ECE324: Assignment 3

Aman Bhargava — 1005189733

3: Data Pre-Processing and Visualization

3.2: Understanding the Dataset

1. There are 11687 high income earners and 37155 low-income earners.
2. The dataset is not balanced. Some possible outcomes of training on an unbalanced dataset is that the model learns the *probability* of a given class regardless of the input features, resulting with worse model performance on new data despite (potentially high) training accuracy.

3.3: Cleaning

1. Initially there were 48842 rows. Now there are 45222. Therefore, 3620 rows were removed.
2. This is a reasonable number of samples to throw out as it is a relatively small portion of the data.

3.4: Balancing the Dataset

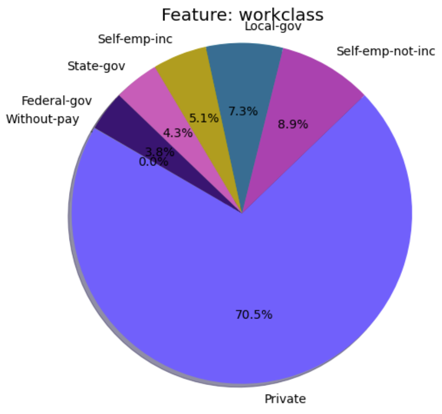
No questions were asked in this section.

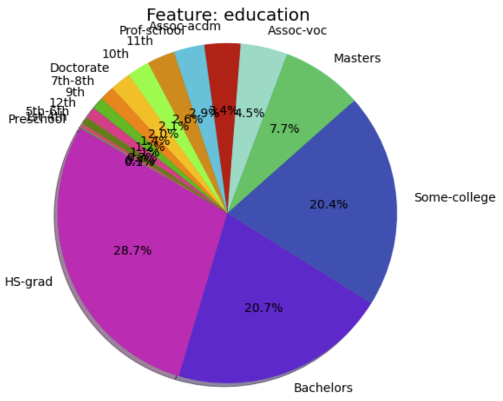
3.5: Visualization and Understanding

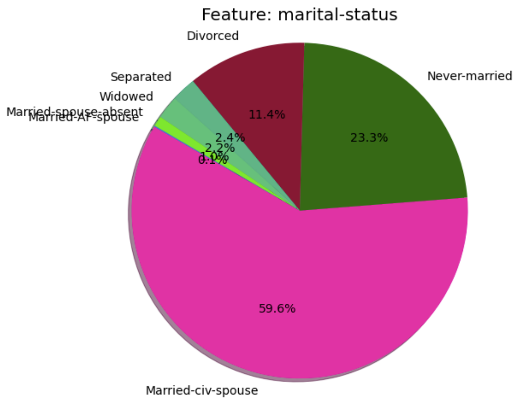
1. The minimum age of individuals was 17. The minimum number of hours per week worked was 1.
2. Privately employed married individuals are represented. Widowed individuals are also under represented. I would not expect this to generalize well to individuals living outside of the united states as over 90% of the samples are from the united states.

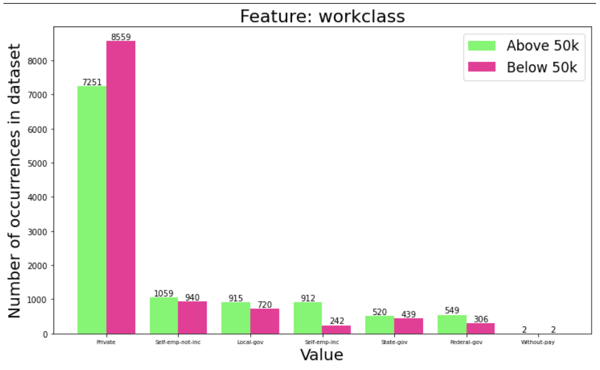
3. The data is anchored to the north american education systems (1-12th grades, college, high school). The educational distribution does not match the rest of the world.
4. High school grads are far more likely to earn below 50k. Those who have never married are more likely to make less than 50k. Those with only a preschool level of education are more likely to make less than 50k.
5. A high school level of education pre-disposes one to have a 38% chance of making more than 50k. A Bachelor's level implies a 68% chance of making more than 50k. It would therefore be best to assume that they do make more than 50k per year.

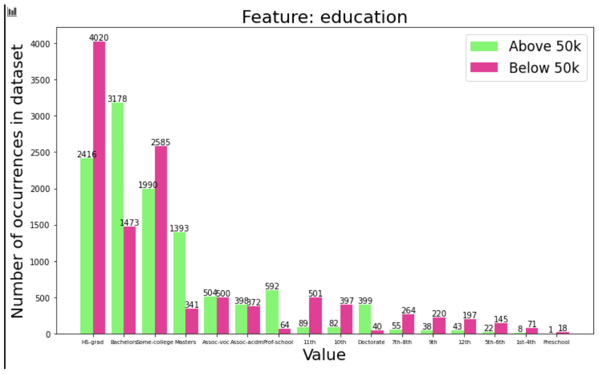
Charts

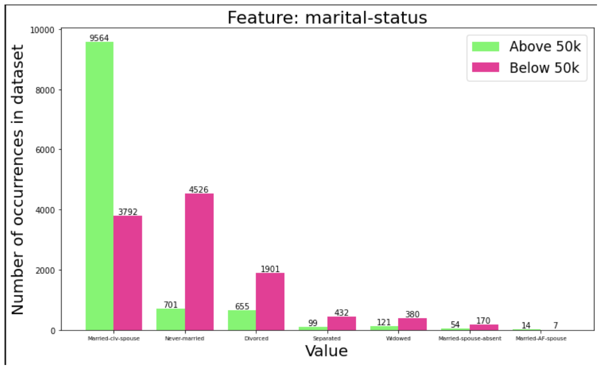












3.6: Pre-processing

1. Integer representation implies cardinality and order to categorical data which is arbitrary and synthetic. The optimal decision boundary is made much more complex by this encoding.
2. Un-normalized continuous data makes gradients have different orders of magnitude. A single learning rate, therefore, is unable to take optimally sized steps for each parameter in the model.

3.7: Train-Validation Split

No questions were asked for this section.

4: Model Training

4.1: Dataset

4.2: DataLoader

1. It is important to shuffle the data during training because a local optima that suits a particular category/type of data that was collected first may be found, impacting the remainder of the training.

4.3: Model

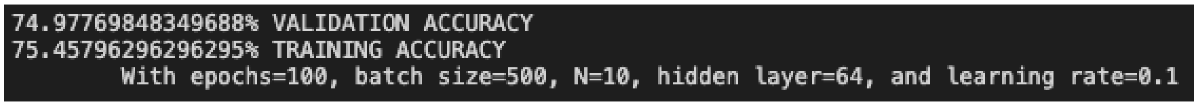
1. The first layer must be the same size as the dimensionality of the input vectors (103). The middle layer was chosen to be of size 64 as it is (loosely) a midpoint between the input and output sizes. The output layer must output a single number, so it has size 1.
2. We think of the output as a probability between 0 and 1 because it enables the 'correctness' of the network to be a smooth function with respect to its parameters. This is the same concept of 'brittleness' as discussed in class. The model can adjust over time and 'learn' what decision it should be less 'certain' about one way or another and slowly adjust itself over time to create an optimal decision boundary. 0 means that the network is 'certain' that the input individual makes less than 50k per year. 1 means that the network is 'certain' that the input individual makes more than 50k per year.

4.4: Loss Function and Optimizer

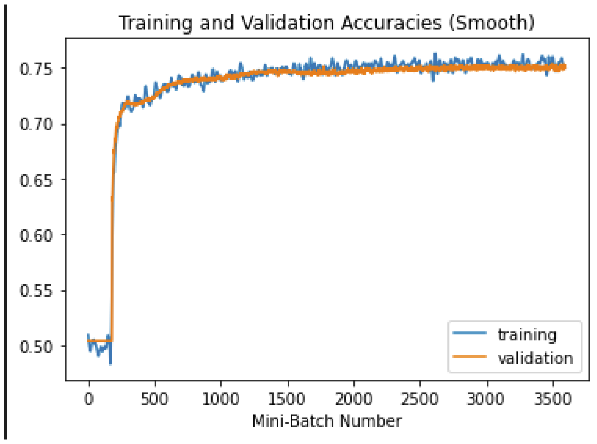
4.5: Training Loop

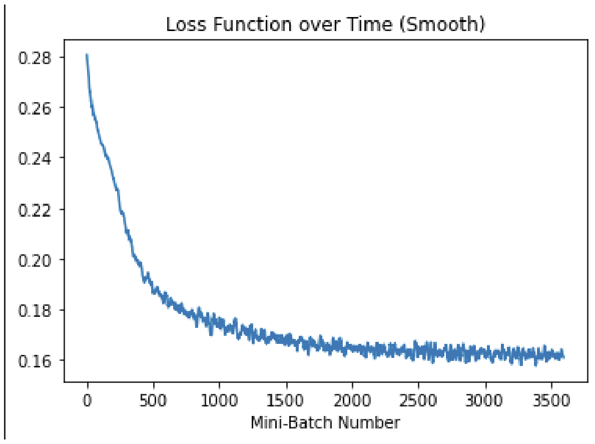
4.6: Validation

1. See screenshots below



```
74.97769848349688% VALIDATION ACCURACY
75.45796296296295% TRAINING ACCURACY
        With epochs=100, batch size=500, N=10, hidden layer=64, and learning rate=0.1
```



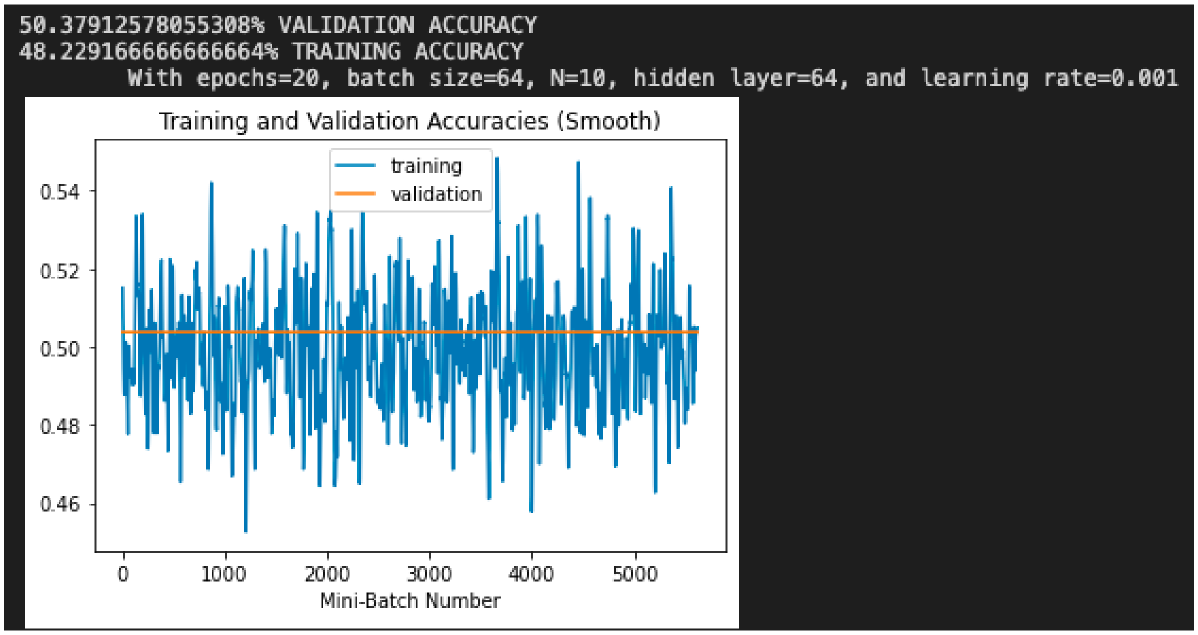


2. The above plots are smoothed using the savgol_filter.

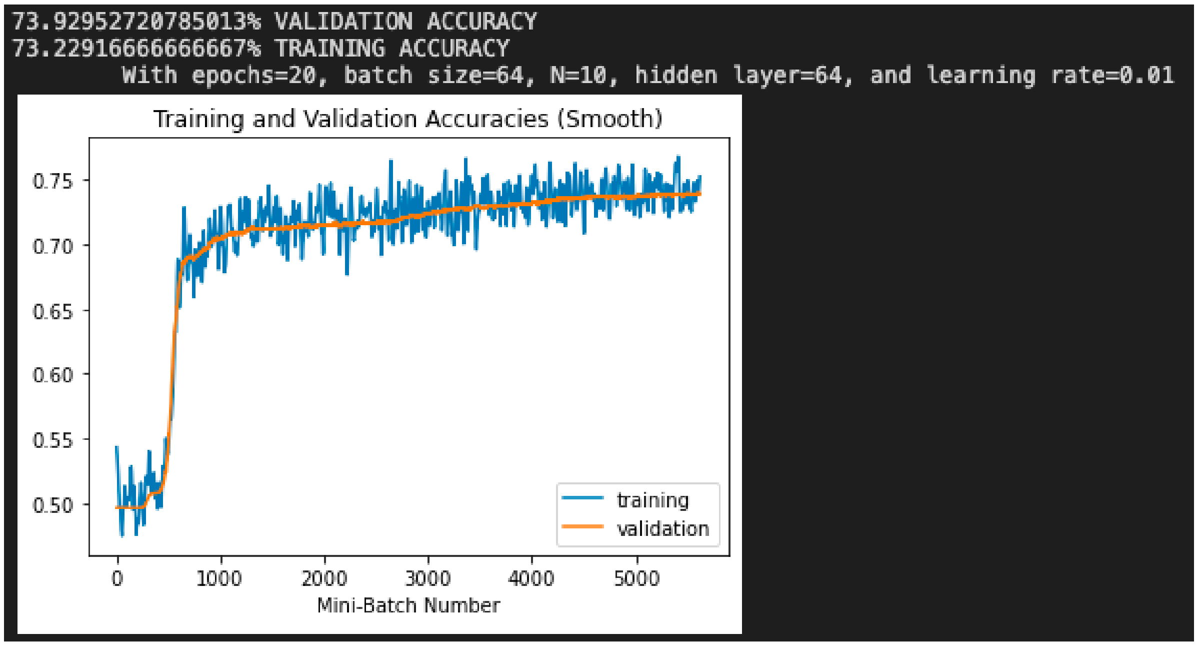
5: Hyperparameters

5.1: Learning Rate

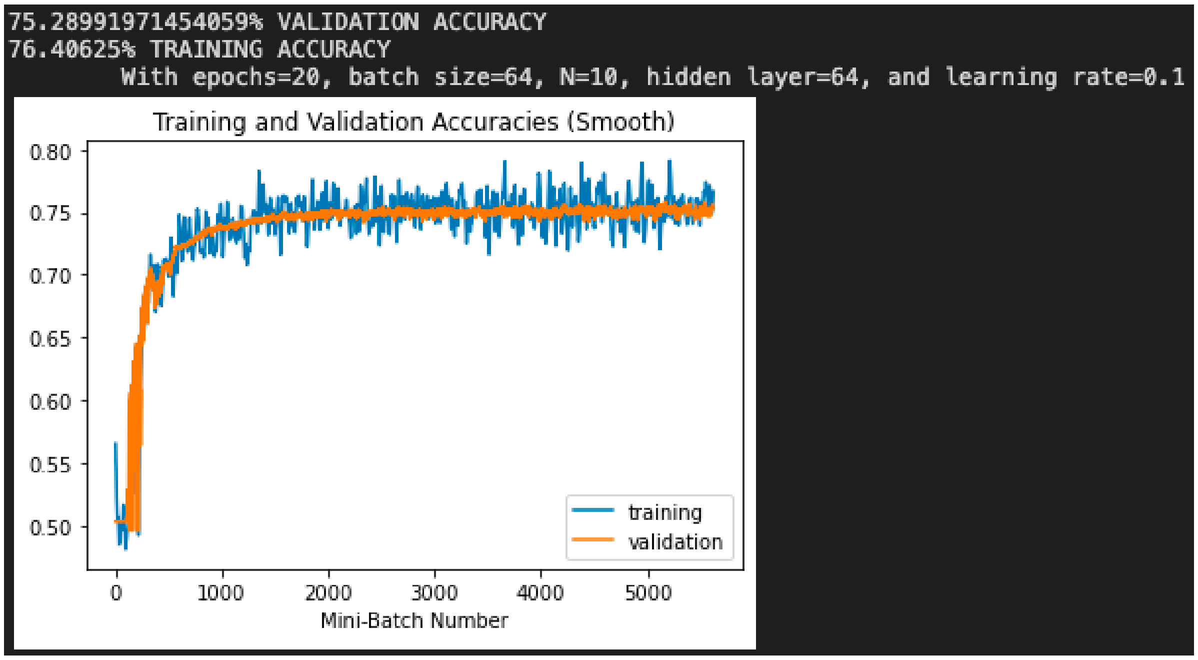
1. Highest validation accuracy for each learning rate in a table:
    1. LR = 0.001 → 50.38%
    2. LR = 0.01 → 73.92%
    3. LR = 0.1 → 75.28%
    4. LR = 1 → 74.50%
    5. LR = 10 → 71.43%
    6. LR = 100 → 49.62%
2. See plots below



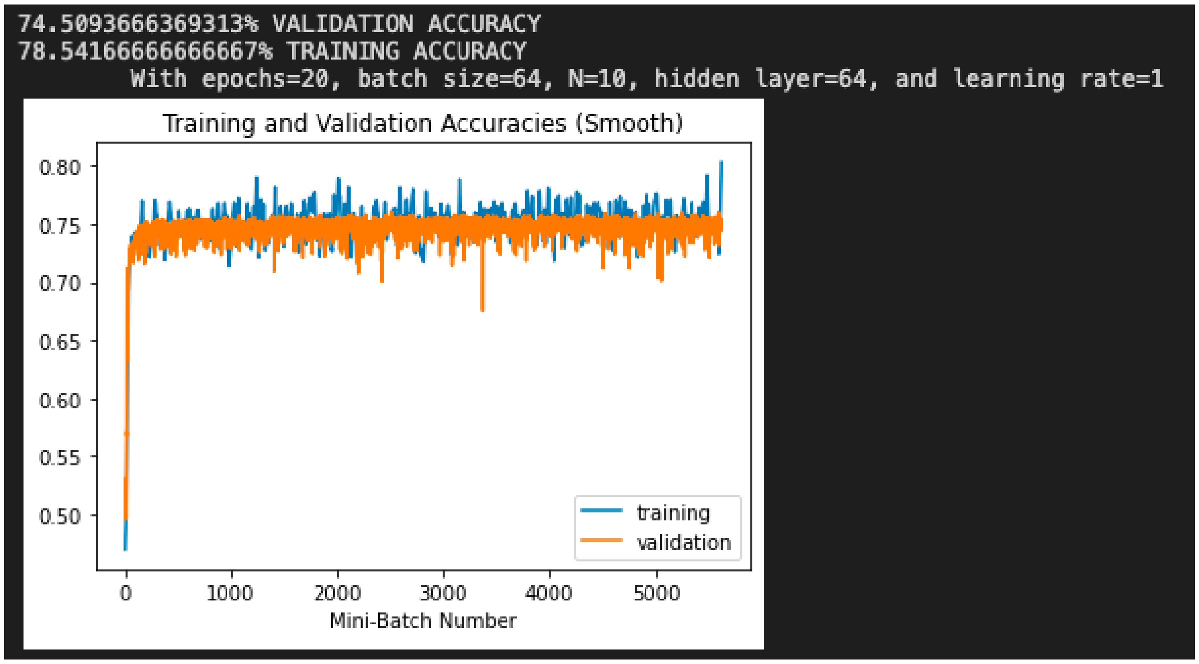
```
50.37912578055308% VALIDATION ACCURACY
48.229166666666664% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=0.001
```



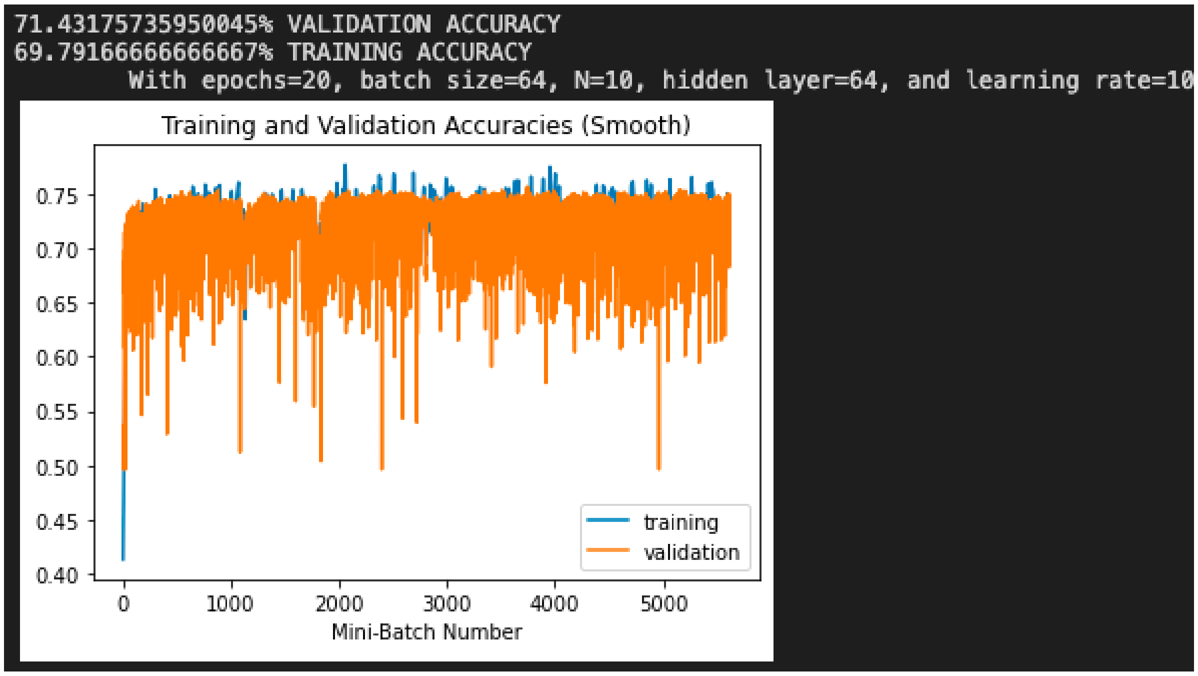
```
73.92952720785013% VALIDATION ACCURACY
73.22916666666667% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=0.01
```



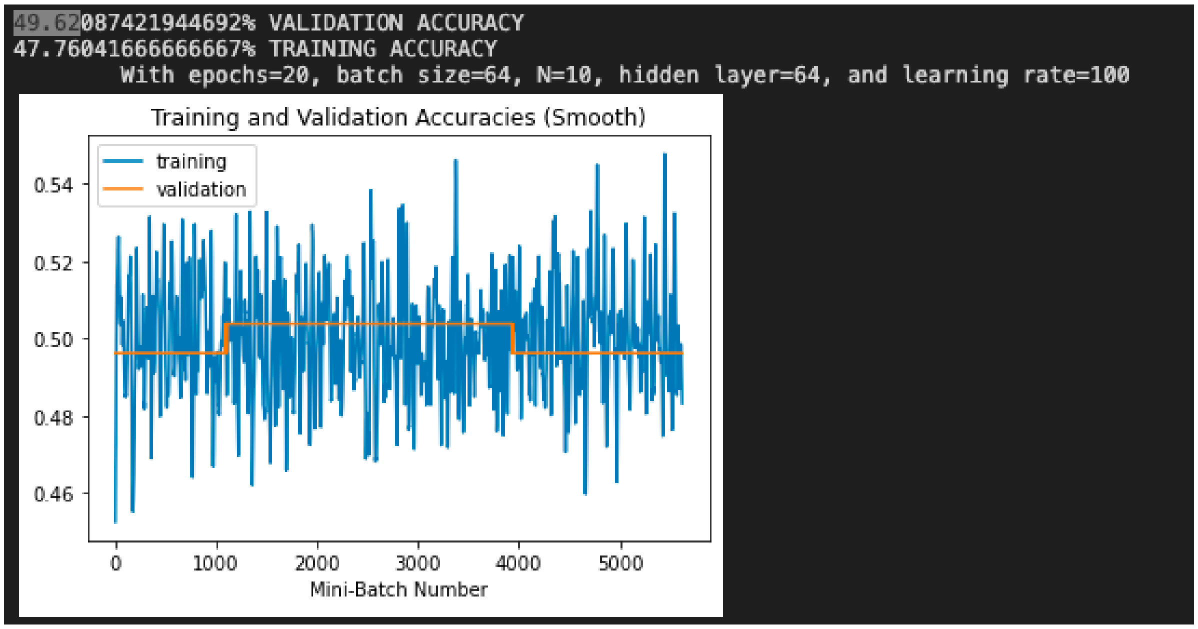
```
75.28991971454059% VALIDATION ACCURACY
76.40625% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=0.1
```



```
74.5093666369313% VALIDATION ACCURACY
78.54166666666667% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=1
```



```
71.43175735950045% VALIDATION ACCURACY
69.79166666666667% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=10
```



```
49.62087421944692% VALIDATION ACCURACY
47.76041666666667% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=100
```

Questons

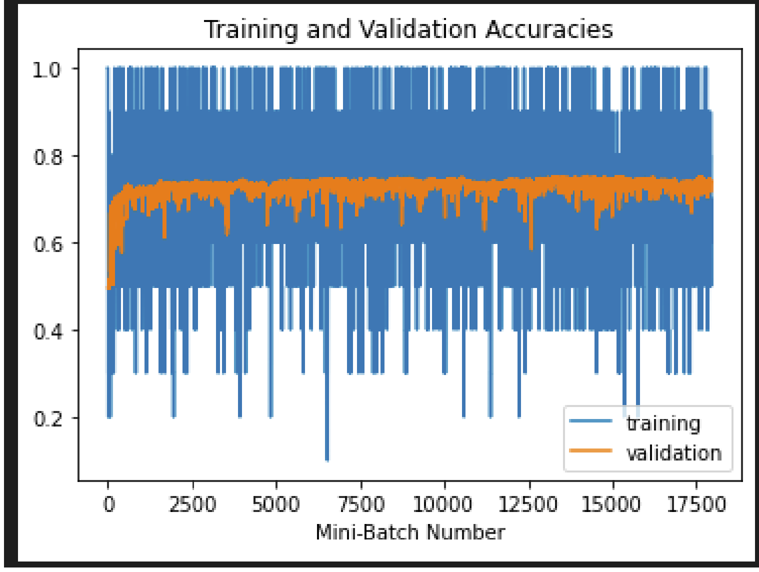
1. The learning rate of 0.1 works the best.
2. Too low of a learning rate means that the parameters don't change enough throughout the training process and the results stay roughly the same throughout from random instantiation. Too high of a learning rate means that the parameters change too much throughout the training process to the point that they over-shoot the local optima. Model performance suffers in both cases.

5.2: Number of Epochs

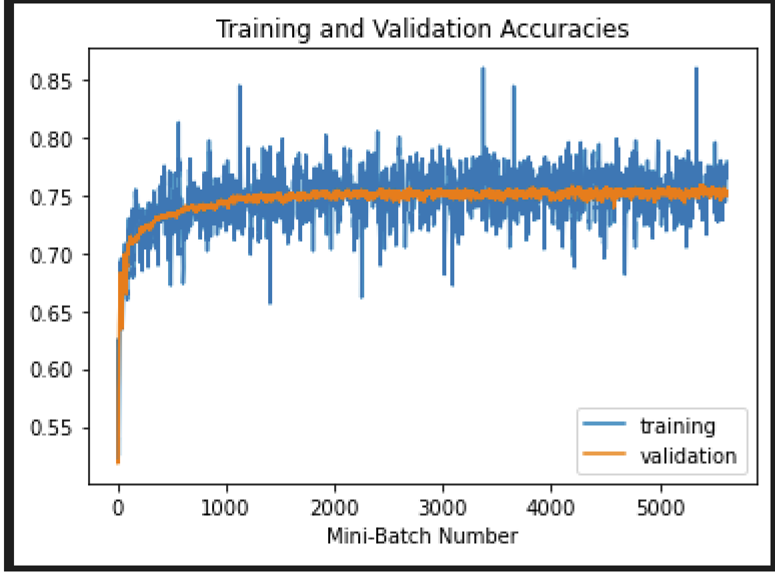
5.3: Batch Size

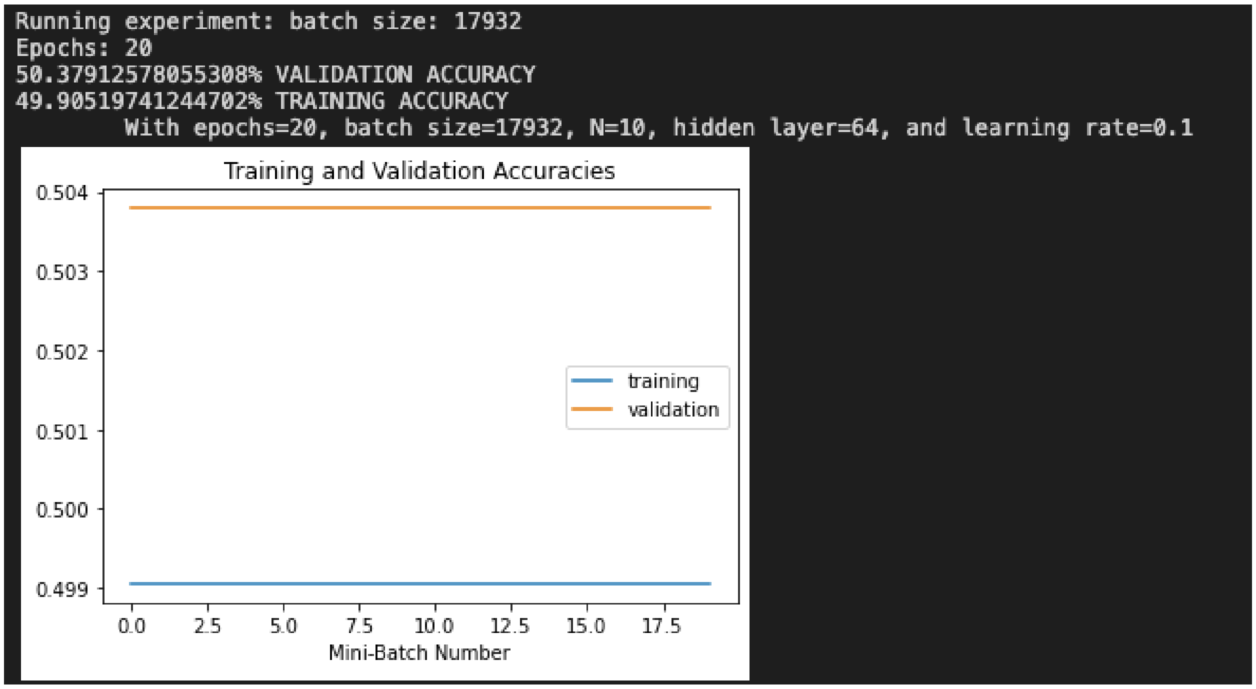
Plots:

```
Running experiment: batch size: 1
Epochs: 1
74.21944692239072% VALIDATION ACCURACY
60.0% TRAINING ACCURACY
        With epochs=1, batch size=1, N=10, hidden layer=64, and learning rate=0.1
```

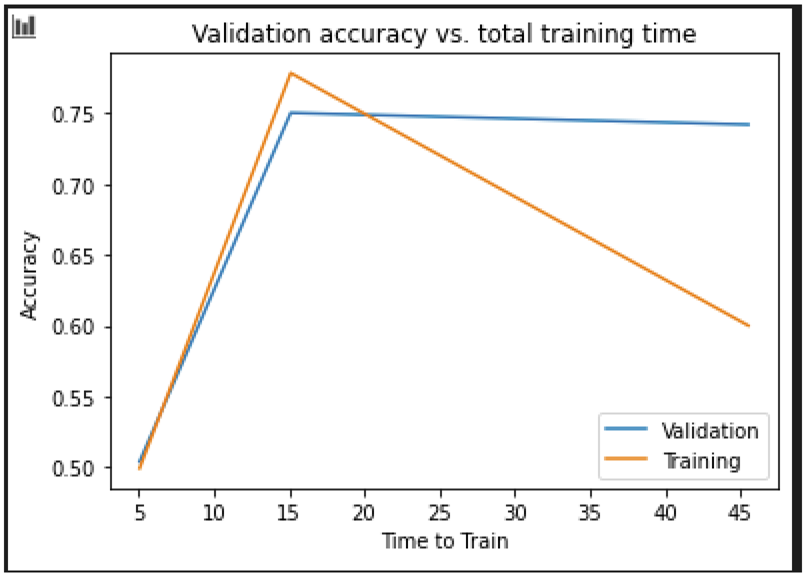


```
Running experiment: batch size: 64
Epochs: 20
75.04460303300624% VALIDATION ACCURACY
77.8645833333333% TRAINING ACCURACY
        With epochs=20, batch size=64, N=10, hidden layer=64, and learning rate=0.1
```





```
Running experiment: batch size: 17932
Epochs: 20
50.37912578055308% VALIDATION ACCURACY
49.90519741244702% TRAINING ACCURACY
        With epochs=20, batch size=17932, N=10, hidden layer=64, and learning rate=0.1
```



Questions:

1. Batch size of 64 gives the highest validation accuracy.
2. Batch size of 64 is the fastest at reaching a good training accuracy.
3. A batch size that is too low increases training time and requires a very small learning rate to work properly, lest the parameters are changed erroneously due to variance between samples. Too large of a batch size leads to

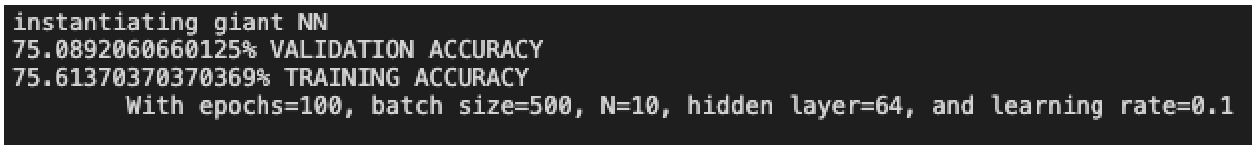
extremely slow (or in this case, no) convergence or apparent change in parameters values.

4. Smaller batch sizes can lead to more erroneous gradient measurements as the model is adjusted due to loss on a small number of training samples rather than a larger number. Small batches are more likely (per the central limit theorem) to not reflect the 'real' characteristics of the data distribution. Meanwhile, a large batch size has a poor ratio of computation per meaningful adjustment of the model. The batch size value should have an order of magnitude roughly at the center point between 1 and the dataset size.

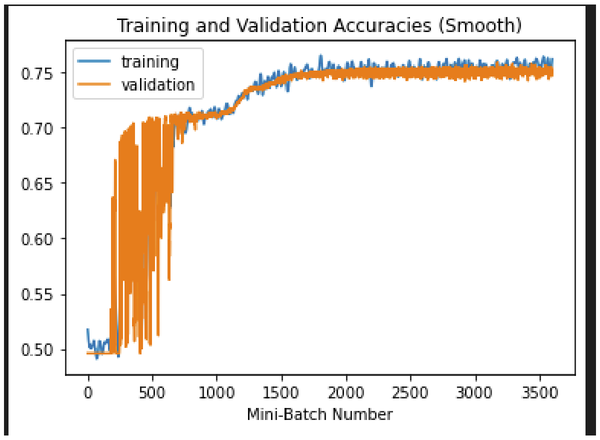
5.4: Under-Fitting

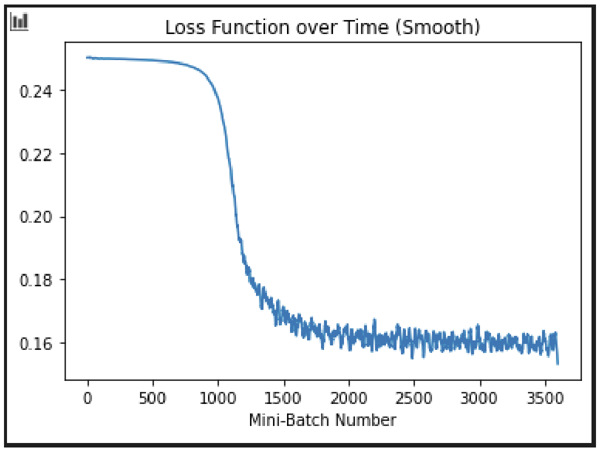
The model does **not** appear to be under-fitting as its performance is on par with the performance of the previous model and it generalizes well between the testing and training dataset.

5.5: Over-fitting



```
instantiating giant NN
75.0892060660125% VALIDATION ACCURACY
75.61370370370369% TRAINING ACCURACY
        With epochs=100, batch size=500, N=10, hidden layer=64, and learning rate=0.1
```

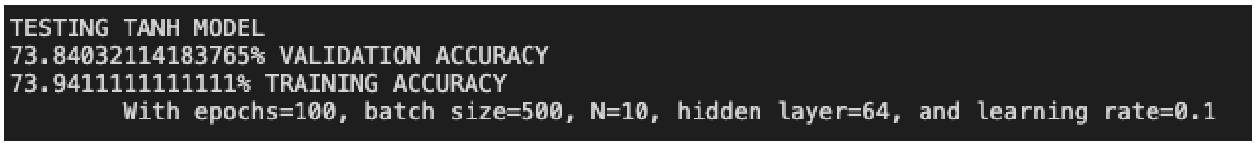




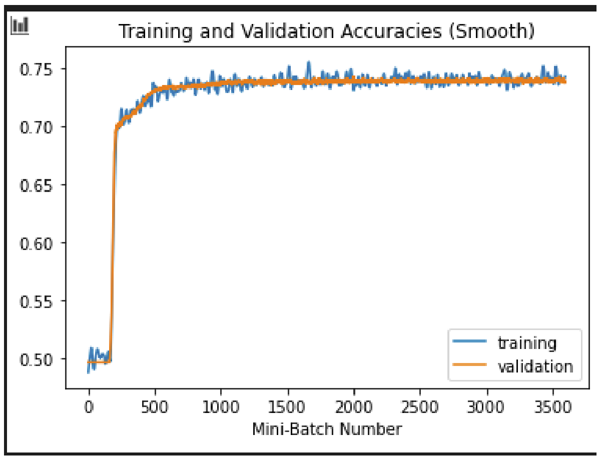
The model does not appear to be over-fitting as its performance on the training and validation sets are roughly equivalent. It does certainly take longer than before to converge, however.

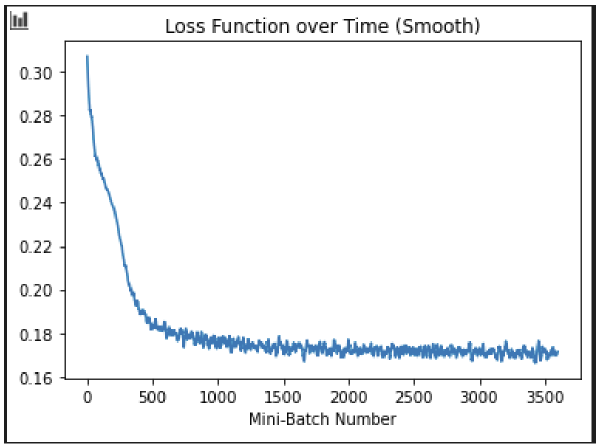
5.6: Activation Function

Tanh Model

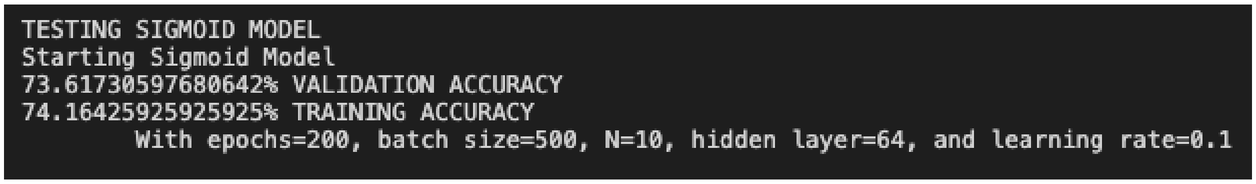


```
TESTING TANH MODEL
73.84032114183765% VALIDATION ACCURACY
73.94111111111111% TRAINING ACCURACY
        With epochs=100, batch size=500, N=10, hidden layer=64, and learning rate=0.1
```

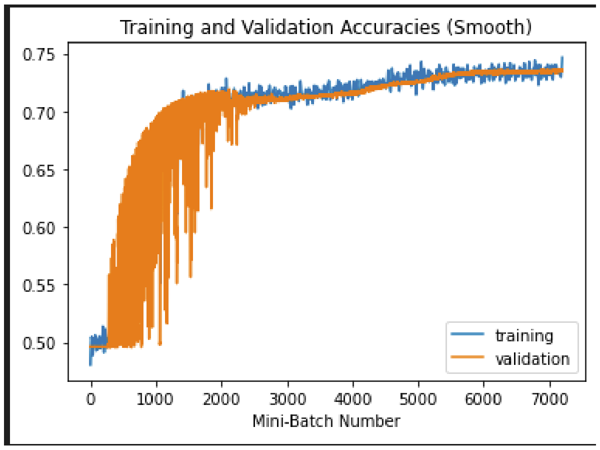


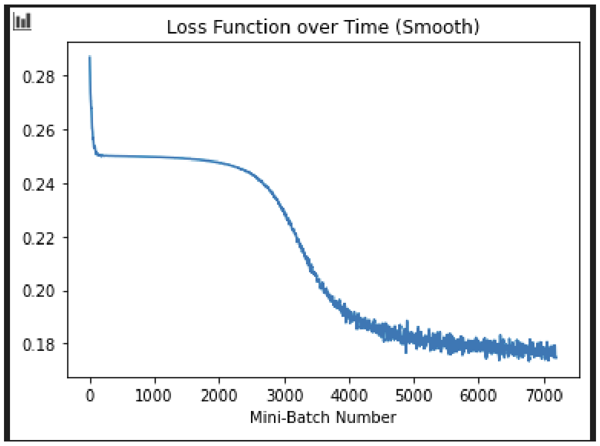


Sigmoid Model



```
TESTING SIGMOID MODEL
Starting Sigmoid Model
73.61730597680642% VALIDATION ACCURACY
74.16425925925925% TRAINING ACCURACY
        With epochs=200, batch size=500, N=10, hidden layer=64, and learning rate=0.1
```



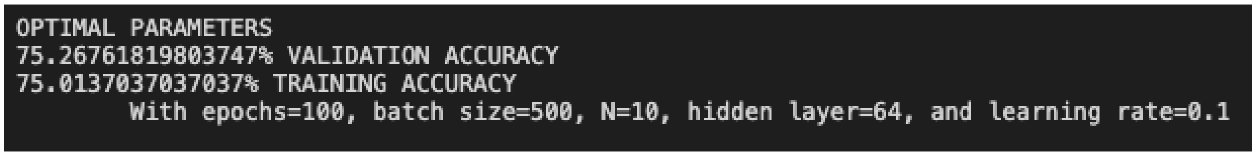


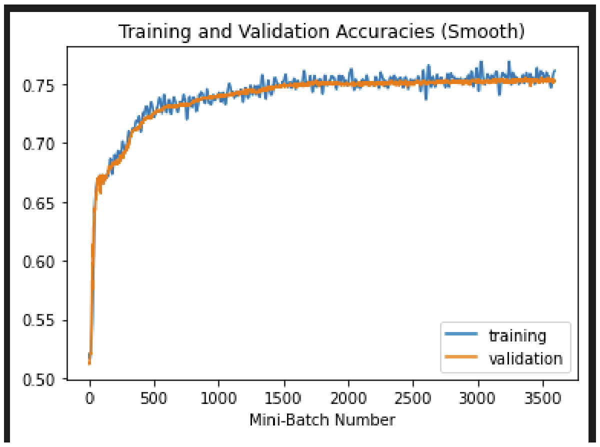
Timing:

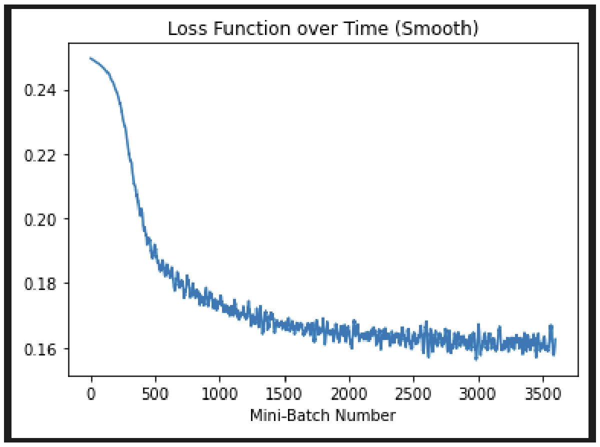
- TanH: 22.816592222996405
- Sigmoid: 49.319170470000245

Sigmoid takes much longer.

5.7: Hyperparameter Search


```
OPTIMAL PARAMETERS
75.26761819803747% VALIDATION ACCURACY
75.0137037037037% TRAINING ACCURACY
        With epochs=100, batch size=500, N=10, hidden layer=64, and learning rate=0.1
```





6: Feedback

1. I spent roughly 9 hours on assignment 3.
2. I at times found it challenging to interpret the requirements stated in the assignment.
3. I enjoyed working with real data and practicing the skill of making a 'real world' data analysis.
4. I found certain parts of the instructions confusing (model layer/activation function specifications, requirements for time vs. accuracy plots, etc.)
5. I found it helpful that pointers were given in terms of what useful functions we might use for various tasks.

Overall I think that the assignment could be made shorter, but it was an enjoyable and useful experience. I was also unable to get my model to perform at the supposed 80% or higher level, despite following all the steps.