COMP 551 Mini Project 2

Skylar Laidman (260799869)
Javier Tobar (260868593)
Diana Serra (260663600)

November 23, 2020

1 Abstract

In this project, we implemented a Softmax categorization algorithm with Stochastic Gradient Descent with Momentum. We then investigated how performance changes when the hyper parameters of Learning Rate α, Batch Size γ and Momentum magnitude β were changed. Finally, we compared the performance to a selection of models, including Gaussian Naive Bayes, K-Nearest Neighbours and Decision Trees. It was found that K-Nearest Neighbors resulted in high training and validation accuracies on the Digits dataset, while Naive Bayes was the superior model for the Credits dataset.

2 Introduction

We implemented a Softmax categorization algorithm with Stochastic Gradient Descent (SGD) using Momentum and an early stopping technique for efficiency. The performance of the models were investigated using two datasets, the Digits dataset [1], which involves categorizing images of written numerals, and the credit-g dataset [2] whose objective is to classify an individual as good or bad credit risk.

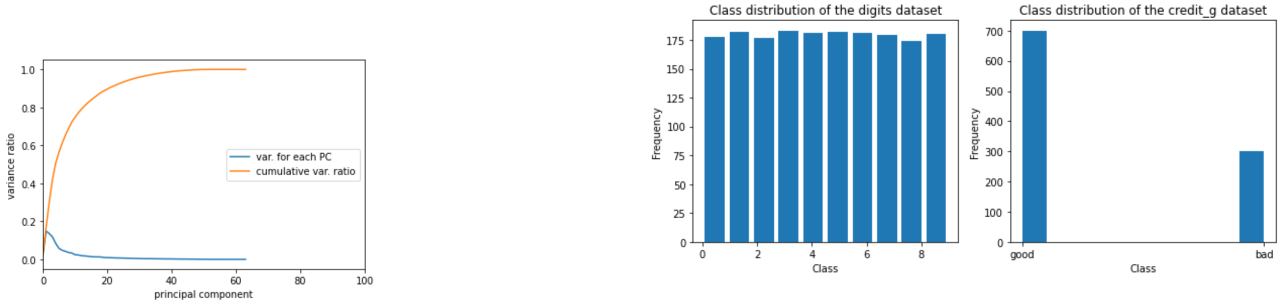
We then investigated how performance changes when the hyperparameters of Learning Rate α, Batch Size γ and Momentum magnitude β were changed. We found that the best hyperparamenters were $\alpha = 0.0001$, $\gamma = 1$ and $\beta = 0.90$ for the Digits dataset, and $\alpha = 0.001$, $\gamma = 1$ and $\beta = 0.95$ for the credit-g dataset. For both datasets, this was done with early stopping. We observed that small and large learning rates do not positively affect the accuracy or speed of training, higher batch size results in faster training time, but lower accuracy, and good momentum values result in a high accuracy and lower training time 4.

Additionally, we compared the performance to a selection of models, including Gaussian Naive Bayes, K-Nearest Neighbours and Decision Trees. It was found that K nearest neighbours provided incredible training and validation accuracy for the digits dataset, while Naive Bayes was better for the credit dataset. However, all the tested classifiers hade much lower accuracies for the credit dataset. Finally, we compared the performance of the SGD early stopping condition used for the previous analyses to that of a generic SGD algorithm with a large number of maximum iterations. Through a small experiment, we found that early stopping leads not only to a much faster run time but also better training and validation accuracies.

3 Datasets

For the Digits dataset, we normalized the data, and used PCA decomposition to compress the dimensions of the data from 64 features to 21 features, with 90% explained variance as shown in Figure 1a. The digits dataset had an almost uniform distribution across all classes (Figure 1b).

For our second dataset, we decided to pick the credit-g dataset which is for binary classification. The objective is to classify an individual as good or bad credit risk by looking at 20 features. It is worth noting that the class distribution was not equally distributed: 300 bad credit and 700 good credit. The data comes from real world data and consequently, more people tend to get approved for credit than denied. Some features are qualitative, for example, the housing feature can be answered by 'rent', 'own' or 'for free'. Other features are numerical, such as the number of existing credits at the bank or the age of the individual. The data was normalized.



(a) The variance of the Digits Dataset

(b) Class Distribution over the two datasets. The Digits dataset contains 10 classes, the Credit dataset contains 2.

Figure 1: Dataset visualization

4 Results

4.1 Logistic Performance

By plotting the duration of each run in seconds against each of the parameters chosen for both datasets (Appendix B Figure 4 and 5), we were able to analyze the performance of our multi-class logistic regression implementation in relation to the optimization hyperparameters. Firstly, the digits dataset was mostly unaffected by the learning rate, with a steep drop off only at the end. Conversely, the Credit dataset had a steep incline in run time until the learning accuracy hits -3. Secondly, batch size improves the performance of both datasets, affecting the digits dataset more severely as evident by the size of the drop. This is to be expected, as higher batch size means less mini batches, and less iterations of Stochastic Gradient Descent. Lastly, increase in momentum for both datasets results in an overall increase in run time, with a slight downward peak at the 0.90 point; however it should be noted that the overall increase is extremely small for both cases.

Additionally, Figures 4 and 5 show the relationship between each hyperparameter and the training and validation accuracies of each dataset. Evidently, both datasets behaved very differently under each hyper parameter; with the most noticeable differences being that the accuracy of the Credit dataset was significantly lower, and that larger batch sizes seemed to have little impact on accuracies in the Credit dataset, while causing accuracies to drop in the Digits dataset. However, we can also draw some similarities; for learning rate, we can see that both datasets had similar curves, with clear peaks in their training accuracy and multiple peaks on the validation accuracy. In addition, in both momentum graphs the training accuracy seems to stay quite uniform while the validation accuracy exhibits strong, definite peaks.

4.2 Comparison of accuracy

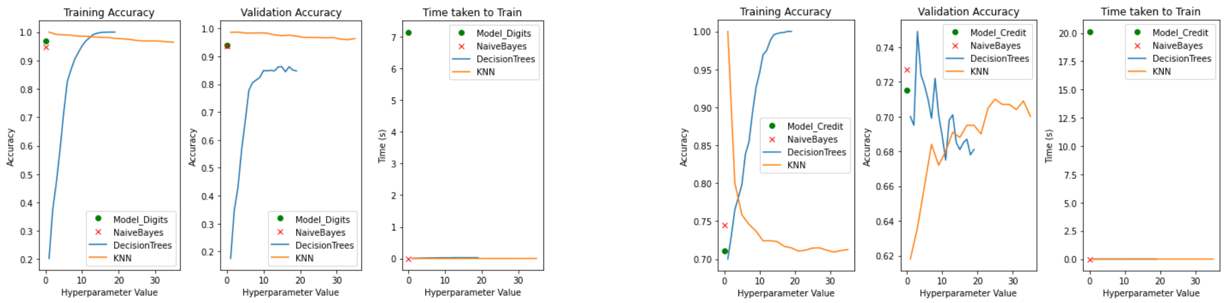
When tuning our hyperparameters for KNN and decision trees (DT), we decided to limit DT's hyperparameter to 25 because there was a significant drop in overall performance when picking a high value hyperparameter. However, KNN's performance was inconclusive around 25, so we decided to have a maximum of 35 neighbors which allowed us to safely determine that our KNN was no longer being optimized.

For the Digits dataset, Naive Bayes obtained a training accuracy of 0.948 and a validation accuracy of 0.939, whereas Model-Digits obtained 0.970 and 0.942 respectively. We note that KNN excelled for this dataset and was able to achieve 0.99 training and validation accuracy. On the other hand, decision trees behaved as expected: high variance.

For the Credit-g dataset, Naive Bayes had a training accuracy of 0.747 and a validation accuracy of 0.730, whereas Model-Credit obtained 0.711 and 0.715 respectively. Although Model-Credit's performance dropped, the other models also suffered a decrease in their accuracy. We can observe KNN's typical behaviour when overfitted: perfect training accuracy with 1 neighbor and high variance. Depending on the partition of our training and testing data, KNN's ideal parameter changed drastically, from 5 to 25.

4.3 Optimization Termination Condition

Due to the large run time of a generic implementation of the SGD optimizer, we decided to implement an early stopping termination condition. Namely, we decided to terminate the algorithm once the validation error had not been decreasing for 20 iterations. To visualize the effectiveness of this condition, we ran our implementation of SGD against one that instead placed a limitation on the maximum number of iterations, which was set to a high number. As suspected, we found that the early stopping SGD had higher training and validation accuracy than its counterpart, in addition to being much faster. [see Appendix A]



(a) Overview of ML models with Digits dataset

(b) Overview of ML models with Credit-g dataset

Figure 2: Comparison of ML models across 2 different datasets

5 Discussion and Conclusion

Our exploration of different optimization hyper parameters and their effects on the training and validation accuracies of our two dataset yielded two conclusions. First, that there were no perfect hyperparameter values; each choice we made had to be a tradeoff between accuracy and speed as more often than not the values that led to the fastest model would lead to low accuracies. Second, that hyperparameters can have significantly different effects on different datasets; as seen by the batch size curves in Appendix B.

Additionally, through our comparison of different classifiers, we concluded the K-Nearest Neighbors was the superior choice for the digits dataset, reaching a training and validation accuracy of 0.99 , while all classifiers tended to have lower accuracies on the Credit dataset. It was concluded that this could be due to the qualitative features of the dataset. We also noted that decision trees tend to have high variance, telling us it might have been more interesting to include a random forest in our comparison models.

For a future project, it would be interesting to investigate the effects on the training and validation accuracies of our models across various T values in early stopping. In this project we only considered a couple of values based on run time rather than accuracy.

6 Contributions

S. Laidman wrote the the SoftMax regression and the GridSearch functions, as well as cowriting the report.

J. Tobar picked and processed the second dataset, as well as code review and cowriting the report.

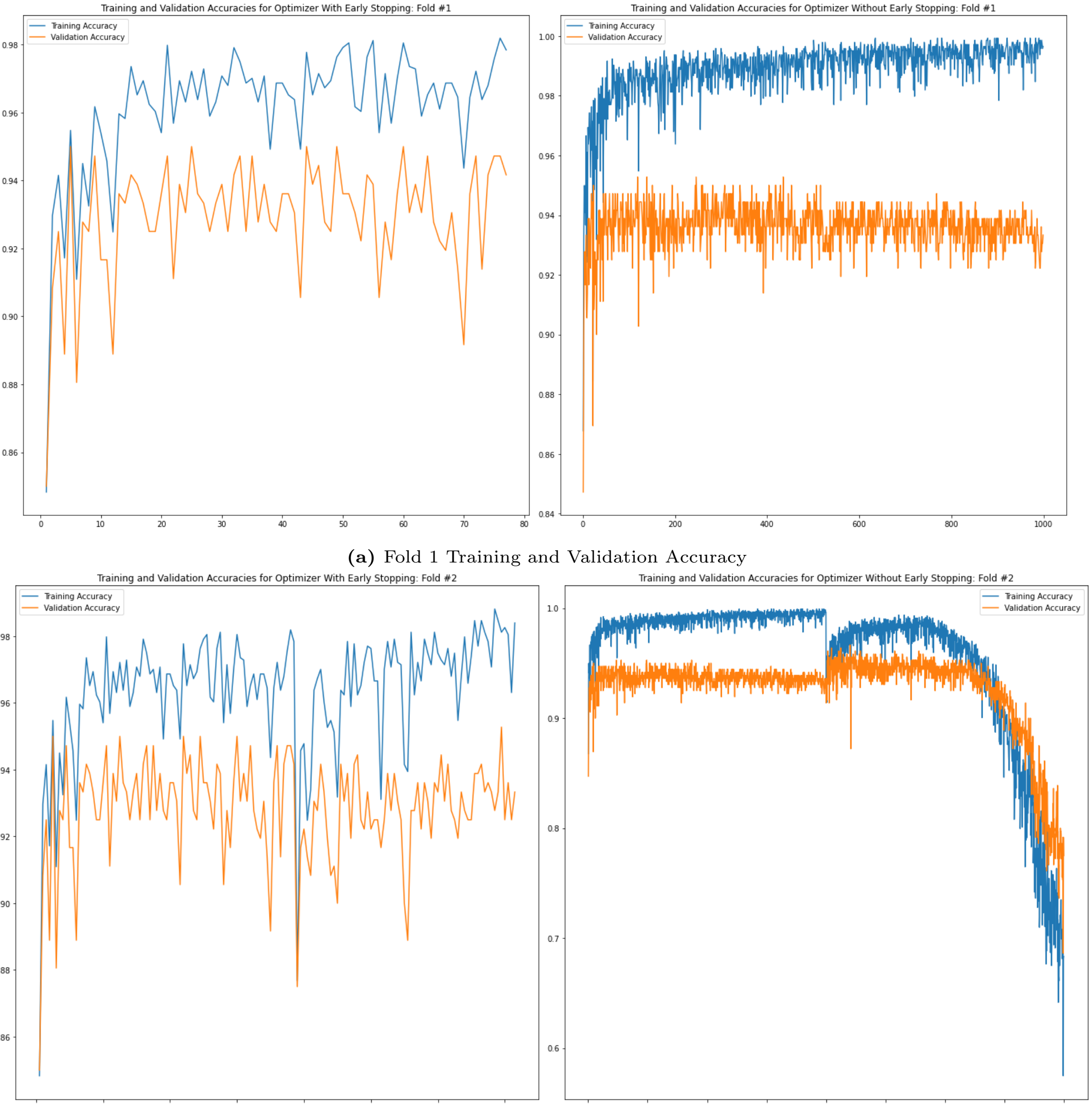
D. Serra wrote the Stochastic Gradient Descent Function and Analysis sections of the code and cowrote the report.

References

[1] C. Kaynak E. Alpaydin. Optical recognition of handwritten digits data set.

[2] Dr. Hans Hofmann. Uci german credit data.

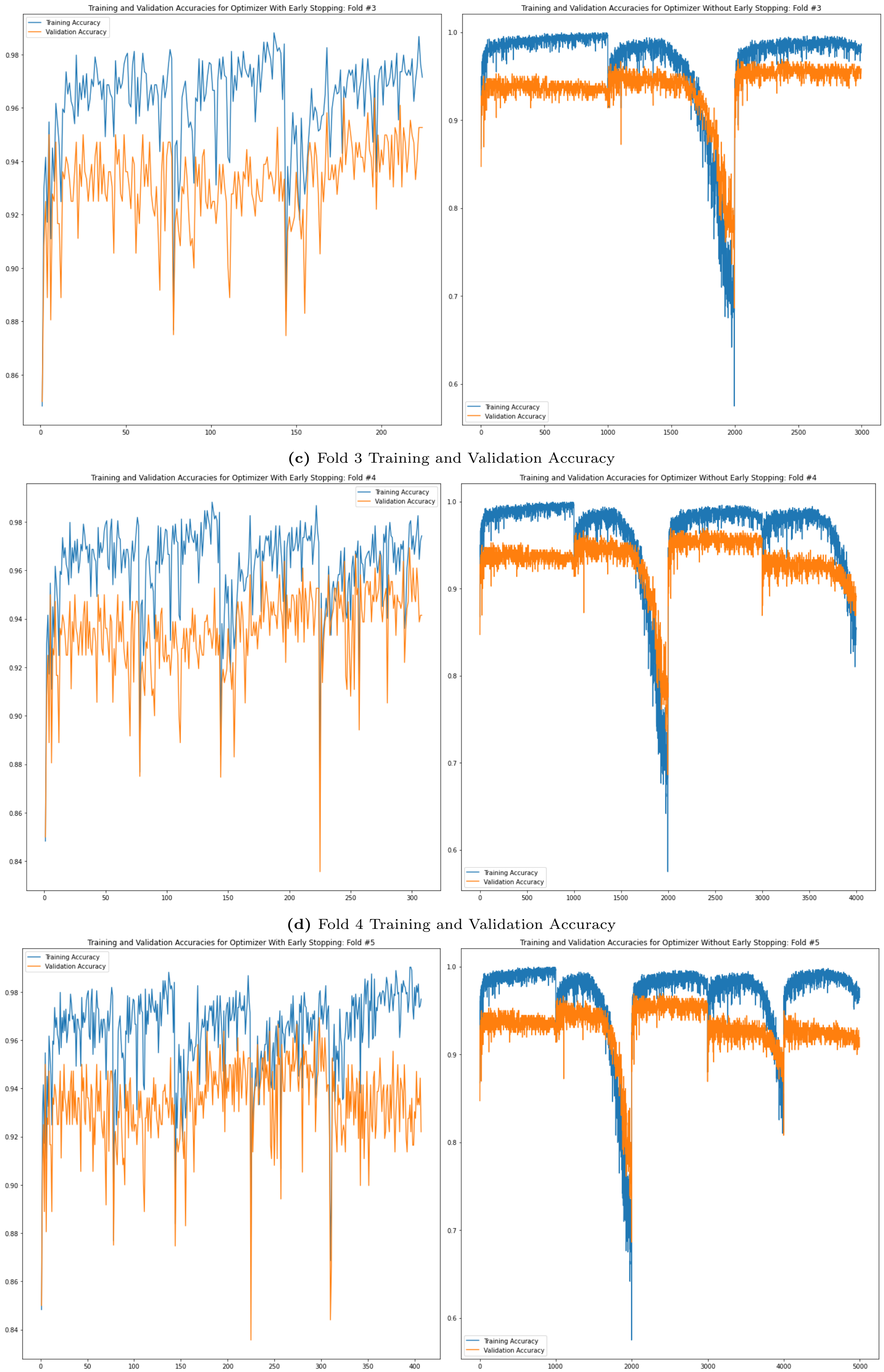
7 Appendix A: Visualization of Termination Condition Analysis



(a) Fold 1 Training and Validation Accuracy

(b) Fold 2 Training and Validation Accuracy

Figure 3: Comparison of Early Stopping vs. Long Running Stochastic Gradient Descent on each fold of Cross Validation



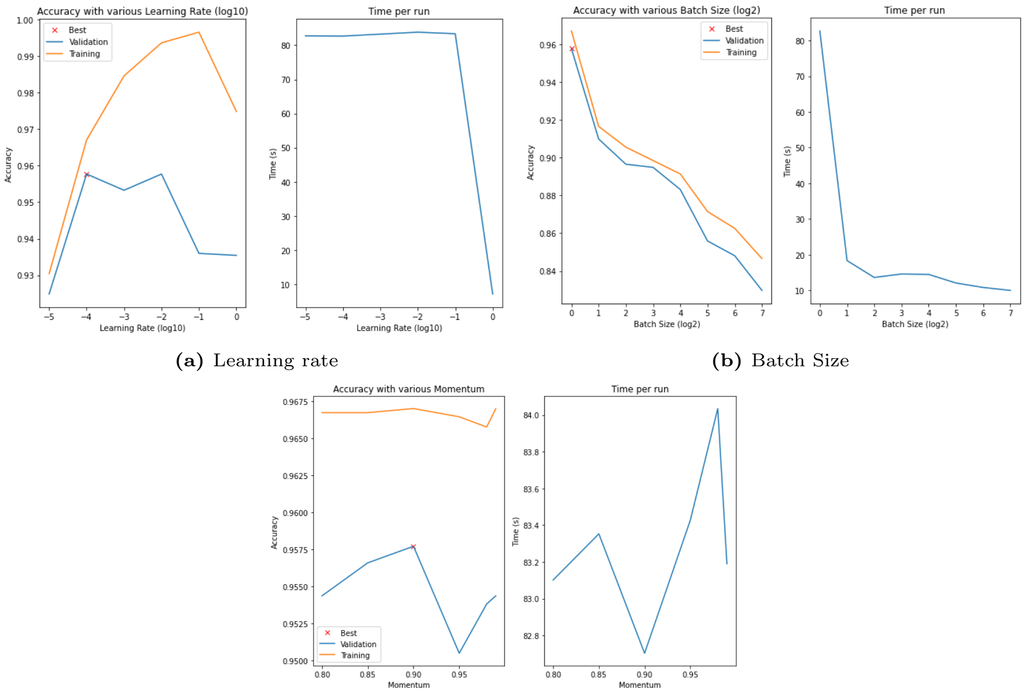
(c) Fold 3 Training and Validation Accuracy

(d) Fold 4 Training and Validation Accuracy

(e) Fold 5 Training and Validation Accuracy

Figure 3: Comparison of Early Stopping vs. Long Running Stochastic Gradient Descent on each fold of Cross Validation (continued)

8 Appendix B: Visualization of Logistic Performance

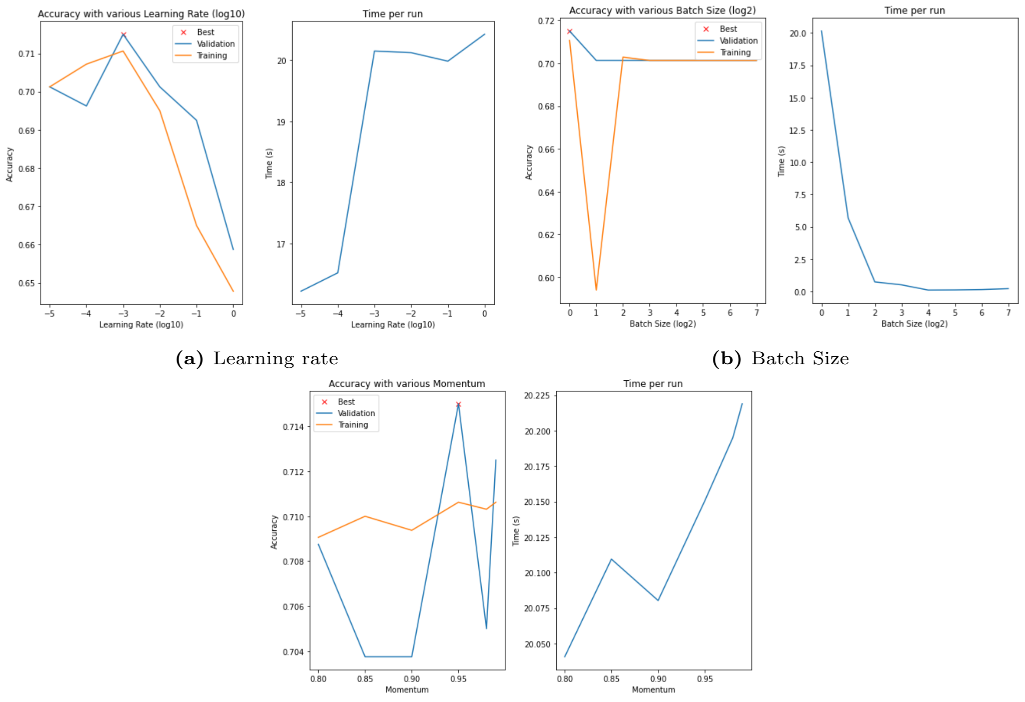

Accuracy with various Learning Rate (log10)
Time per run
Accuracy with various Batch Size (log2)
Time per run
Accuracy with various Momentum
Time per run

(a) Learning rate

(b) Batch Size

(c) Momentum

Figure 4: Various hyperparameters on the digit dataset, trained without early stopping. The optimal values are Learning Rate = 0.0001, Batch Size = 1, Momentum = 0.9


Accuracy with various Learning Rate (log10)
Time per run
Accuracy with various Batch Size (log2)
Time per run
Accuracy with various Momentum
Time per run

(a) Learning rate

(b) Batch Size

(c) Momentum

Figure 5: Various hyperparameters on the Credit dataset, trained with early stopping. The optimal values are Learning Rate = 0.001, Batch Size = 1, Momentum = 0.95