Project Report

Understanding the Dataset

1) There are a total of 11,687 high income earners (>50k) and 37,155 low income earners (<=50k) in the dataset.
2) The dataset is not balanced, as the total number of low income earners is more than triple that of the high income earners. Training on an unbalanced dataset can lead to model problems such as not being able to learn a sufficient amount of characteristics from the minority class and mostly only predicting the majority class. In this case, because there are much more low income earners labels in the dataset, the model can potentially fail to recognize enough characteristics of the high income earners, leading to the predictions being "<=50k" almost all the time.

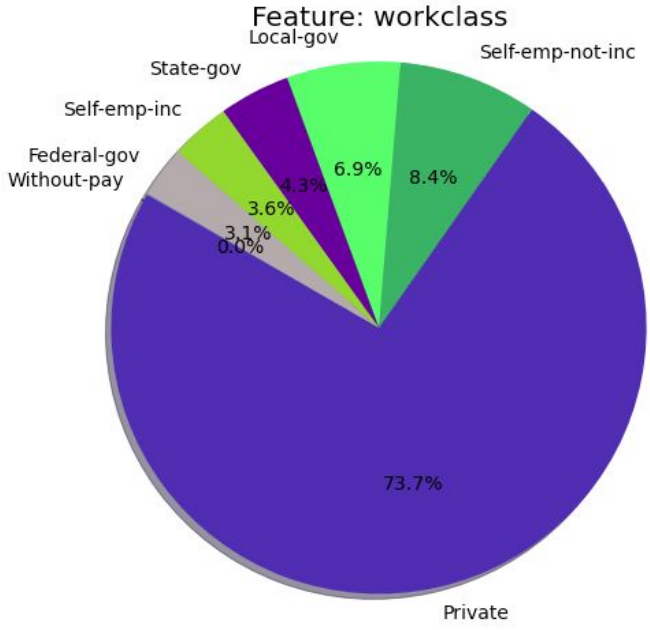
Data Cleaning

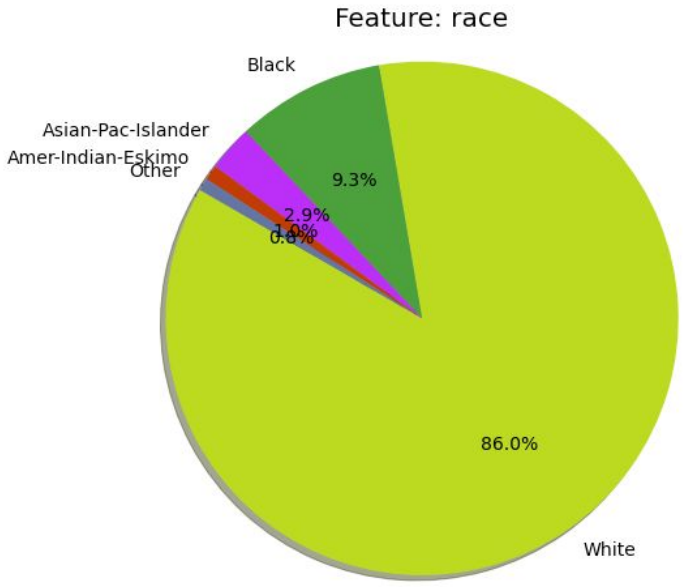
1) A total of 3,620 entries were removed. A total of 45,222 entries are left.
2) This is a reasonable number of samples to throw out, because relative to the size of the dataset, the removed 3,620 entries is less than 0.0001% and can be considered as negligible.

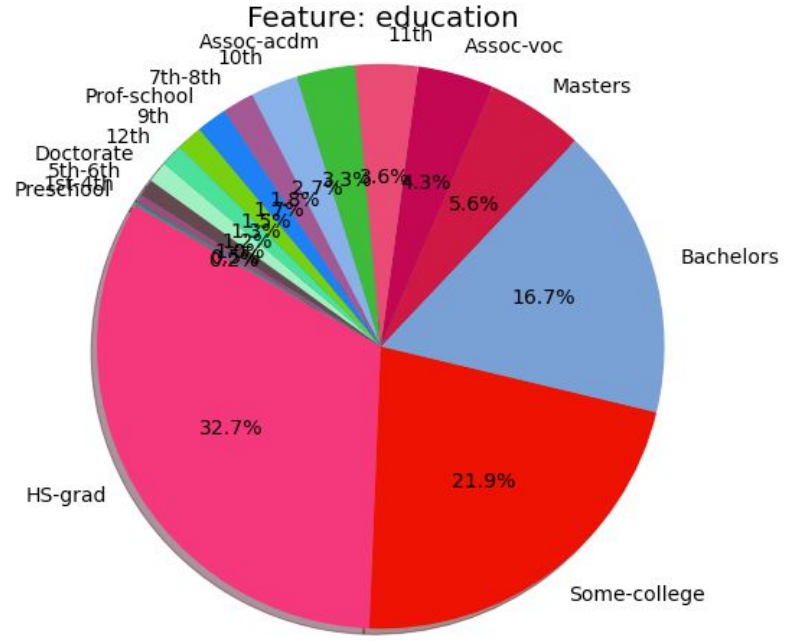
Data Visualization and Understanding

1) The minimum age of individuals is 17 years old; the minimum number of hours worked per week is 1 hour.

Pie charts

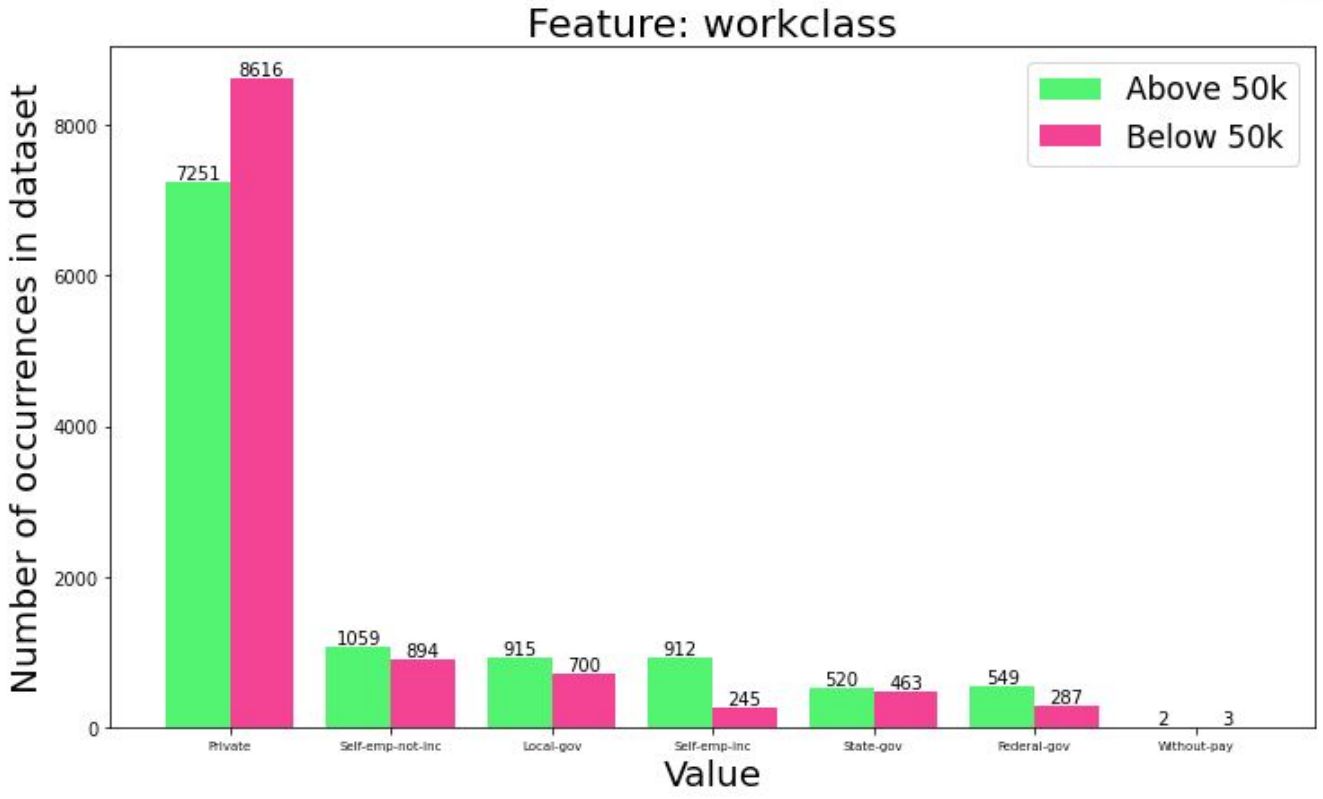


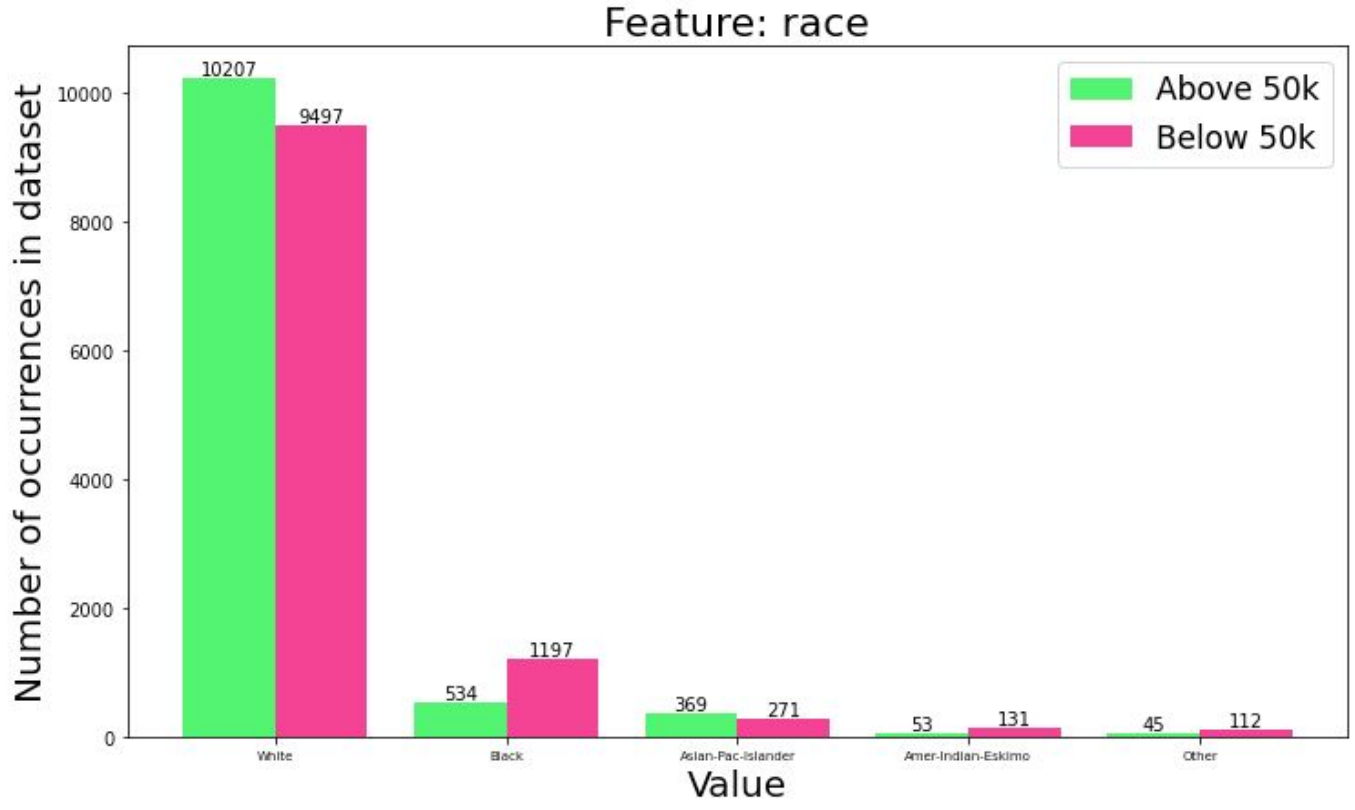


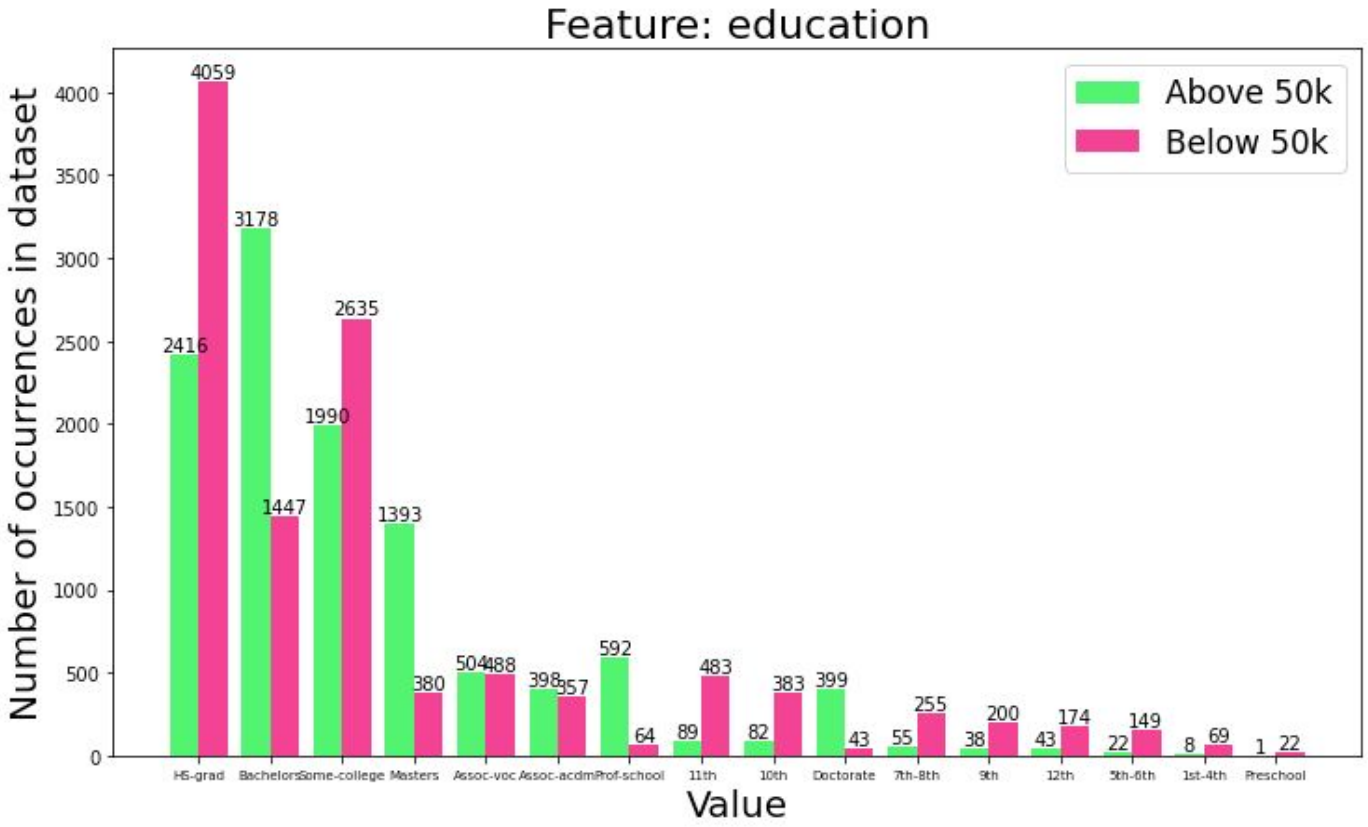


2) The work class and race are over-represented. For example, 73.7% of the work class categorical feature is dominated by private industry and 86.0% of the race feature is dominated by white. On the contrary, federal government jobs only consist of 3.1% of workclass and 1.5% of the education class is 12th grade. Hence, the model will learn much more features from the over-represented classes compared to the under-represented ones, leading to the binary income classification results being focused on for example, white people who work in the private sector.
3) As mentioned in 2), education classes consisting of people who dropped out or do not decide to pursue school before high school graduation are under-represented (about 1-4% for every class). Moreover, federal government and self-employed workers are also under-represented in the work class feature.

Binary bar charts







4) The top 3 features are: White race, Private work class, and HS-grad education.
5) HS-grads have about a 35% chance of earning more than 50k annually (below 50% and not much likely). If given the education is Bachelors and nothing else, the best assumption is that they earn over 50k because according to the dataset, the people with bachelor degrees who earn over 50k are more than double compared to the ones with bachelor degrees who earn below.

Data Pre-Processing

1) The major disadvantage of using an integer representation for categorical data is that some categorical features may be given more numerical importance than others due to the arbitrary assignment of large integers. In reality, however, they should all be considered of the same importance.
2) The major disadvantage of using un-normalized continuous data is that some features will have higher values than some others due to the nature of these features. For example, age is a continuous data feature that usually ranges from 1 to 100; annual salary is another continuous data feature, but it can range from 10k to 100k or even more, thus having a bigger impact on the weights and would result in model learning more from this feature. This shows that in order for the model to learn from all the features equally, these continuous data must be normalized to account for the gap in their numerical values.

DataLoader

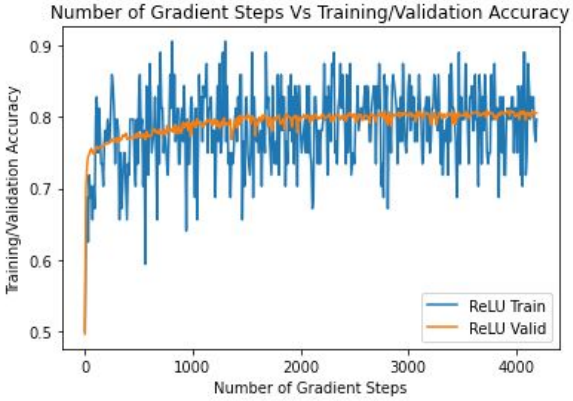
1) The importance of shuffling the data during training is that it makes the batch become more generalized and is more representative of the entire dataset. Failure to do so could result in batches being very similar to one another, especially during training when the gradients are calculated for every batch, updating the weights and bias. As a result, the model is likely to be updated the same one for 1 batch compared to the previous batch, leading to a less generalized result.

Model

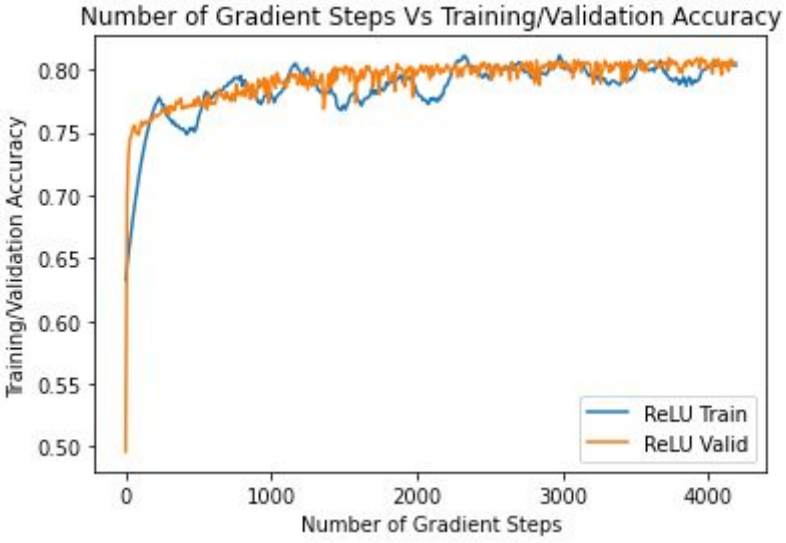
1) The size of the first layer is chosen to be 103 by 64. This is because the input data has 103 columns, therefore the input layer of the MLP needs to have 103 rows to match the input data. 64 is chosen to be the total number of neurons in the first year because usually a neuron number that is ⅔th of the input layer plus the output layer is selected. The output of the second layer is 1 because this is a binary classification problem, and the model only needs to output either "0" or "1", indicating whether or not the income is above 50K.
2) The output of the neural network is represented as a probability between 0 and 1 because there are multiple layers involved in the network, where every layer contains a level of uncertainty that is carried forward by the next layer in the network. Hence, the final output can be considered as an output for a probability function that takes the input features as its input. And because probabilities are between 0 and 1, the output must also map to a range between 0 and 1. In the case of this problem, an output of 0 indicates that there is absolutely 0% chance of this particular person earning over 50K, and an output of 1 indicates the person has a 100% chance of earning over 50K.

Validation

1) *Batch size*: *64* *(N = 10)*
   MLP hidden size: 64
   Learning rate: 0.5
   Epochs: 15
   Random seed: 0
   Validation Accuracy: 0.806



2) **Smoothed graphs** *(using savgol filter)*
Batch size: 64 *(N = 10)*
MLP hidden size: 64
Learning rate: 0.5
Epochs: 15
Random seed: 0
Validation Accuracy: 0.806



Hyperparameters Exploration

Learning Rate

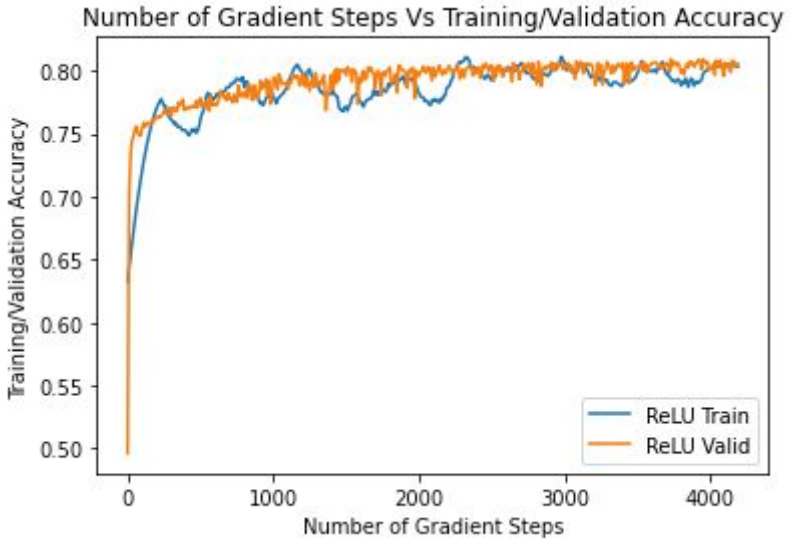
1. *Epochs: 15*
Batch size: 64 (eval_every = 10)
Layer number: 2 (Linear Relu followed by Linear output)
Hidden layer size: 64
Activation function for 1st layer: ReLU
Random seed: 0

Learning rate	Highest Validation Accuracy
0.001	0.745
0.01	0.773
0.1	0.811
1	0.810

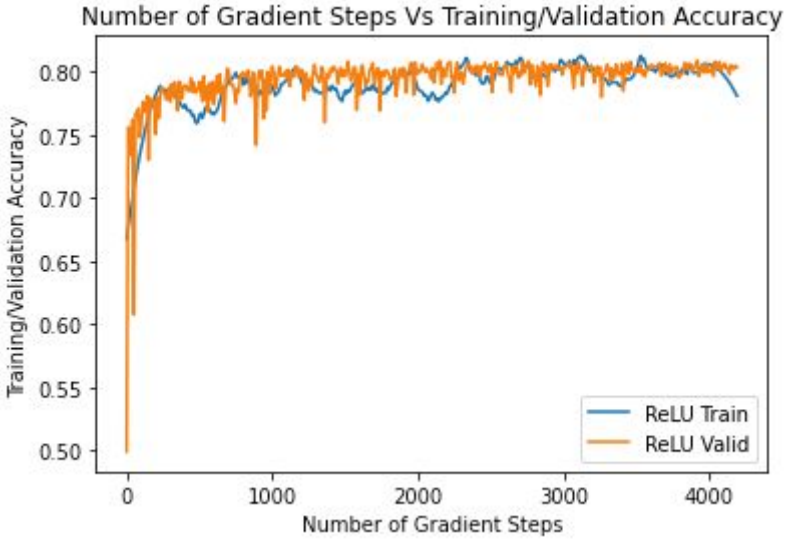
10	0.501
100	0.501
1000	0.501

2. For the 3 graphs below, *eval_every = 10, filter = savgol (for training data only)*

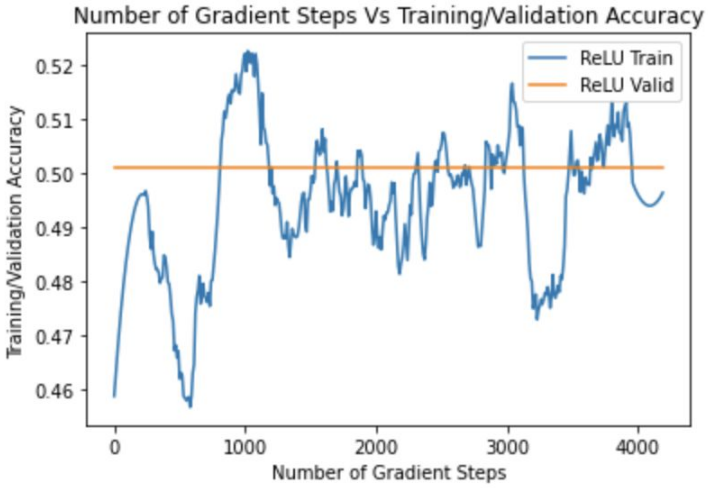
a) **lr = 0.1**



b) **lr = 1**



c) **lr = 100**



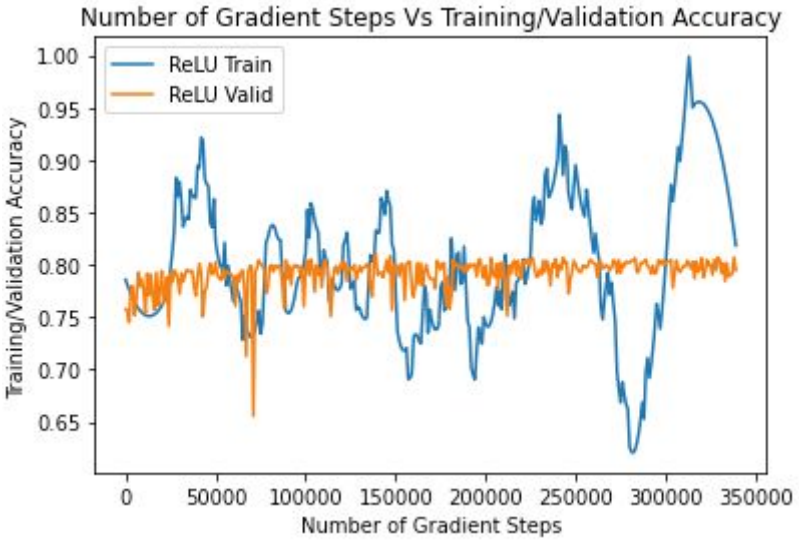
- A learning rate of 0.1 works the best as it yields the best, highest validation accuracy (0.811).

- When the learning rate is too low (i.e, 0.001), the validation accuracy is relatively low (<0.75 in this case) as the loss function is struggling to reach its local/global minimum (moving in the right direction but not fast enough); on the other hand, if the learning rate is too high (i.e, 10, 100, 1000), the loss function will 'bounce' back and forth about the local/global minimum, leading to a stable, low validation accuracy (0.501 in this case).

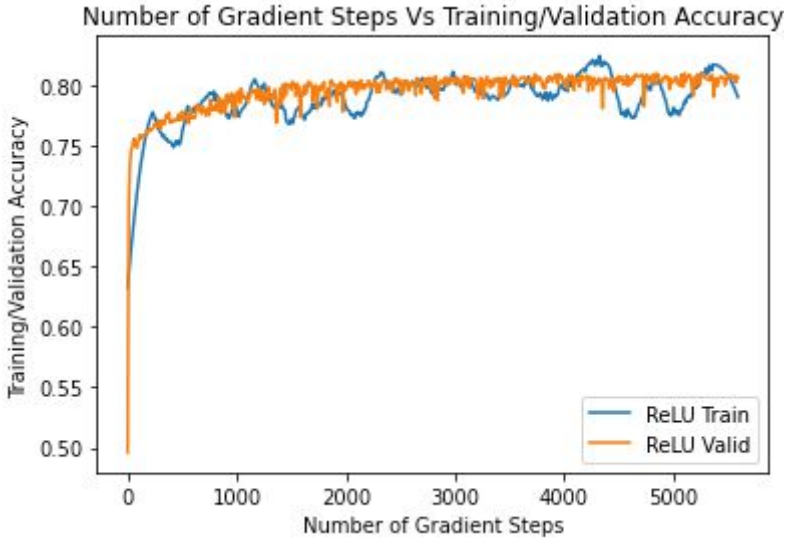
Batch size

1) Number of gradient steps Vs Accuracy plots for 3 different batch sizes (filter applied for training data only)

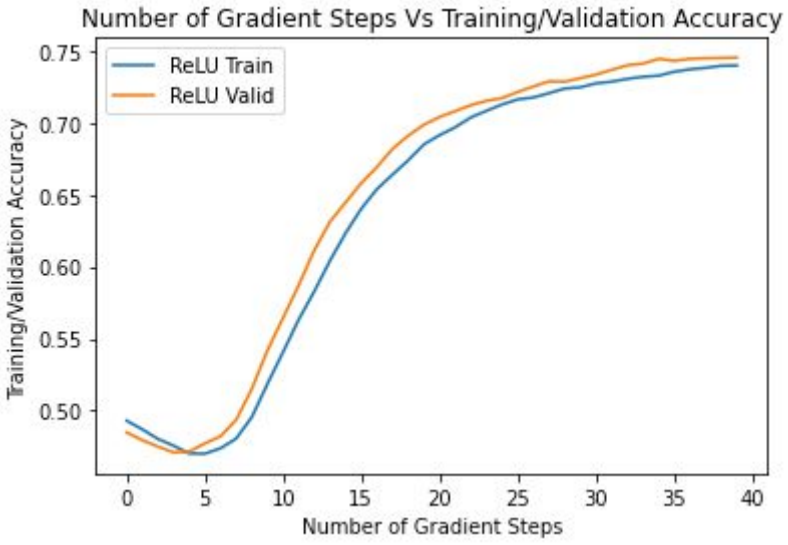
a) Batch size = 1
eval_every = 1000
Epochs = 20



b) Batch size = 64
eval_every = 10
Epochs = 20

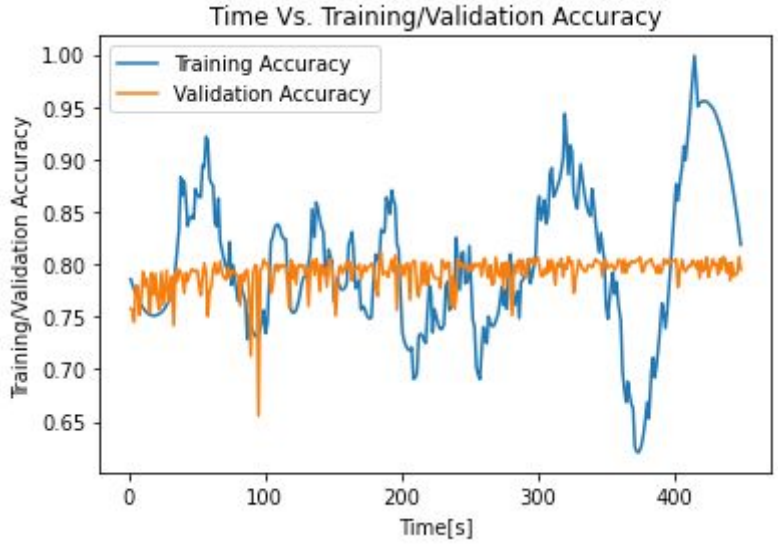


c) Batch size = 17932
eval_every = 1
Epochs = 40
(No filter applied)

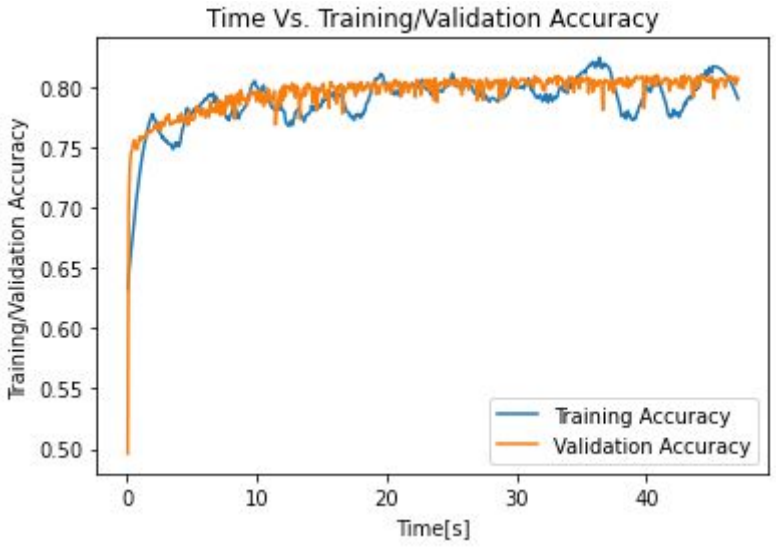


2) Time Vs Accuracy plots for 3 different batch sizes

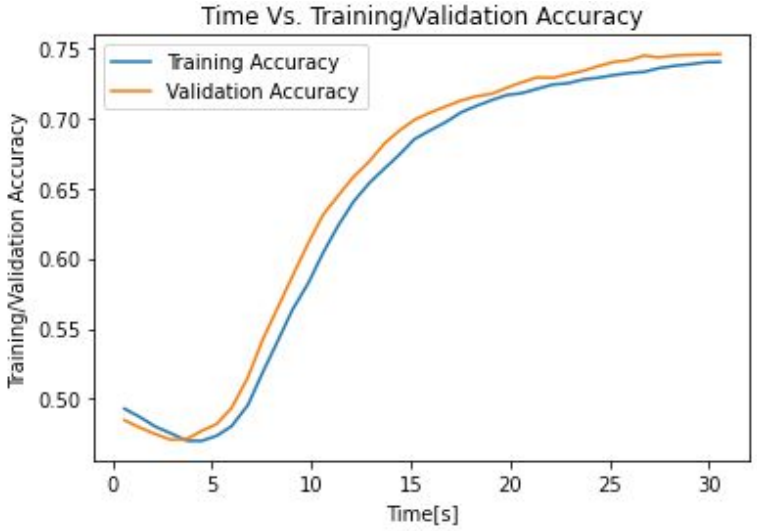
a) Batch size = 1
eval_every = 1000
Epochs = 20



a) Batch size = 64
eval_every = 10
Epochs = 20



a) Batch size = 17932
eval_every = 1
Epochs = 40
(no filter applied)



- A batch size of 64 gives the highest validation accuracy (0.813).

- A batch size of 64 is the fastest to reach a high validation accuracy in terms of number of gradient steps, and to reach its maximum accuracy in terms of time.

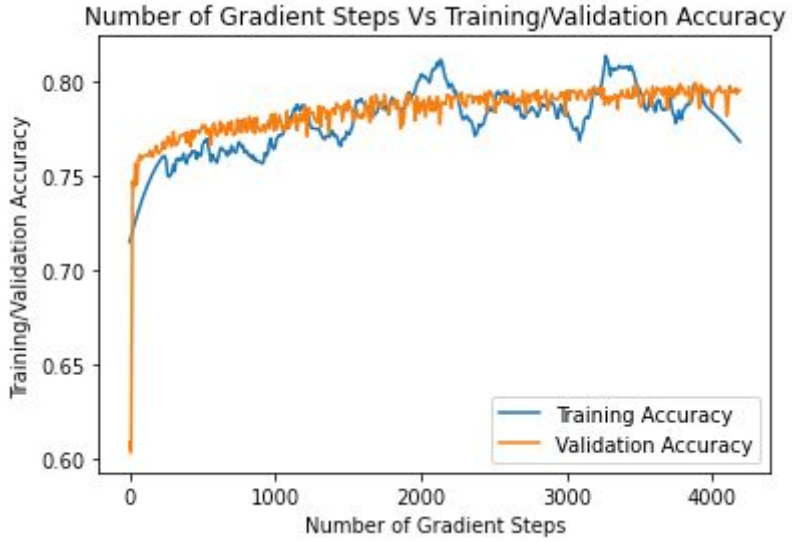
- When the batch size is too low, the gradient descent becomes noisier (as can be observed in the 2 graphs in part 1 and 2) and the model will constantly suffer from sample bias due to the overfitting of the mini-batch distribution instead of the entire

dataset distribution. When the batch size is too high, SGD method simply becomes a regular GD method, which can suffer from being stuck at particular local minima and not being able to converge easily, leading to poor training and validation accuracy.

- An advantage of using a smaller batch size is that the model is training on a more randomized dataset, leading to a model that is more "generalized"; a disadvantage is that it could require a tremendous amount of time due to the number of gradient steps increasing. Conversely, an advantage of using a larger batch size is that the computation time for the model will decrease due to the reduction in the number of gradient steps. However, it would require more epochs for the model to converge, and the model can get stuck at specific local minima of the loss function, leading to a poor training/validation accuracy.

Underfitting

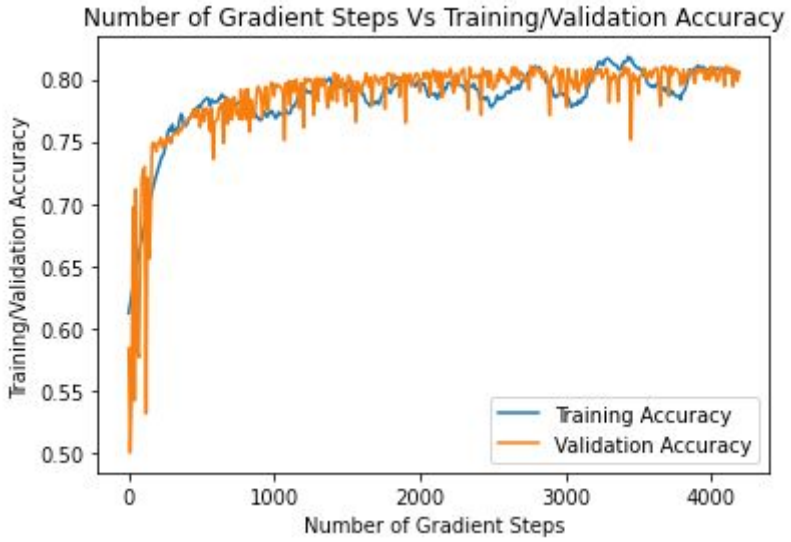
1) *Batch size = 64*
   Epochs = 15
   lr = 0.1
   (filter applied)



2) The model achieved a final validation accuracy of 0.795, slightly worse than the best model so far. The model doesn't quite look like it's underfitting, due to the stable, relatively high training and validation accuracy. However, this can be caused due to the following factors: a) the dataset size is large therefore training is more accurate; b) hyperparameters such as learning rate, batch size, activation function, and so on are all optimized, leading to a more accurate model; c) the problem itself (binary classification) is simple enough that it doesn't require too complex of a model, which diminishes the significance of hidden layer(s).

Overfitting

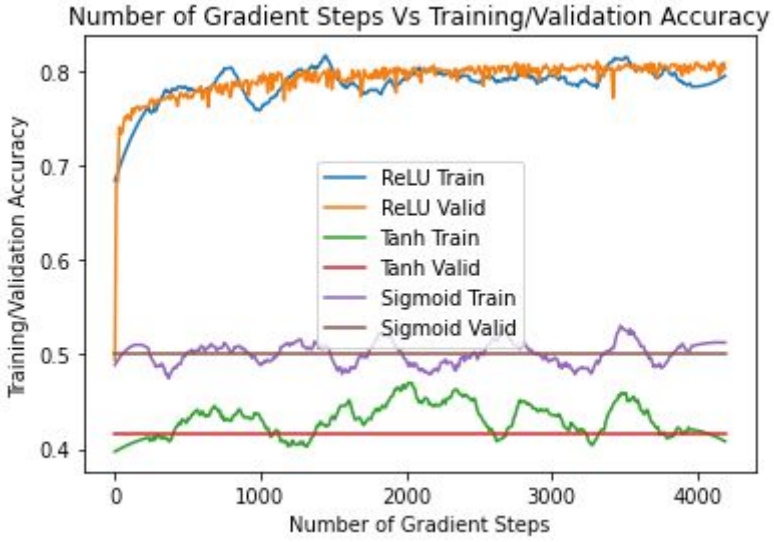
1) *Batch size = 64*
   Epochs = 15
   lr = 0.1
   (filter applied)



2) The model achieved a final validation accuracy of 0.805, which is very similar to the best model. It doesn't look like it's overfitting. However, similar to the reasoning in 5.4, the large dataset and small batch size can overweigh the importance of hidden layers in the neural network, leading to an optimal result.

Activation Function

1) *Batch size = 64*
Epochs = 15
Learning rate = 0.1
(filter applied)



Overall, using the ReLU activation function yields the highest training and validation accuracy. On the other hand, tanh and sigmoid functions both yield similar training and validation accuracies, as can be observed by the overlapping in the plot above.

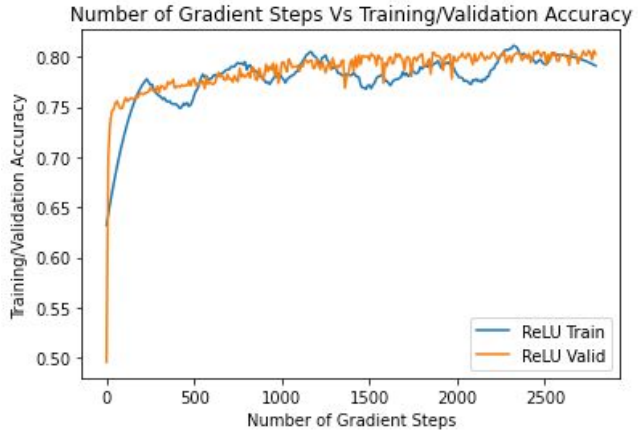
2) *Same hyperparameters as part 1*

Activation Function	Time [s]
ReLU	39.791
Tanh	39.288
Sigmoid	38.716

In terms of time for the 3 different activation functions, they all have similar training times which is around 40 seconds.

Hyperparameters Search

<u>Best model Hyperparamters:</u>
Random seed: 0
Batch size: 64
Epochs: 10
Learning rate: 0.10
Hidden layer: 64
Activation function: ReLU
Model construction: 1 linear ReLU layer followed by a linear output layer, with Sigmoid function as the output function
Eval_every: 10



Highest Training accuracy: 0.906
Final Training accuracy: 0.814

Highest Validation accuracy: 0.806
Final Validation accuracy: 0.803