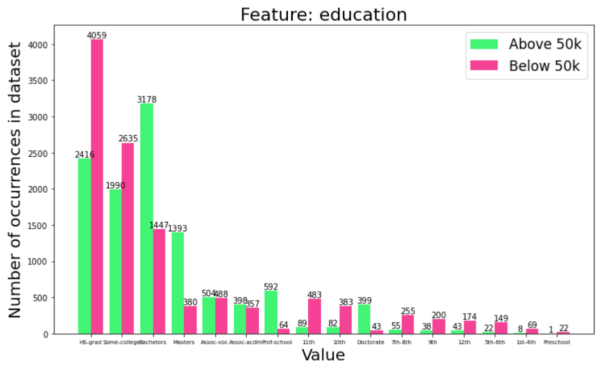
Are certain groups over or under-represented? For example, do you expect the results trained on this dataset to generalize well to different races?

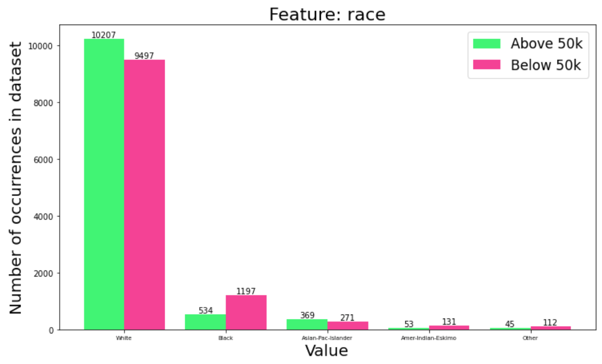
Yes, there are certain groups over/under represented; for example, 87.9% of the sample is white, while 7.7% of the sample is black. Thus, the expectation is results will not generalize well to different races, instead prediction accuracy will much lower for a non-white person.

What other biases in the dataset can you find?

Other biases in the dataset include the unequal distribution of gender (73.5% male, 26.5% female), underrepresentation of immigrants (92% listed US as native country), and loose grouping of occupations.







Esmat Sahak
ECE324

List the top three features, that you identify using this method, that are very useful for distinguishing between high and low salary earners?

1. Marital status and relationship (they are similar categories).
2. Education.
3. Occupation.

Suppose you're told that someone has high school-level of education. How likely would they be to earn above 50K? Given their education is "Bachelors" and nothing else about a person what is the best assumption as to whether they earn above or below 50K?

If someone has high school-level education, they have a probability of $\frac{2416}{2416+4059} \times 100\% \simeq 37.32\%$ of earning above \$50,000. If their education is bachelors, the better assumption is that they earn above \$50,000 ($\frac{3178}{3178+1447} \times 100\% \simeq 68.71\%$).

<u>3.6: Pre-processing</u>

What are some disadvantages of using an integer representation for categorical data?

The problem with integer representation is that it suggests an implicit order. The algorithm as a result may interpret some values to be *better* than others, introducing a source of bias.

What are some disadvantages of using un-normalized continuous data?

Data with larger absolute values will have a greater influence on the output (recall $z = \sum_{i=0}^{n-1} w_i I_i + b$). To mitigate this, the weights must be adjusted accordingly, but this may require more training data and takes more training effort. **Note: bonus at end of document.**

<u>4.2: DataLoader</u>

Why is it important to shuffle the data during training? What problem might occur during training if the dataset was collected in a particular order (e.g. ordered by income) and we didn't shuffle the data?

It is important to shuffle data as it mitigates the risk of batches that are not representative of the dataset - this will cause poor gradient estimates. If data is not shuffled after each epoch, bias is introduced as the gradient operation of the current batch is dependent on the result of the previous batch. Furthermore, the element of randomness also helps converge to a minimum (and avoid local minima) as the gradients are more likely to point in the correct direction.

Give a justification for your choice of the size of the first layer. What should the size of the second (output) layer be?

The size of the output layer should be one. I chose the size of the first layer to be 50 neurons as it is near the average of the number of inputs and outputs ($\frac{103+1}{2} = 52$). Generally, a number between the number of inputs and size of the output layer should be selected for the size of hidden layers. Typically, powers of 2 are chosen for binary classification, so 64 would be a good choice as well.

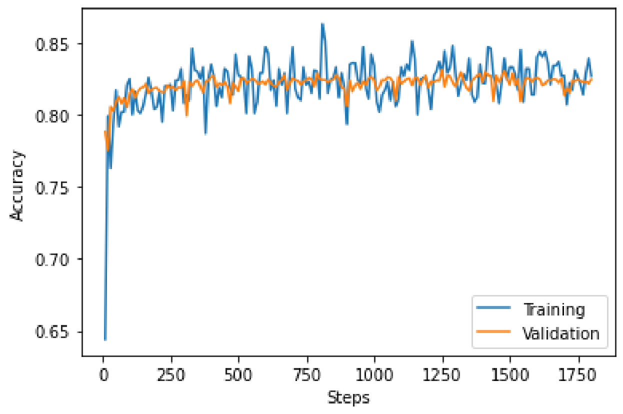
Esmat Sahak
ECE324

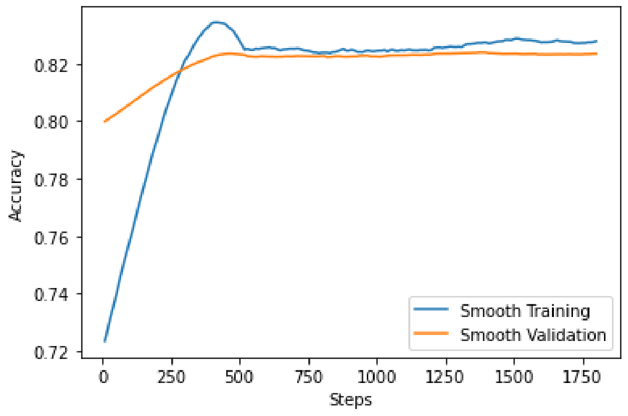
Why do we think of the output of the neural network as a probability between 0 and 1? What do the specific values of 0 and 1 mean if they are output from the network?

We think of the output as a probability as the sigmoid function maps predictions to probability estimates (note that sigmoid is not a valid probability distribution function but bounds output between 0 to 1). It is more appropriate to predict the likelihood of earning over $50K as predictions are not ensured to be correct. An output of 0 indicates the predictor is certain the person does not earn over $50K, while an output of 1 indicates the opposite.

<u>4.6: Validation</u>

Epochs = 10, Batch size = 100, MLP hidden size = 50, Learning rate = 1, Seed = 10





What validation accuracy does your model achieve?

The model achieves a validation accuracy of 0.8245.

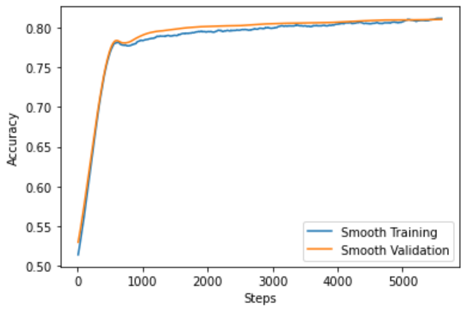
<u>5.1: Learning Rate</u>

Epochs = 20, Batch size = 64, MLP hidden size = 64, Seed = 10	
Learning Rate	**Accuracy of Validation Data**
0.001	0.7857
0.01	0.8109
0.1	0.8267
1	0.8220
10	0.8160
100	0.5009
1000	0.5009

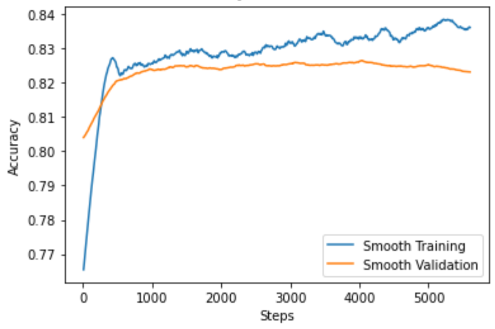
Esmat Sahak
ECE324

Epochs = 20, Batch size = 64, MLP hidden size = 64, Seed = 10

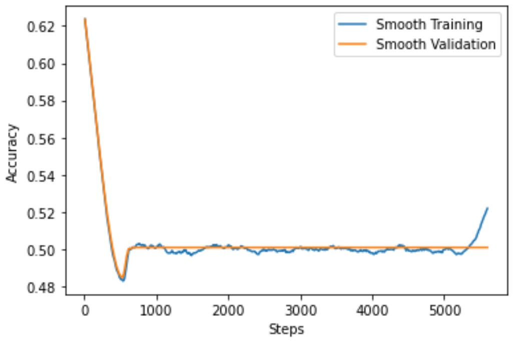
Learning rate = 0.01



Learning Rate = 1



Learning rate = 100



Which learning rate works the best?

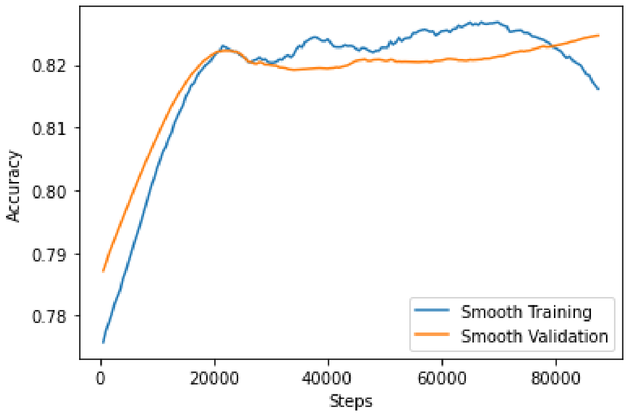
A learning rate of 0.1 generated the highest training and validation accuracy.

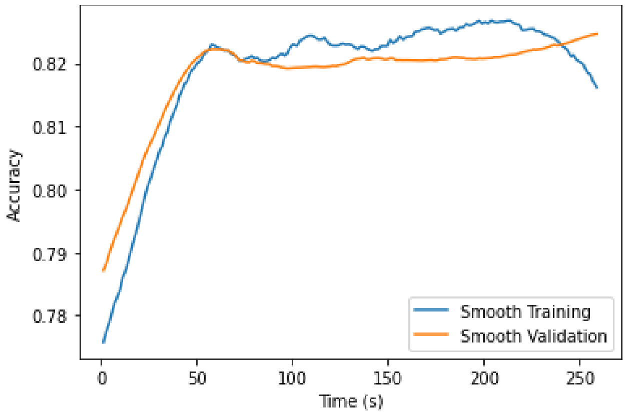
What happens if the learning rate is too low? Too high?

If the learning rate is too low, the training will take longer as more epochs are needed to converge to some minimum value. It is also more likely to get stuck in a local minima with a lower learning rate. If the learning rate is too high, the training error increases as parameters become too large.

5.3: Batch Size

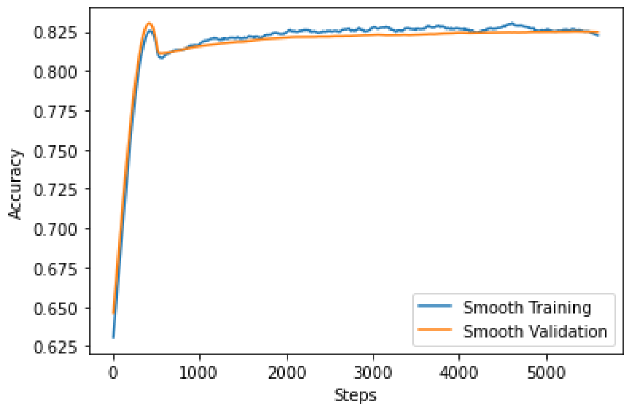
Batch size = 1, Epochs = 5, N = 500, MLP hidden size = 64, Seed = 10, Learning rate = 0.1

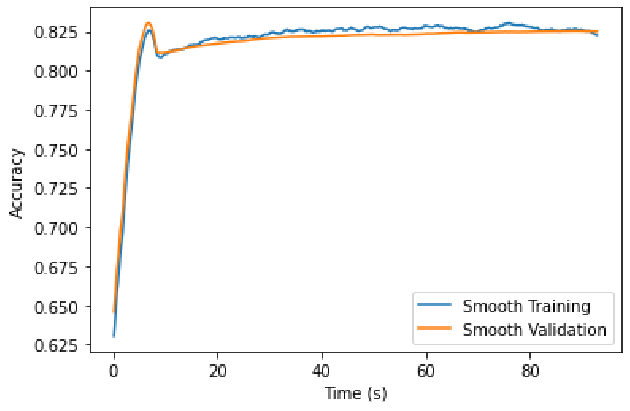




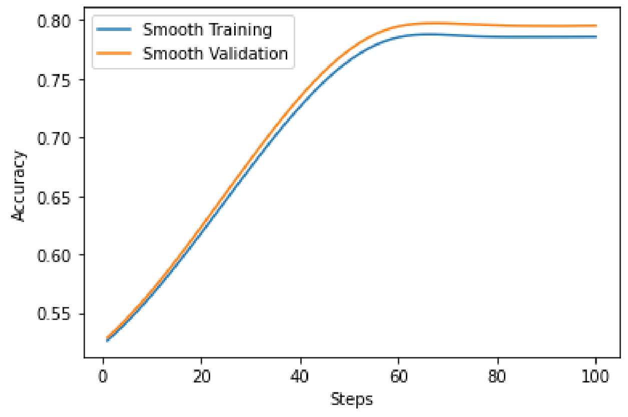
Esmat Sahak
ECE324

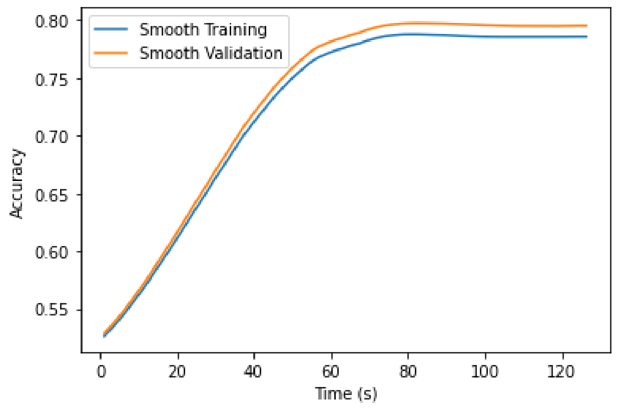
Batch size = 64, Epochs = 20, N = 10, MLP hidden size = 64, Seed = 10, Learning rate = 0.1





Batch size = 17932, Epochs = 100, N = 1, MLP hidden size = 64, Seed = 10, Learning rate = 0.1





Which batch size gives the highest validation accuracy?
A batch size of 64 gives the highest validation accuracy.

Which batch size is fastest in reaching a high validation accuracy in terms of the number of steps? Which batch size is fastest in reaching its maximum validation accuracy in terms of time?
Batch size of 17932 is fastest in terms of number of steps.
Batch size of 64 is fastest in terms of time.

What happens if the batch size is too low? Too high?
It takes longer to compute the gradients for a larger batch size which increases total training time (if number of epochs is constant). Smaller batch sizes produce noisier plots, which implies it may not converge as it isn't consistently moving in the direction which minimizes loss. The noise is beneficial in some context as it is less likely to get stuck in some local minima.

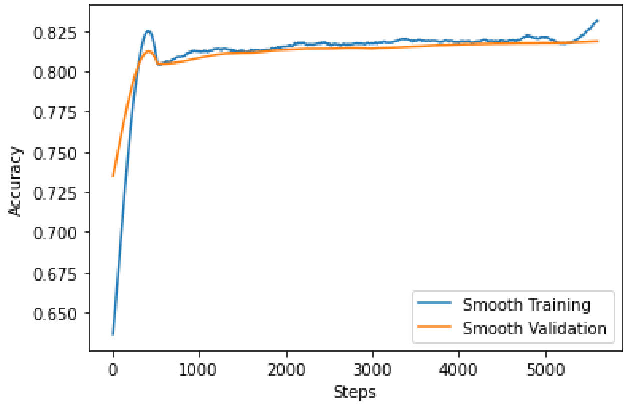
State the advantages and disadvantages of using a small batch size. Do the same for large batch size. Make a general statement about the value of batch size to use (relative to 1 and the size of the dataset).

Esmat Sahak
ECE324

Smaller batches reduce training time (assuming constant epochs) but generate less accurate results due to the noise. This noise is somewhat beneficial as it helps escape local minima. Larger batches generate more accurate results but take much more time to train. To maximize efficiency, batch size should be set to some value between 1 and the size of the dataset such that its low enough for fast training but high enough for accurate results.

5.4: Under-fitting

Batch size = 64, Epochs = 20, N = 10, MLP hidden size = 64, MLP layers = 1, Seed = 10, Learning rate = 0.1

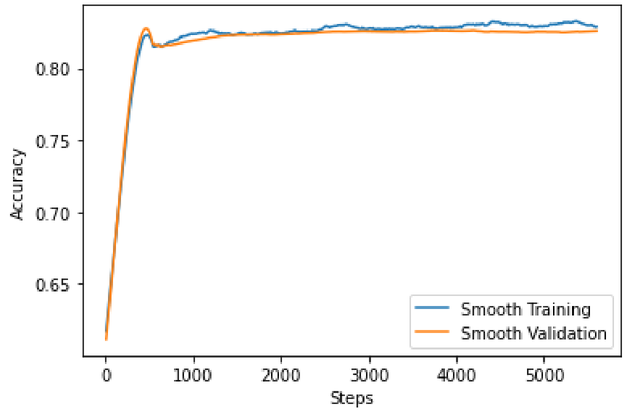


What validation accuracy does the small model achieve? How does this compare to the best model you've trained so far? Is this model underfitting?

The validation accuracy is 0.8189 which is relatively close to the best model trained (≅ 0.83). This model is not underfit.

5.5: Over-fitting

Batch size = 64, Epochs = 20, N = 10, MLP hidden size = 64, MLP layers = 4, Seed = 10, Learning rate = 0.1



What validation accuracy does the large model achieve? How does this compare to the best model you've trained so far? Is this model over-fitting?

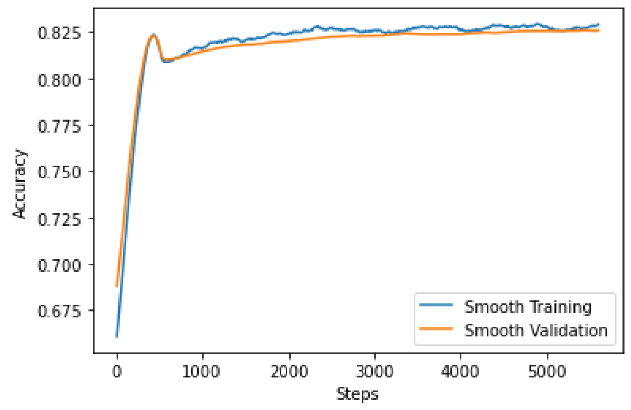
The validation accuracy is 0.8267 which is relatively close to the best model trained (≅ 0.83). This model is not overfit.

Esmat Sahak
ECE324

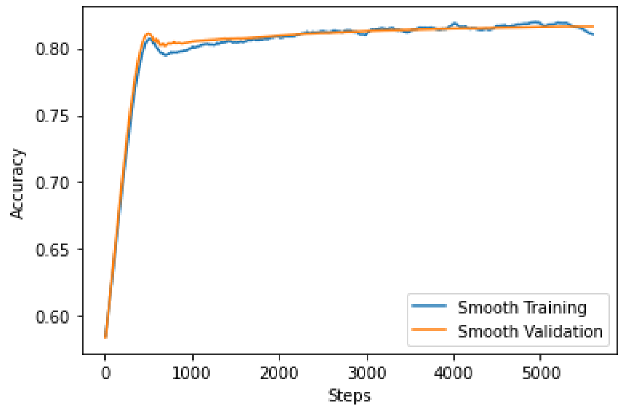
5.6: Activation Function

Batch size = 64, Epochs = 20, N = 10, MLP hidden size = 64, MLP layers = 2, Seed = 10, Learning rate = 0.1

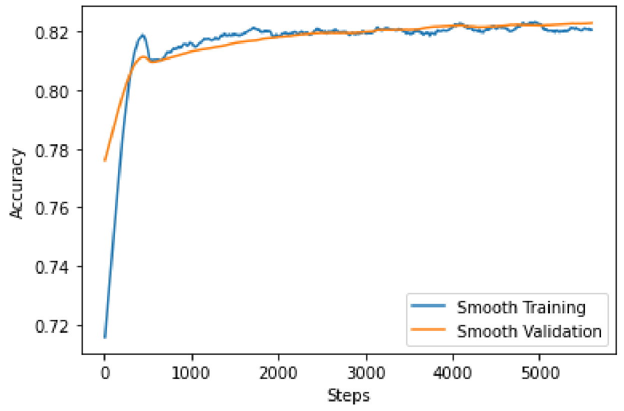
ReLU



Sigmoid



tanh



Do you notice any qualitative or quantitative differences?

Sigmoid takes longer to converge and generates a slightly lower validation accuracy. The training accuracy for ReLU is slightly higher than the validation accuracy, this is indicative of good fitting. ReLU also generates slightly higher validation accuracies.

Is there a difference between the activation functions in terms of how long they take to run?

Batch size = 64, Epochs = 20, N = 10, MLP hidden size = 64, MLP layers = 2, Seed = 10, Learning rate = 0.1	
Activation Function	Time (s)
ReLU	88.905
Sigmoid	108.278
tanh	96.491

Sigmoid is the slowest, so it is more computationally expensive. ReLU is the fastest, and is therefore the least expensive.

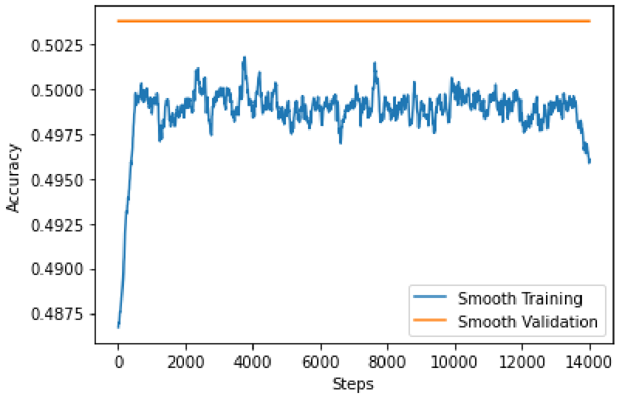
5.7: Hyperparameter Search

Best configuration: learning rate = 0.1, batch size = 64, epochs = 20, N = 10, MLP hidden size = 64, MLP layers = 2, seed = 10, activation function = ReLU

Esmat Sahak
ECE324

<u>Bonus</u>

Batch size = 64, Epochs = 50, N = 10, MLP hidden size = 14, MLP layers = 2, Seed = 10, Learning rate = 1e-6



It is evident that un-normalized data generates very poor results. The accuracies hover around 50% implying that this is no better than a random predictor. The validation accuracy doesn't even change (behaviour similar to the dying ReLU problem) despite testing for different variations of hyperparameters.

<u>Feedback</u>

How much time did you spend on assignment 3?
10-12 hours. There was no added value in increasing the length of the lab, our schedules are very busy as it is. As a result, I couldn't enjoy the nice weather as Professor Rose recommended.

What did you find challenging?
The lack of appropriate supplementary resources where self-learning was required (e.g. dataloader links were useless). I'm not sure why these topics couldn't be covered in tutorial (maybe then they'd be more useful).
Running the code over and over tested my sanity.

What did you enjoy?
Nothing. I don't gain pleasure from completing assignments.

What did you find confusing?
The clarity of the document was poor in some instances (e.g. it mentions to plot training accuracy per step then mentions to plot it as an average).
The reason for looping when computing accuracy when there exist **much faster** PyTorch methods.

What was helpful?
Yviel Hernandez Castillejos' patience with my questions.