ECE324 Assignment 3

Frederick Boyd

**NOTE:** unless otherwise stated, a seed of 1 was used

# PyTorch & Single Neuron Classifier from Assignment 2

1. *For the single-neuron classifier that you instantiate, what is the full name of the tensor object that contains the weights, and what is the name of the object that contains the bias?*

The full name of the tensor object that contains the weights is fc1.weight.

The full name of the tensor object that contains the bias is fc1.bias.

1. *What is the name of the tensor that contains the calculated gradients of the weights and the bias?*

fc1.grad

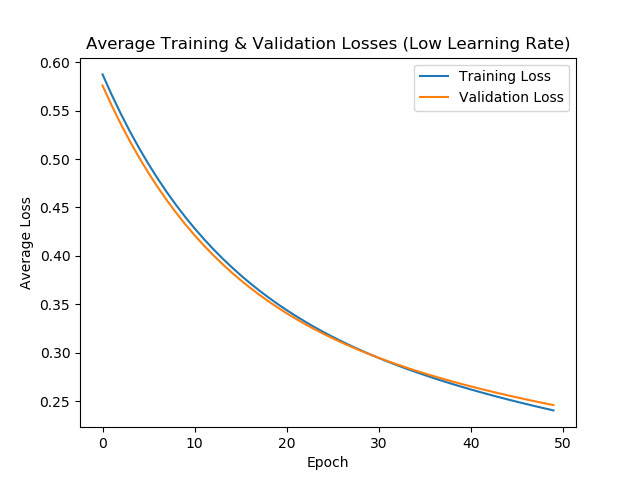
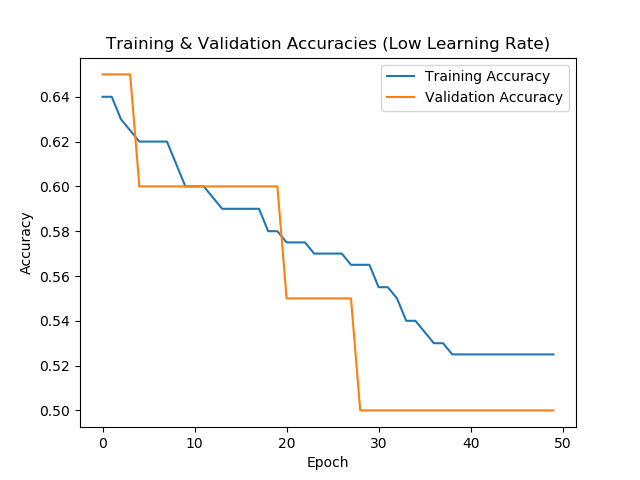
1. *Which part of your code computes gradients (i.e. give the line of code that causes the gradients to be computed). Explain, in a general way, what this line must cause to happen to compute the gradients and how PyTorch ‘knows’ how to compute the gradients.*

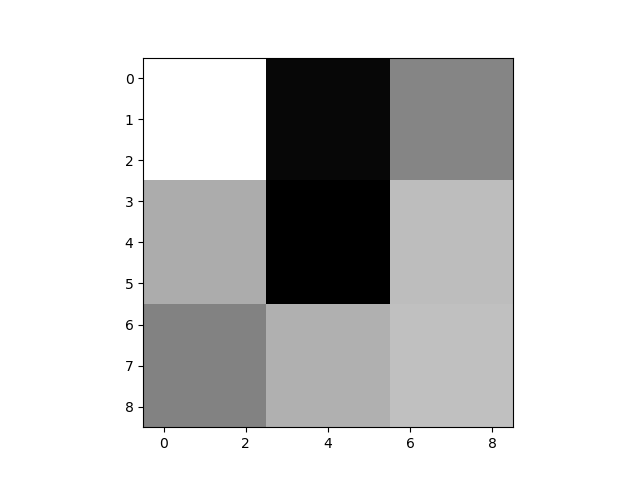
The line of code which causes gradients to be computed is loss.backward(). In general, this line must cause PyTorch to calculate the partial derivative of the loss function with respect to each parameter. In this case, it needs to calculate ten gradients each time it is called: nine for each weight and one for the bias.

PyTorch knows how to compute the gradients because the weight and bias tensors have a requires\_grad flag that is set to true by default. It lets PyTorch know that we want to calculate the gradient of these values.

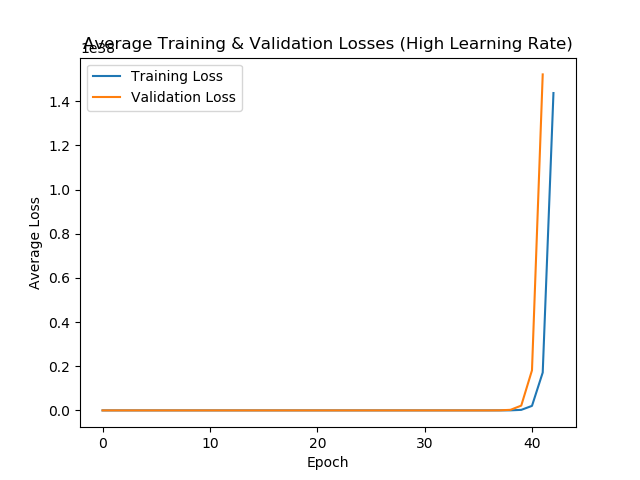
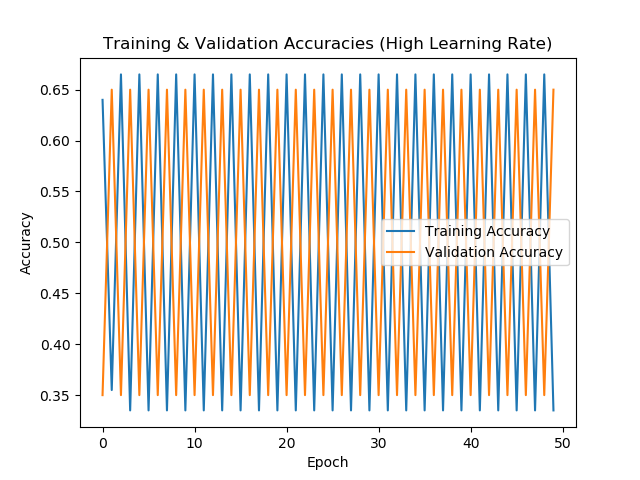
1. *Give the training/validation plot versus epoch, and the accuracy vs. epoch for the three cases required in Assignment 2 at the end: a too-slow learning rate, a too-fast learning rate, and a ‘just-right’ learning rate.*

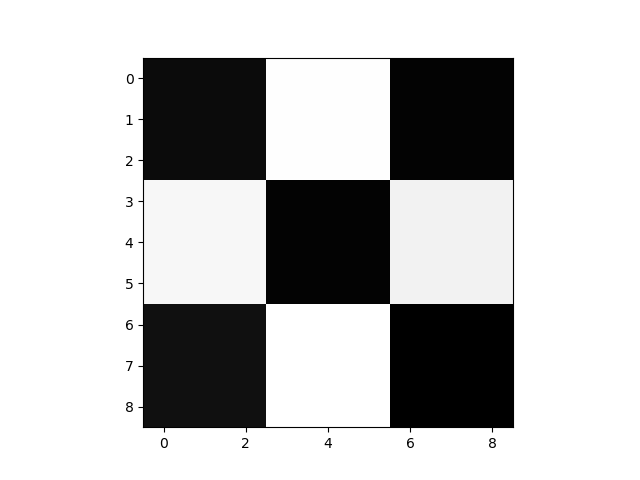
See Figures 1-9 below.



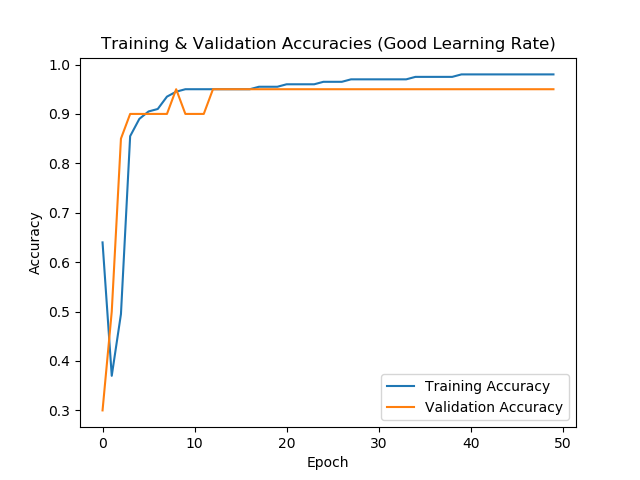
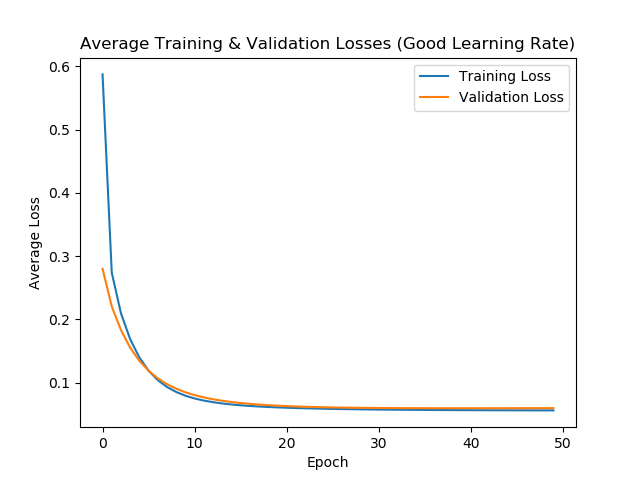


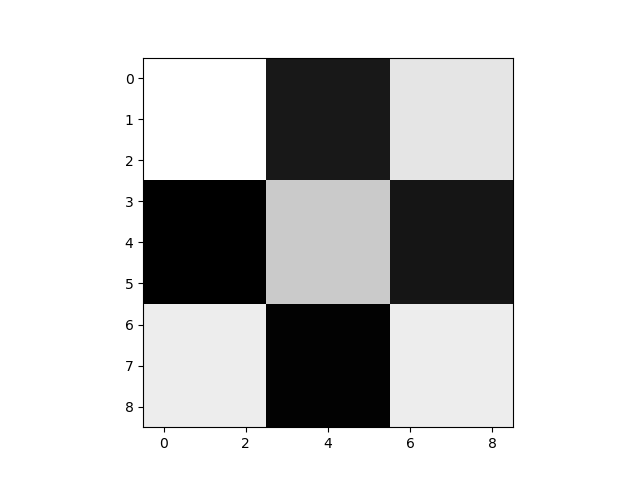
Figures 1, 2, 3: Plots of training and validation accuracies and losses using a training rate of 0.005 that is too low. There is also a visualization of the final weights using the dispKernel function.





Figures 4, 5, 6: Plots of training and validation accuracies and losses using a training rate of 0.5 that is too high. There is also a visualization of the final weights using the dispKernel function.



Figures 7, 8, 9: Plots of training and validation accuracies and losses using a training rate of 0.15 that is good. There is also a visualization of the final weights using the dispKernel function.

# Data Pre-processing and Visualization

## Understanding the Dataset

1. *How many high income earners are there in the dataset? How many low income earners?*

There are 37155 low income earners and 11687 high income earners in the dataset.

1. *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 dataset is not balanced because there are an unequal number of low and high income earners. Training with unbalanced data makes it more difficult to detect rare things in the dataset. For example, if we are trying to detect insurance fraud, a model that always reported claims as valid would be correct most of the time. Essentially, this could result in the model overfitting to always report claims as valid regardless of the actual data contained within the claim.

## Cleaning

1. *How many samples (rows) were removed by the above cleaning process? How many are left?*

The original dataset has 48842 rows. After cleaning there are 45222 rows. As such, 3620 rows were removed by the cleaning process.

1. *Do you think this is a reasonable number of samples to throw out?*

This seems like a reasonable number of samples to throw out, as it is only about 7% of the original dataset.

## Visualization and Understanding

1. *What is the minimum age of individuals and the minimum number of hours worked per week for the dataset?*

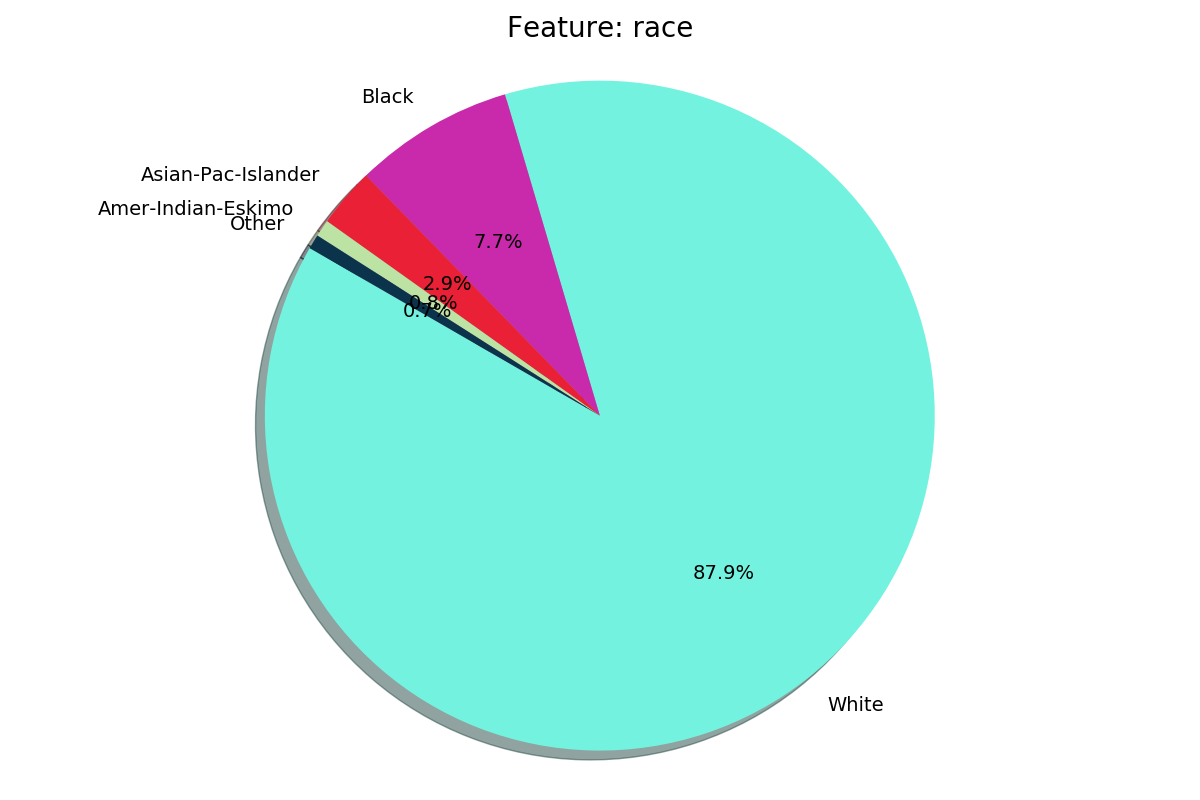
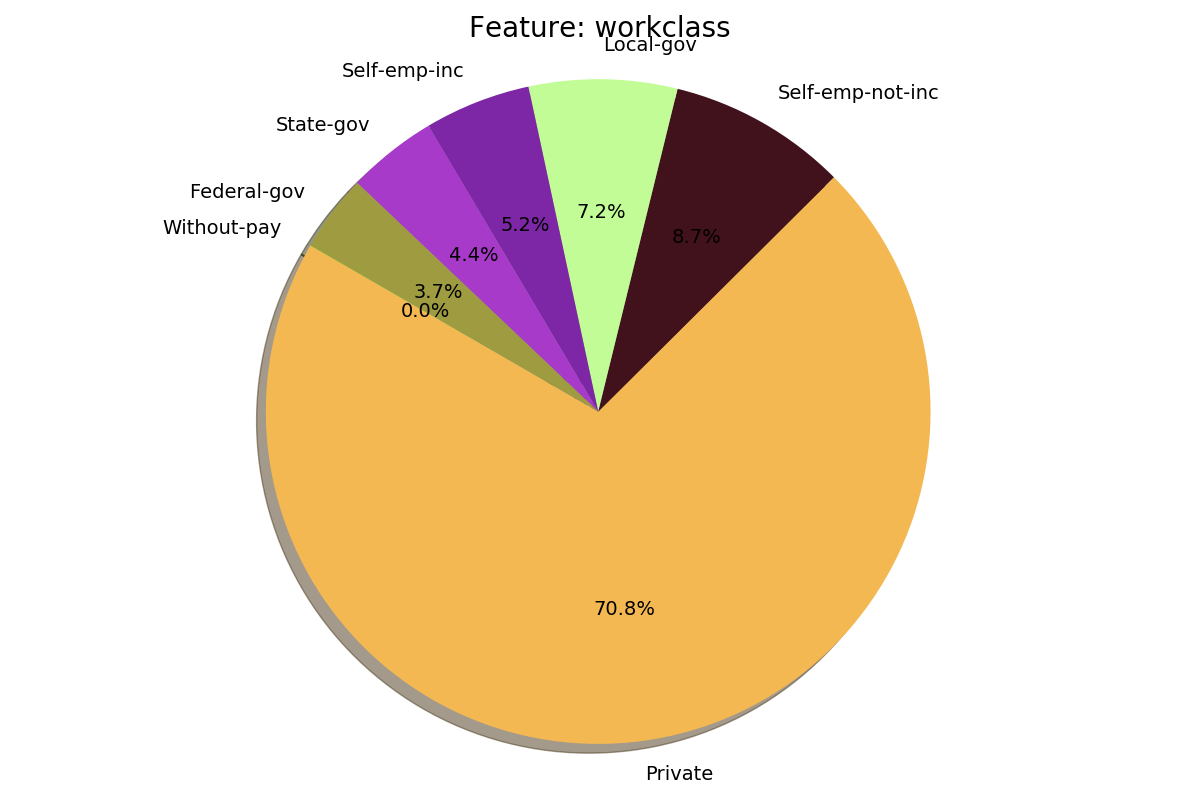
Minimum age: 17

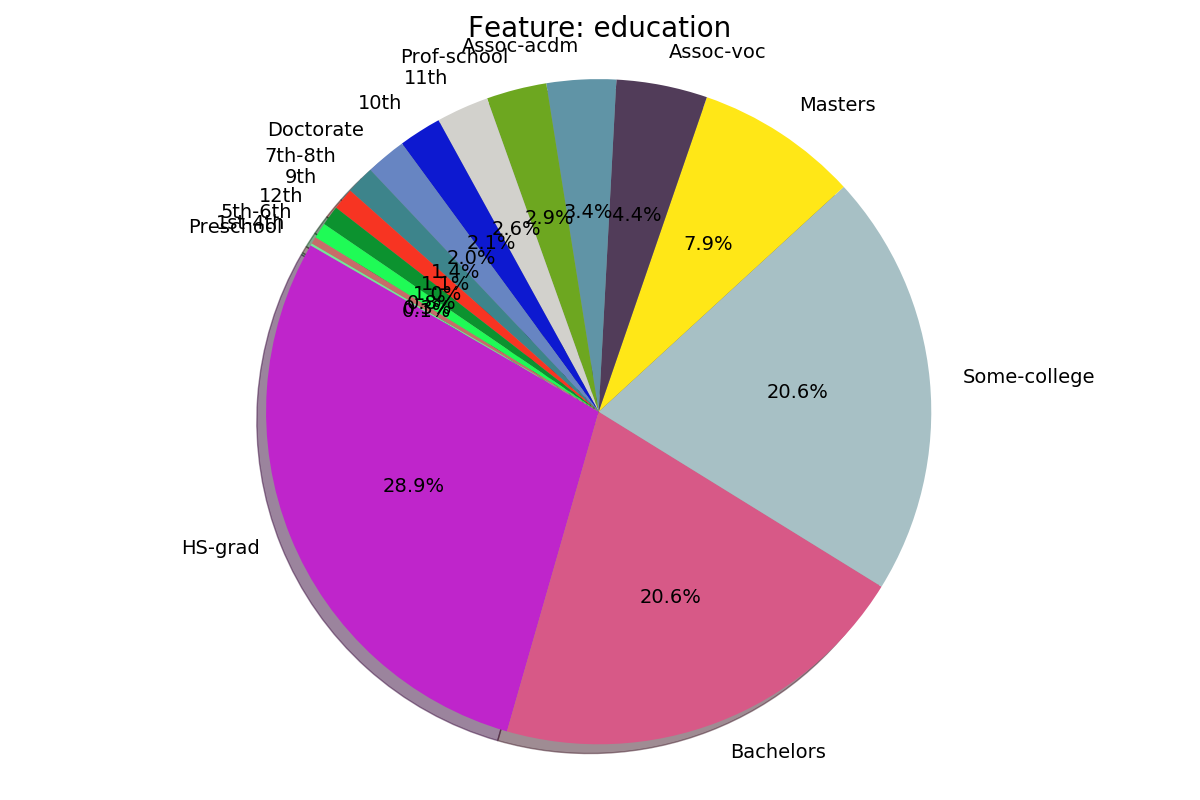
Minimum hours per week: 1

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

For the work class and race features, the private sector and white race, respectively, are over-represented. This can be seen in Figures 1 and 2 where approximately three quarters of each pie chart is consumed by that group. For the education feature, groups with very low and high levels of education are under-represented, whereas high school diplomas, college degrees, and bachelor’s degrees make up most of the dataset.

I would not expect the results trained on this dataset to generalize well to different races, a variety of work classes, or different levels of education.

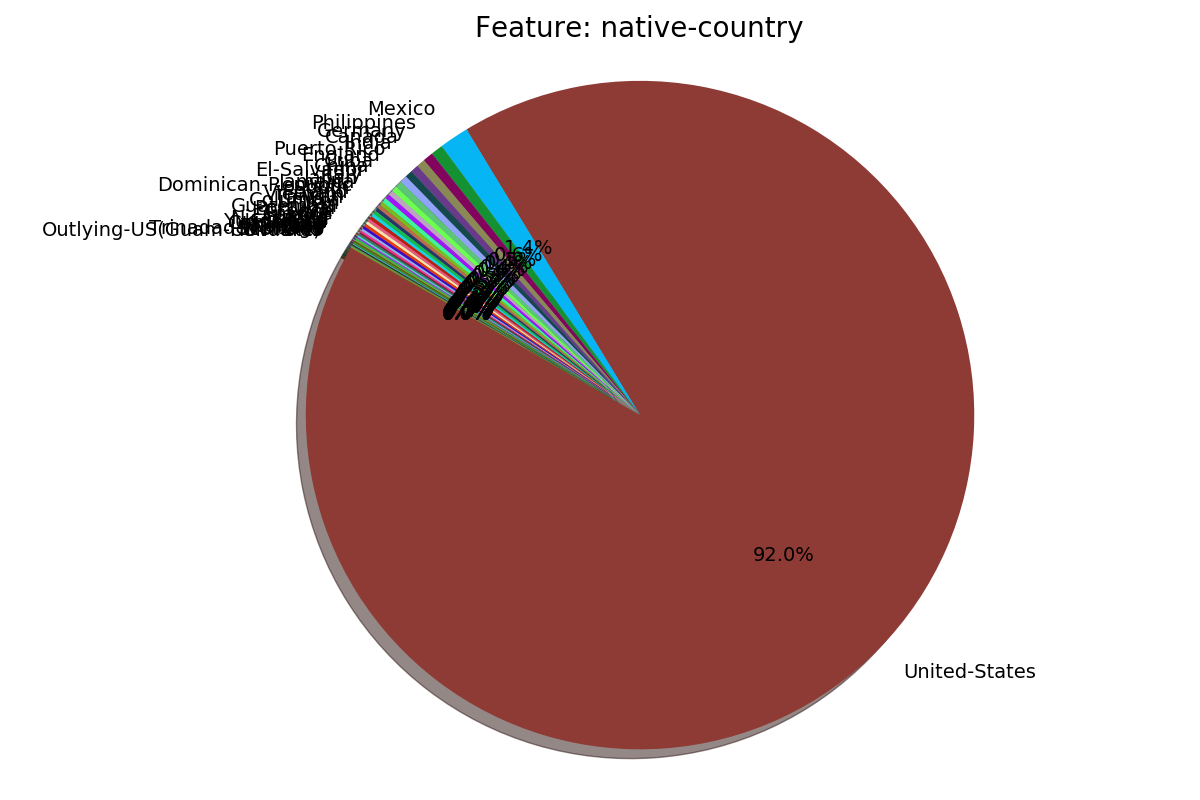
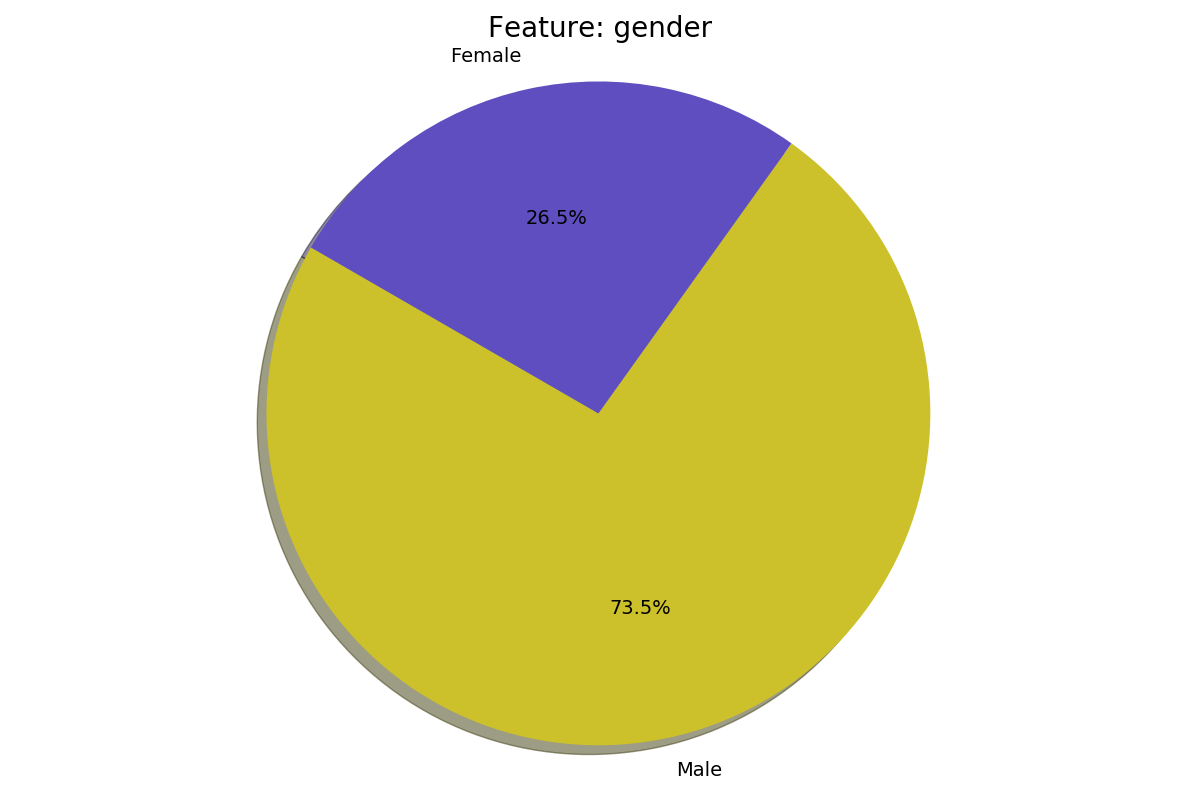




Figures 1, 2, 3: Pie chart of the first three categorical features in the adult dataset.

1. *What other biases in the dataset can you find?*

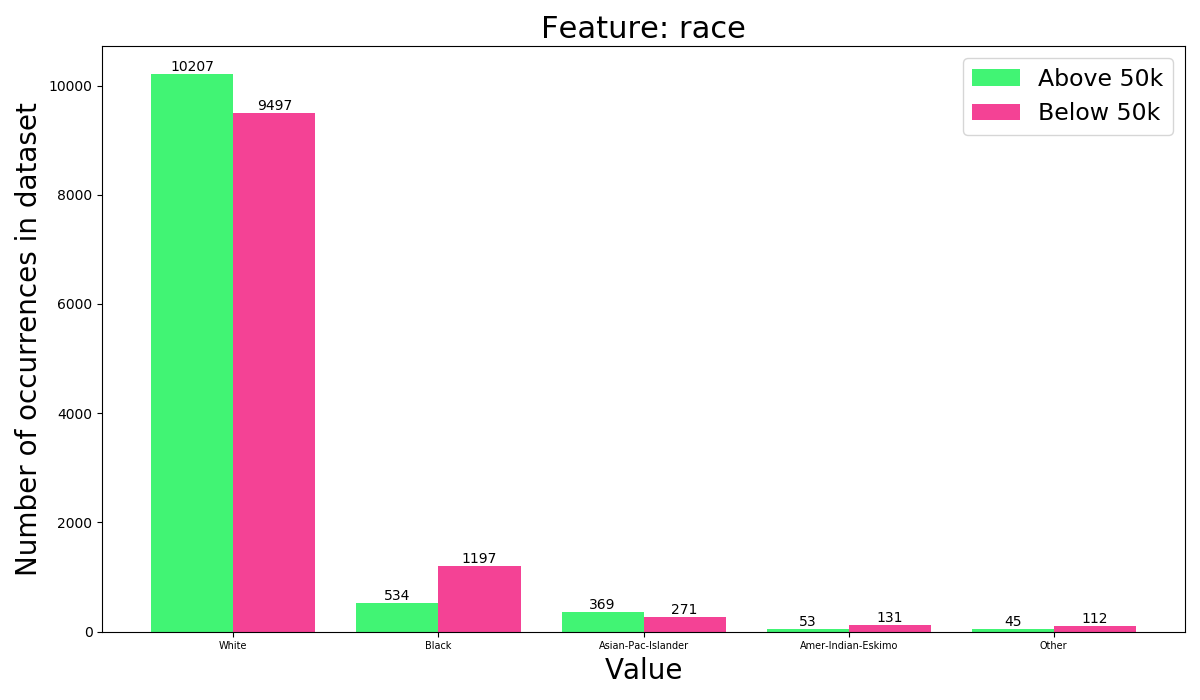
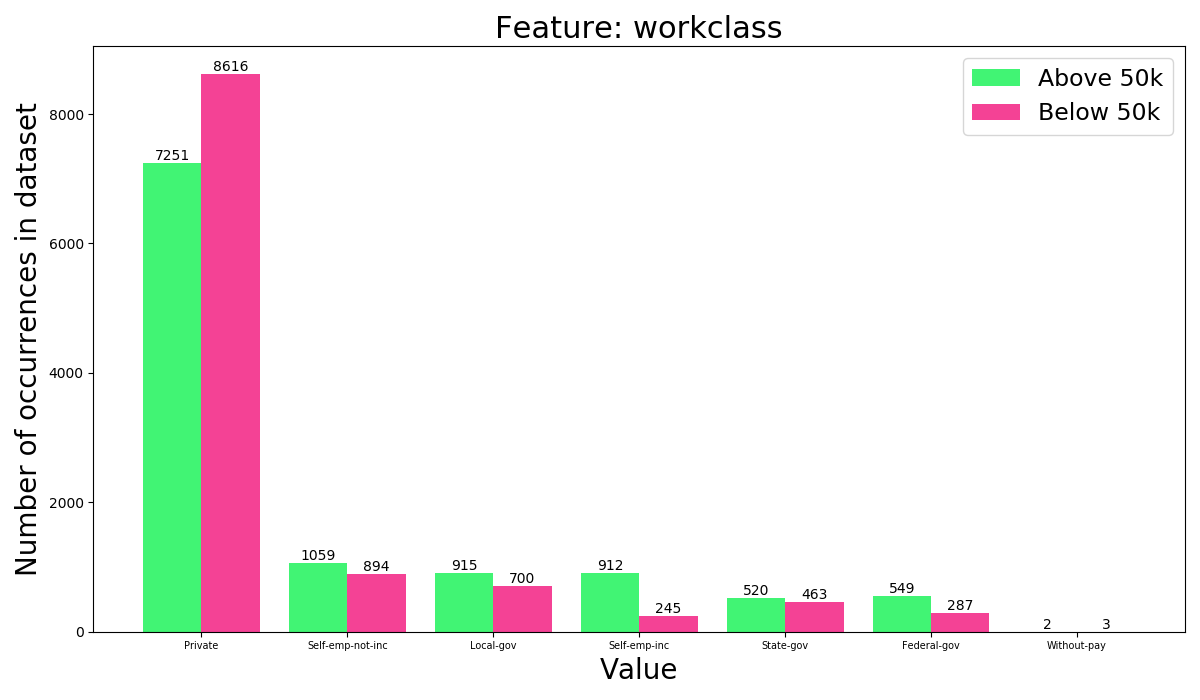
Other significant biases include the male-to-female ratio and native country. The dataset is approximately three quarters male and only one quarter female, which is not representative of reality. Additionally, over 90% of the data contains people whose native country is the United States. Although this data was obtained from a US census, immigrants should make up a greater percentage of the data in order to ensure the final model is generalized.

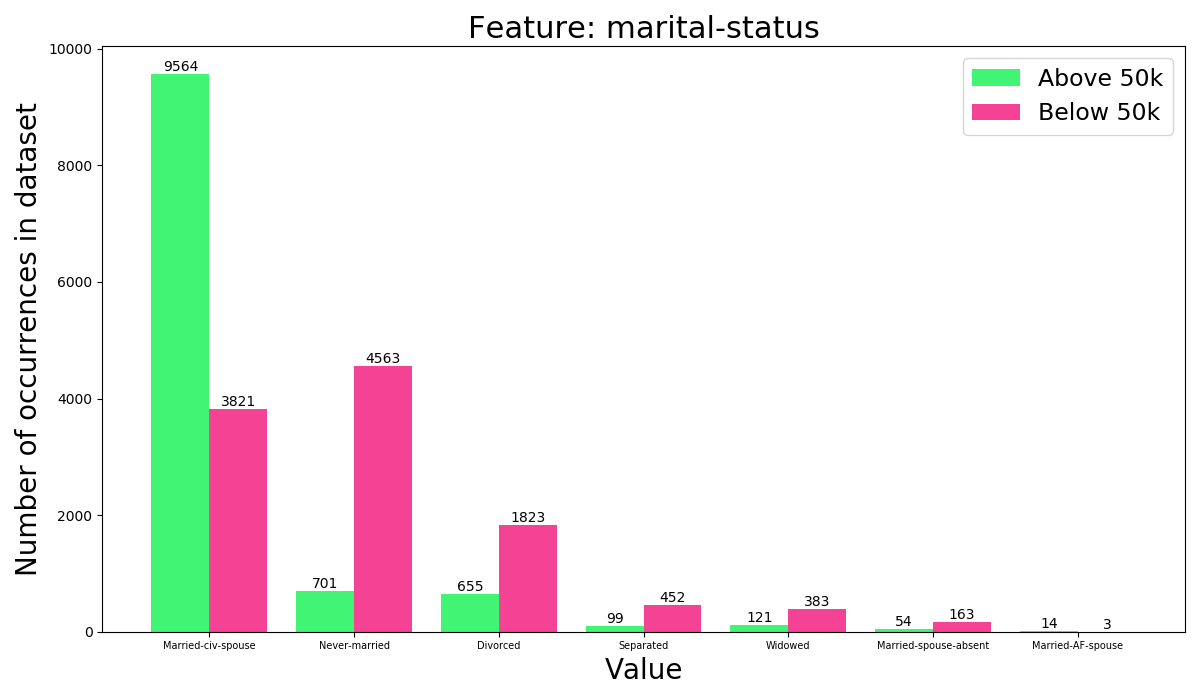
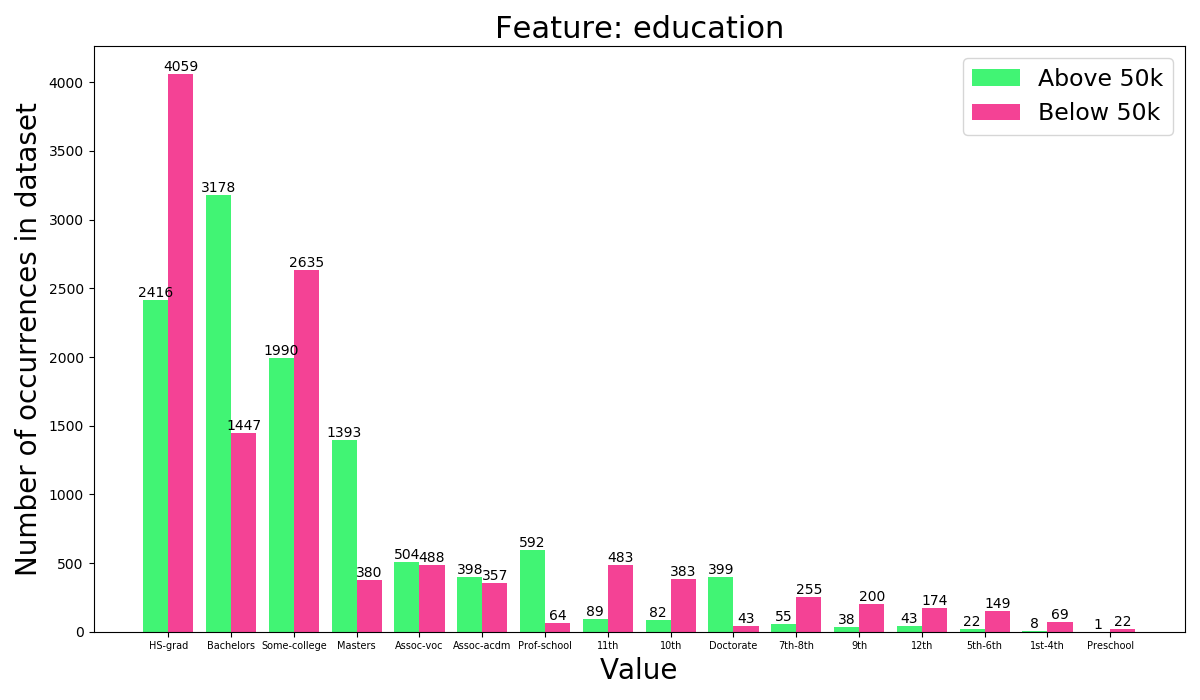


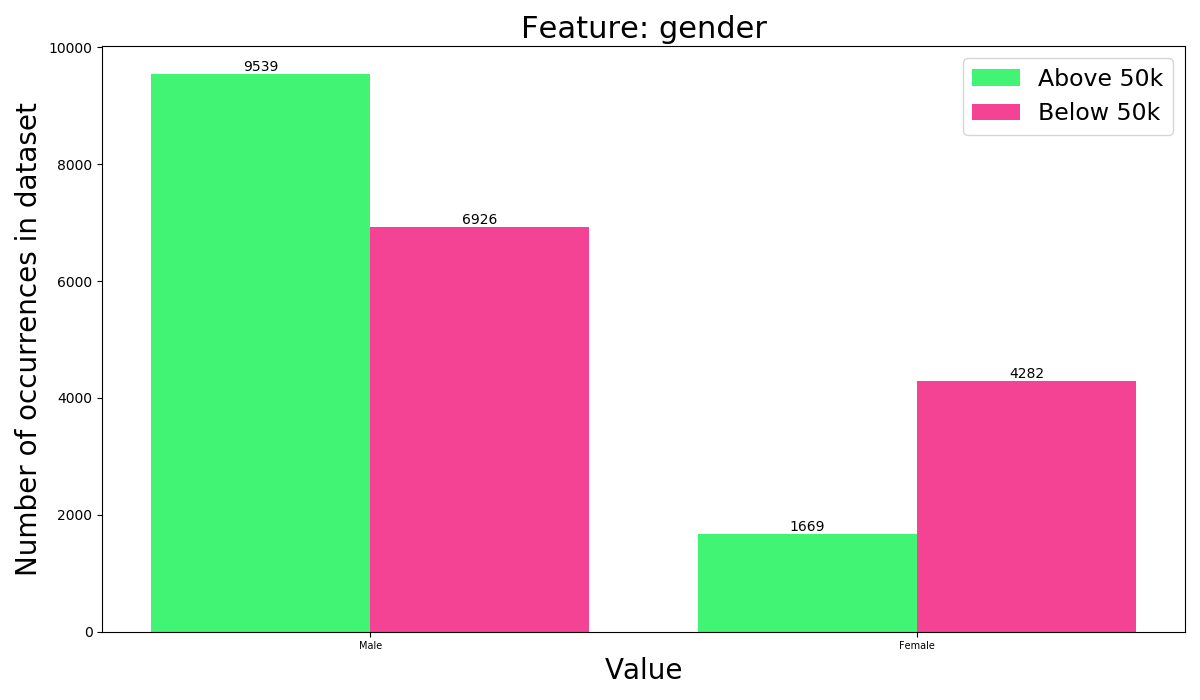
Figures 4, 5: Pie charts of other biases in the dataset. There is a male-female ration imbalance and a disproportionate number of people whose native country is the United States.

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

From observing the plots generated by the binary\_bar\_chart function for each categorical feature, the top three features useful for distinguishing between high and low salary earners are education, marital status, and gender. Binary graphs for these features can be found in Figures 8, 9, and 10.







Figures 6, 7, 8, 9, 10: Binary bar charts for work class, race, and education.

1. *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 a high school level of education, they have an approximately 37% chance of making more than 50K.

Given that someone has a “Bachelors”, it would be better to assume that they earn above 50K.

## Pre-processing

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

A disadvantage of using integer representation for categorical data is that the computer may see a fake hierarchy in the data. When looking at categorical data, there is typically no hierarchy between the various categories. However, when representing these categories as integers, one category may have a larger or smaller number than another category. This leads to the computer seeing that there is some order in the categories where there is none, since all the categories are equal. As such, this may impact the training results.

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

Using un-normalized continuous data exposes the neural network to a wide range of numbers. For example, the continuous data in the adult dataset ranges from 0 to 1490400. Some categories of continuous data will inherently be much larger on average than others. These larger values will have a greater influence on the performance of the neural network, when ideally each continuous category of data should be considered equally during training. Normalizing the data significantly reduces the its range while preserving each column’s relative values.

# Model Training

## DataLoader

1. *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?*

If data was presented to the neural network without shuffling, it will overfit for the training data because it will always be presented in the same order, leading to poor performance during validation. Additionally, if the data is presented in a particular order, the neural network will not become generalized enough because it will only have learned about that specific label for a while.

## Model

1. *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 first layer should be roughly between the size of the input and output layers. Using too few neurons (i.e. a number close to the output layer size) will result in underfitting because there will be too few neurons in the layer to detect signals in the dataset. Having too many neurons may result in overfitting and the network taking too long to train. If there are too many neurons and not enough training data, the training loop may be unable to properly optimize all the weights. Also, more neurons mean that more computations must be done in order to run a single input through the network, thereby increasing the training time.

Consequently, the size of the first layer was chosen to be 50, which is approximately half the size of the input layer. The size of the output layer was 1 since this is a binary classification problem.

1. *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 of the neural network as a number between 0 and 1 because this is a binary classification problem. The output value is a probability because the neural network is only making a prediction based on previous data it has seen – it is not certain that the prediction it is making will be correct. As such, rather than outputting a binary value, it reports a probability between all the possible outputs.

The outputs from the network are 0 if it is predicted that income is <=50K and 1 if income is >50K.

## Validation

1. *Using Matplotlib, make a plot of the training accuracy and a plot of the validation accuracy as a function of the number of gradient steps (not per epoch as previously, but per mini-batch). For the training accuracy, you don't need to report the accuracy of the entire train dataset (this will take too long to run), just report the accuracy on the last N min- batches. Include the plots in your report. Report the batch size, MLP hidden size, and learning rate that you used. If you aren't getting over 80% validation accuracy, don't worry, as we will tune the hyperparameters in the next section. What validation accuracy does your model achieve?*

* Batch size: 1000
* MLP hidden size: 50
* Learning rate: 0.15
* Epochs: 20
* Eval every: 10
* Validation accuracy: 80.4%

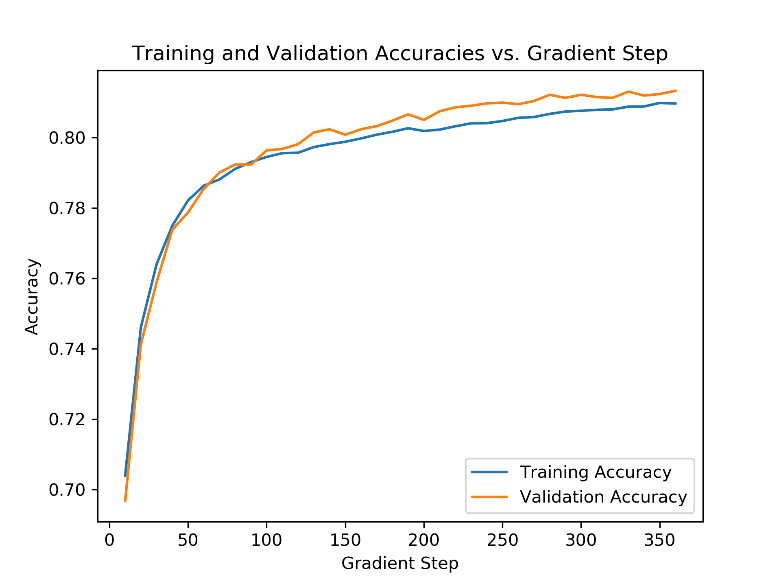
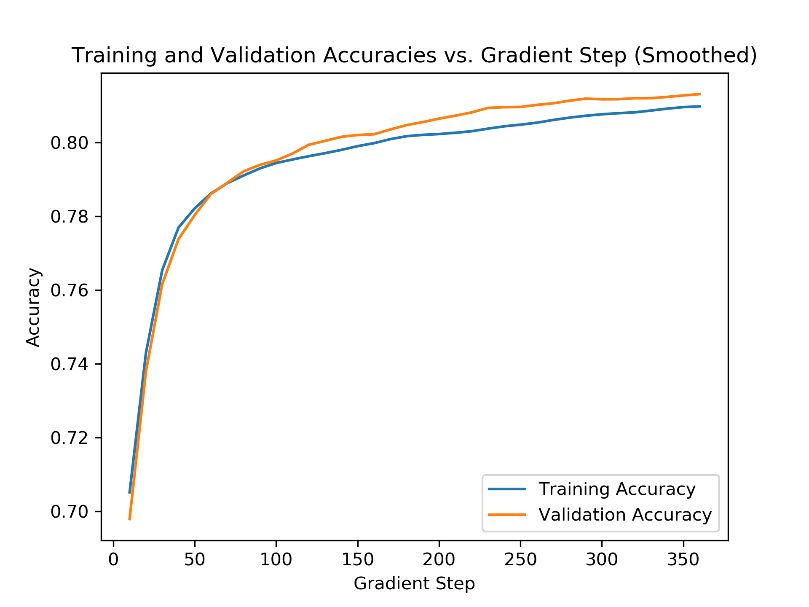


Figure 11: Plot of training and validation accuracies as a function of the gradient step.

1. *The performance of your network may oscillate. Add smoothed plots to your graphs better visualize the results. You can use any smoothing algorithm you wish; one possibility is the savgol filter from scipy.signal. Play around with the smoothing algorithm's parameters to get a plot that is less noisy but maintain the essence of the original plot. If N is small, the training error may oscillate a lot. In this case, you can apply high smoothing to the train plot or increase N and batch size.*



Figures 12, 13: Plots of training and validation accuracies as a function of the gradient step. The plot on the right is smoothed using the savgol\_filter from scipy.signal using a window\_length of 7 and a polyorder of 3.

# Hyperparameters

## Learning Rate

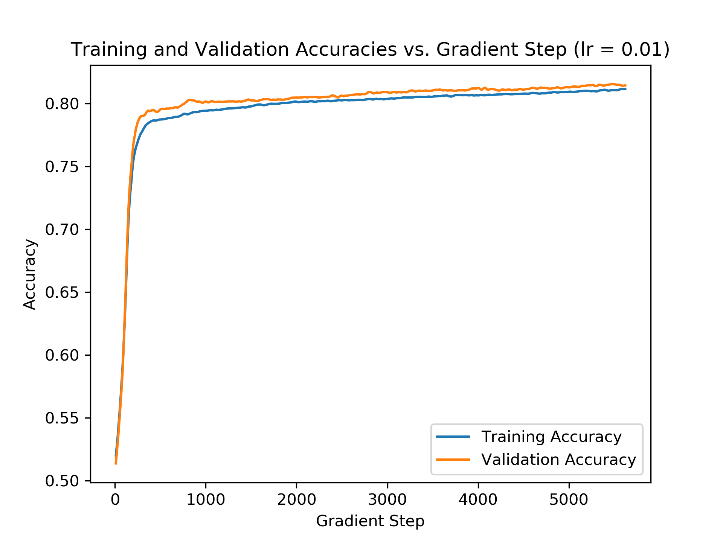
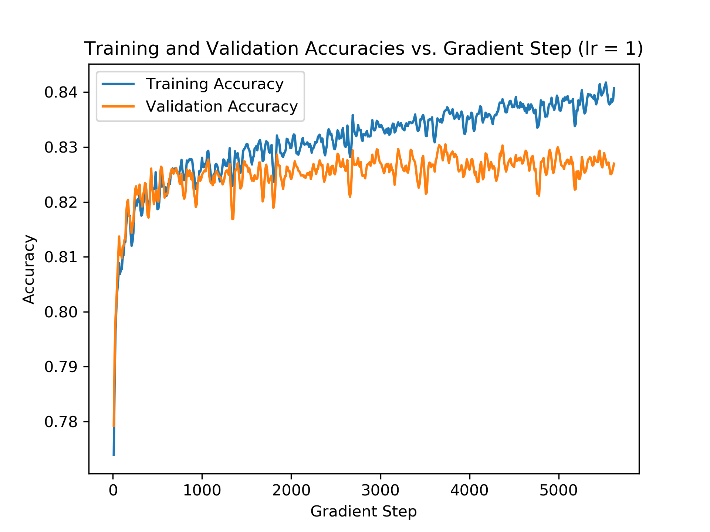
1. *Report the highest validation accuracy for each learning rate in a table. Include this table in your report.*

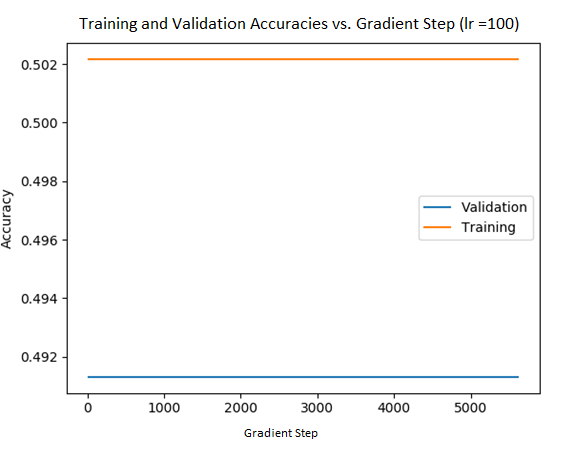
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Learning Rate* | *0.001* | *0.01* | *0.1* | *1* | *10* | *100* | *1000* |
| *Validation Accuracy (%)* | 79.66 | 81.42 | 82.62 | 82.62 | 83.78 | 50.22 | 50.87 |

1. *Make a plot of training accuracy (achieved on the training data) and validation accuracy (achieved on the validation data) as a function of the number of steps for learning rate equal to 0.01, 1, and 100. Include the plots in your report.*

Hyperparameters:

* Batch size: 64
* Epochs: 20
* Eval every: 10
* Hidden layer size: 64



Figures 14, 15, 16: Training and validation accuracies plotted against the gradient step for learning rates of 0.01, 1, and 100, respectively.

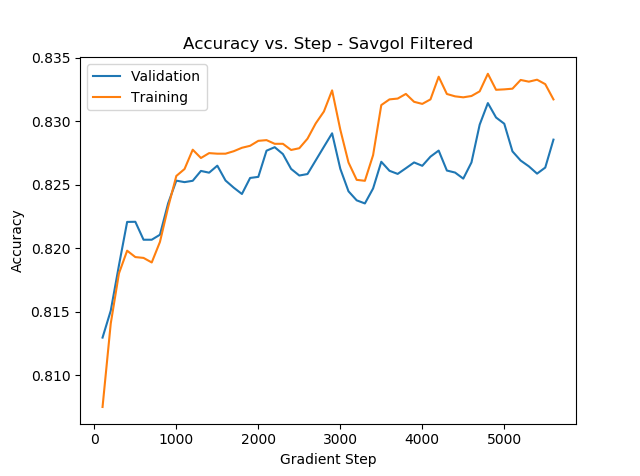
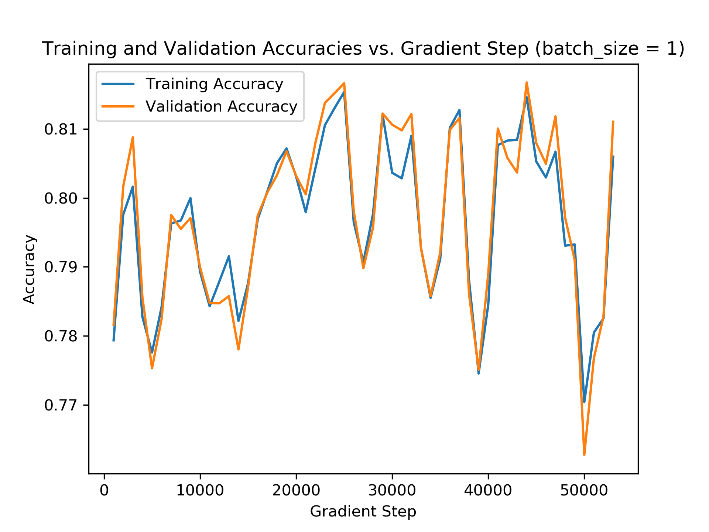
## Batch Size

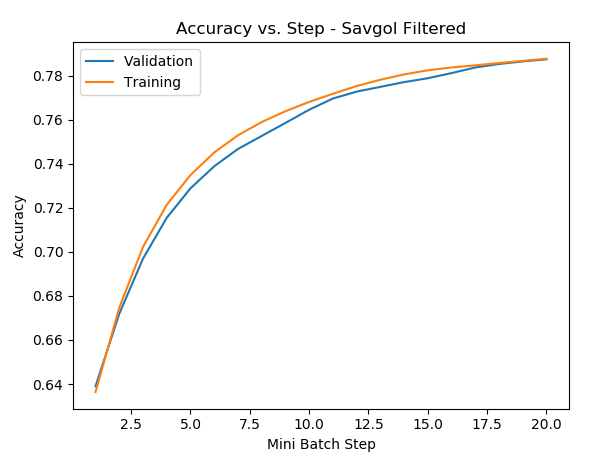
Plots comparing the accuracies of batch sizes 1, 64, and 17932 can be found in Figures 17 – 19.

Plots comparing the accuracies against time for batch sizes 1, 64, and 17932 can be found in Figures 20 – 22.

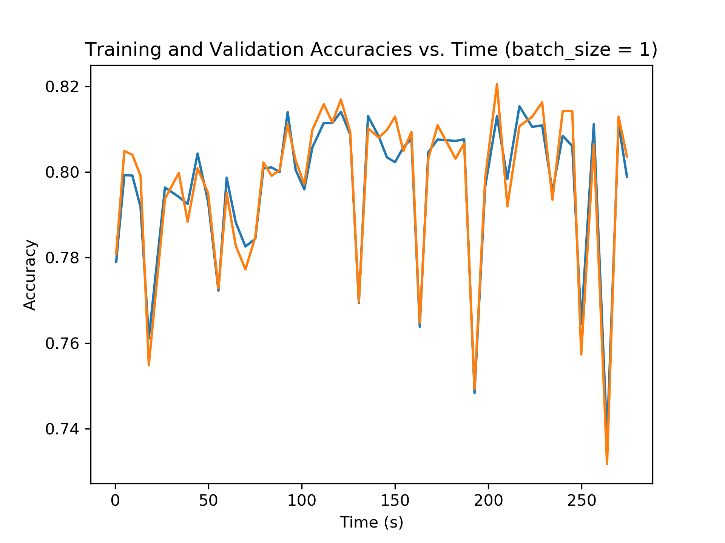
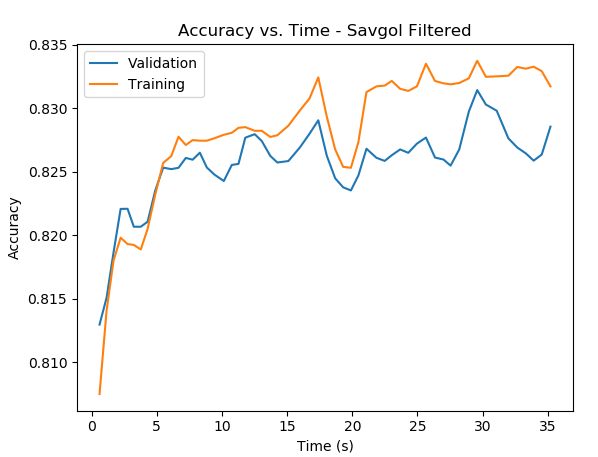
Hyperparameters:

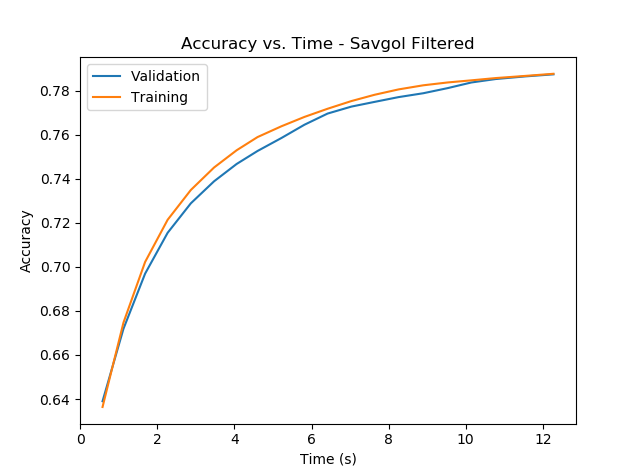
|  |  |  |  |
| --- | --- | --- | --- |
| *Batch Size* | *Learning Rate* | *Epochs* | *Eval Every* |
| *1* | 0.5 | 3 | 1000 |
| *64* | 0.5 | 20 | 100 |
| *17932* | 0.5 | 20 | 1 |





Figures 17, 18, 19: Plots of training and validation accuracies against the gradient step for batch sizes 1, 64, and 17932, respectively.



Figures 20, 21, 22: Plots of training and validation accuracies against time for batch sizes of 1, 64, and 17932, respectively.

1. *Which batch size gives the highest validation accuracy?*

|  |  |  |  |
| --- | --- | --- | --- |
| *Batch Size* | *1* | *64* | *17932* |
| *Validation Accuracy (%)* | 80.3 | 83.2 | 78.7 |

A batch size of 64 gives the highest validation accuracy of 83.2%.

1. *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?*

The batch size of 17932 reached a high validation accuracy in the least number of steps, as seen in Figure 19.

The batch size of 17932 also reached a high validation accuracy in a relatively quick period, as seen in Figure 22. The batch size of 64 reached a similar level of accuracy of about 82-83% within 5 seconds, as seen in Figure 21. The batch size of one appears to reach an accuracy of over 80% very quickly, but it appears to be unstable as the accuracy fluctuates significantly while training. This instability can be observed in Figure 20.

1. *What happens if the batch size is too low? Too high?*

A batch size that is too low appears to result in the neural network becoming unstable during training. The training and validation accuracies fluctuate significantly and there is no clear upward trend, as seen in Figure 17. This also results in the neural network taking a significantly longer time to train.

A batch size that is too high trends upward but is unable to reach an accuracy above 80%. This accuracy is lower than using a medium batch size, assuming the number of epochs remains constant between the two. As seen in Figures 18 and 19, a large batch size (17932) results in final training and validation accuracies of around 78%, whereas a medium batch size (64) results in accuracies of around 83%. Although accuracies using a large batch size are relatively lower, it is faster to train when compared to a medium batch size, as seen in Figures 21 and 22. When using a medium batch size, the model takes about 35 seconds to train, whereas it only takes about 12 seconds when using a large batch size.

1. *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).*

|  |  |  |  |
| --- | --- | --- | --- |
| *Small Batch Size* | | *Large Batch Size* | |
| *Advantages* | *Disadvantages* | *Advantages* | *Disadvantages* |
| May reach relatively high accuracy quickly | Unstable behaviour, long training times, minimize chance of getting stuck in local minima | Quick to train, stable with clear upward trend | Relatively low accuracy |

Generally speaking, a batch size in the order of magnitude of (32 – 512) should be used. This range offers a reasonable balance between computational speed and speed of convergence. For large datasets, larger batch sizes should be used to reduce the amount of training time. Similarly, smaller batch sizes should be used for small datasets.

## Under-fitting

Hyperparameters:

* Batch size: 64
* Learning rate: 0.5
* Epochs: 20
* Eval every: 10
* Hidden layer size: 50

1. *Change your MLP model to have no hidden layers and only one linear layer that maps the input directly to output (but still apply the Sigmoid function before the output). Make a plot of the train and validation accuracy versus*

Removing the hidden layer of the MLP model appears to result in it underfitting the dataset. Using only one linear layer to train the model results in the validation accuracy being slightly higher than the training accuracy, as seen in Figure 23.

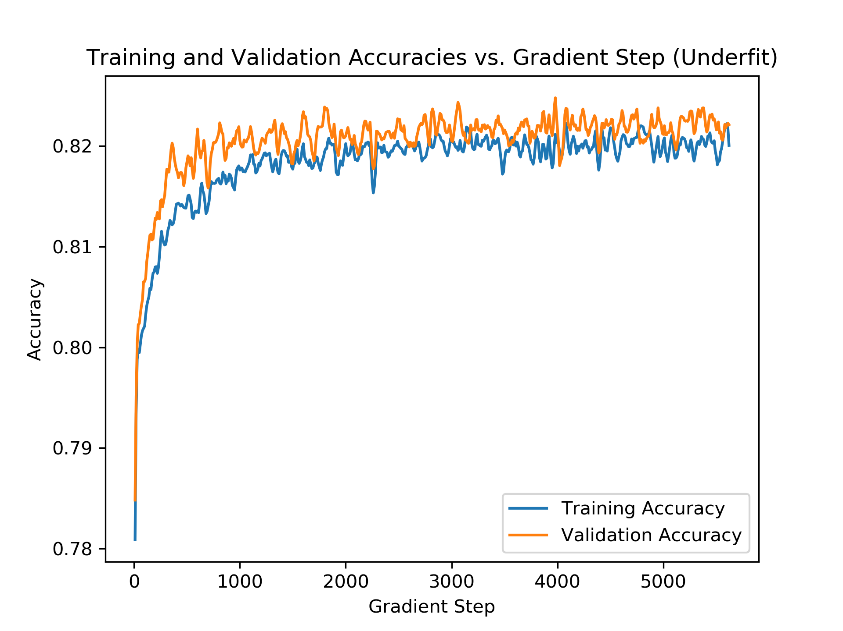


Figure 23: Example of underfitting the dataset by removing the hidden layer in the neural network. Note that the validation accuracy is higher than the training accuracy.

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 small model achieves a validation accuracy of 82.22%. This is very similar to the best model trained so far, which achieves a validation accuracy of around 85%.

As described in the previous question, this model is underfitting because the validation accuracy is greater than the training accuracy. This behaviour can be seen in Figure 23.

## Over-fitting

Hyperparameters:

* Batch size: 64
* Learning rate: 0.5
* Epochs: 20
* Eval every: 10
* Hidden layer size: 50

1. *Change your MLP model to have a total of four layers (and so we would say it has 3 “hidden” layers), with the hidden layers of size 64. Note that training might take a while longer. Make a plot of the training and validation accuracy versus the number of steps. Include the plots in your report.*

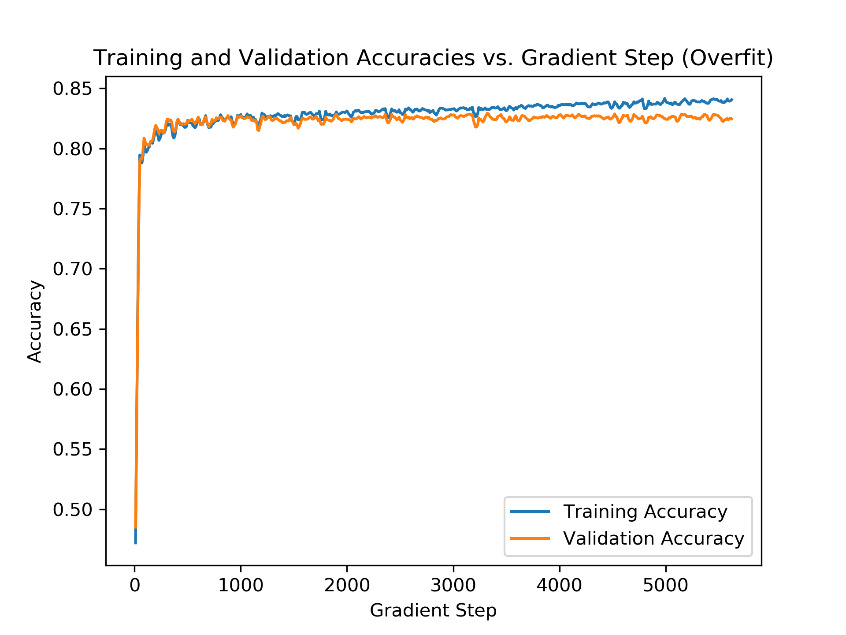


Figure 24: Example of overfitting the dataset by creating three hidden layers in the neural network.

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 overfitting?*

The large model achieves a validation accuracy of 82.43%. This is like the best model trained so far, which achieves a validation accuracy of about 83%.

This model is overfitting because there is a clear gap between the training and validation accuracies that continued to widen throughout the training process.

## Activation Function

Hyperparameters (best model):

* Batch size: 64
* Learning rate: 0.1
* Epochs: 20
* Eval every: 50
* Hidden layer size: 50

1. *Take the best model you've trained so far. Replace the ReLU activation function (used in the hidden layer(s)) in your model with a tanh, and then a Sigmoid activation function. Plot the train and the validation accuracy of all three models on the same graph. Do you notice any qualitative or quantitative differences? Include the plots in your report.*

Each activation function’s training and validation accuracies tend to stay fairly close to each other when compared to the difference between activation functions. This helps to visualize the impact of various activation functions on accuracies. From looking at Figure 25, it appears that the ReLU activation function produces the best results, then tanh, and finally sigmoid in last place.

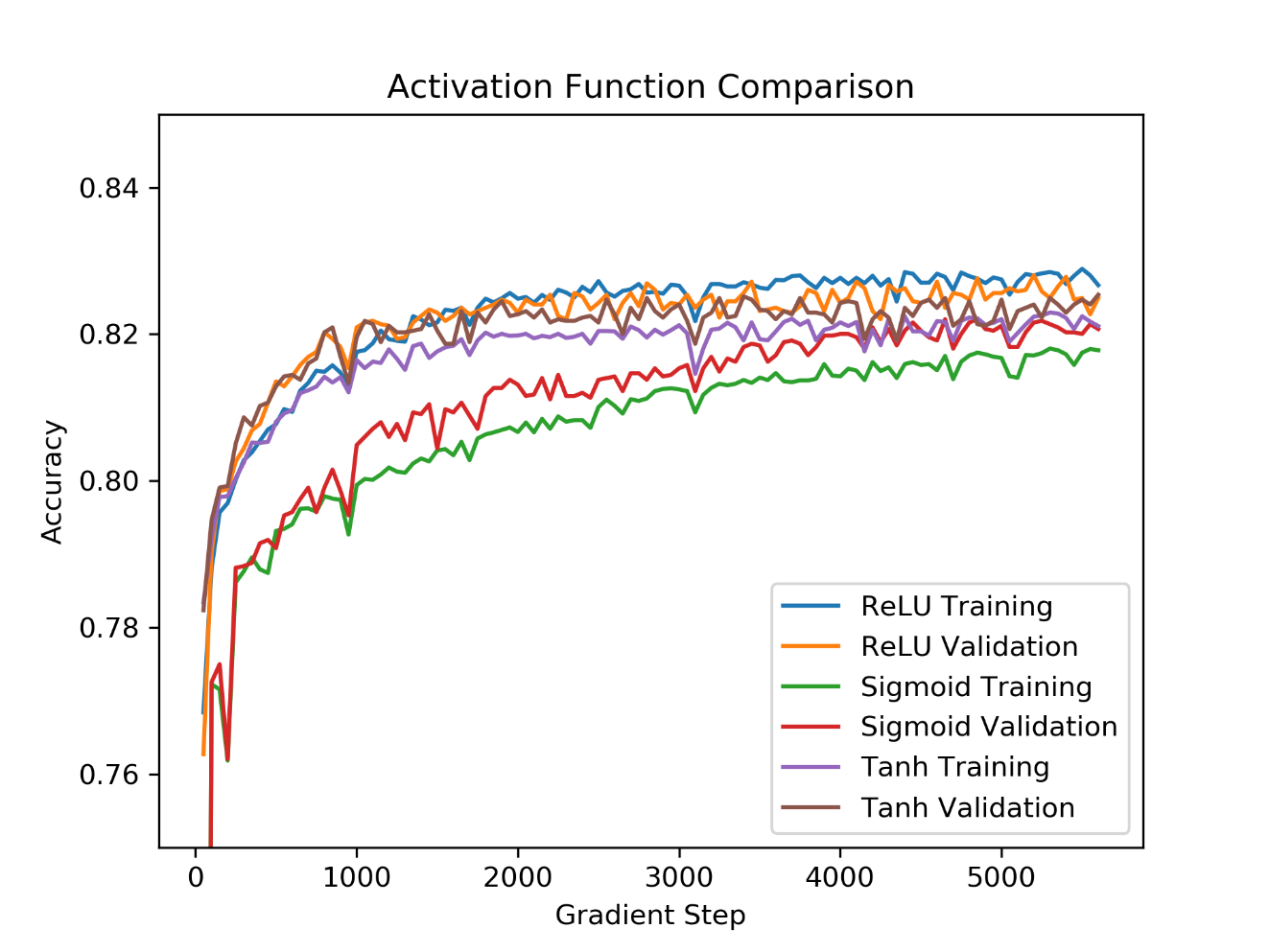


Figure 25: Comparison of various activation functions using the best hyperparameters detailed above.

1. *Measure the time of each of the training runs with each activation function. Include a table of the times in your report. Is there a difference between the activation functions in terms of how long they take to run?*

|  |  |  |  |
| --- | --- | --- | --- |
| *Activation Function* | *ReLU* | *Sigmoid* | *Tanh* |
| *Time* | 36.5 | 37.5 | 40.3 |

There is a difference between the activation functions in terms of how long they take to run. Their order from fastest to slowest is: ReLU, sigmoid, tanh.

## Hyperparameter Search

The following hyperparameters were determined to be the best after thorough tuning and testing:

* Batch size: 64
* Learning rate: 0.1
* Epochs: 20
* Eval every: 50
* Hidden layer size: 50
* Hidden layer activation function: ReLU
* Random seed: 1

Accuracies:

* Validation accuracy: 82.5%
* Training accuracy: 82.8%

# Feedback

1. *How much time did you spend on assignment 3?*

~20 hours

1. *What did you find challenging?*

* Data wrangling (i.e. cleaning, balancing)
* Converting data to one-hot encoded vectors
* Figuring out how to use numerous new libraries (e.g. PyTorch, sklearn, pandas)

1. *What did you enjoy?*

* Finally seeing the loss going down and accuracy going up after getting the model working for the first time
* Visualizing the difference between various hyperparameter adjustments

1. *What did you find confusing?*

* Instructions on how to make some of the plots
* Understanding what to do for the one-hot encoding
  + Might have been nice to see a simple example to help us intuitively understand what we’re supposed to be doing
  + Why convert to integers first – why can’t we just use the OneHotEncoding class directly?
* Handling various data structures and types between all the libraries (e.g. lists, numpy arrays, PyTorch tensors; float32, float64)

1. *What was helpful?*

* Asking peers questions
* Searching for simple tutorials online