COMP551 Project 2

Zeying Tian, Kevin Li, Ao Shen

October 27, 2021

Abstract

In the first part of the study, we explore the effects of various modifications, including mini-batch and momentum, to the Logistic Regression model using standard gradient descent, on the diabetes dataset. The baseline full-batch model converges to 76.5% test accuracy at 840,000 epochs with $2 \cdot 10^{-4}$ learning rate. Adding mini-batch induces intense oscillations, and models do not converge, though it may rise to higher accuracy sooner than the baseline. Momentum has no significant impacts in terms of performance, yet it does improve the convergence speed. We have noticed that momentum can be effective in reducing oscillations introduced by mini-batch.

In the second part, we apply Logistic Regression models to text classification on a dataset to tell apart human-wrote and computer generated news. We have experimented with various preprocessing settings and have reached 78.3% accuracy on the test set.

1 Introduction

Logistic Regression is a widely-adopted linear classifier. In this experiment, we have built a Logistic Regression model with gradient descent, which allows for mini-batch and momentum. One thing to notice is that we use "epoch" instead of "number of iterations." In one epoch, the model goes over the entire training set, yet for mini-batch, the model updates weights each batch. As such, using epochs offers us a consistent view regardless of the choice of batch size. Generally speaking, mini-batch models are less stable as they only use a portion of the training set. One way to counter oscillations brought by mini-batch is via momentum. Momentum allows us to take into account of previous gradients, and thus extreme gradients in mini-batch can be averaged out.

We first select a baseline model with $learning\_rate = 2 \cdot 10^{-4}$, which converges at 840,000 epochs, with a validation accuracy of 77% and 76.5% accuracy on the test set. We first explore how batch size influences model performance. Concretely, we have found that mini-batch models do not converge, though they often rise to higher accuracy sooner than the baseline model. Secondly, we experiment with momentum on our baseline model. momentum does not influence the final performance (All models reached the same validation and test accuracy) but it has increased the speed of convergence from 840,000 to 714,000.

In the second part, we dive into the fake news data set. We have explored various preprocessing settings, including stemming, term-frequency vectorization, etc. Overall, the preprocessing setting with binary unigram-bigram *CountVectorizer* works the best, achieving 78.3% in the test set.

2 Datasets

We have two datasets in this experiment, the first one to predict diabetes, and the second one to distinguish fake news. For both datasets, we do not have missing data.

The diabetes dataset has 8 features, including one's blood pressure, BMI, skin thickness, and so on. Our task is to predict whether one has diabetes. 600 instance are used for training, 68 instances for testing, and 100 instances for validation. In this study, we are using the dataset as it is, i.e., we do not apply any preprocessing. Distributions of feature can be viewed in the Appendix. According to two studies [1, 2], it is possible that some of the features may be irrelevant, and by removing such features we may achieve a better performance.

The fake news dataset consists of human-wrote and computer-generated news, and we are going to tell which one is real and which is not. There are over 30,000 instances for training, 4000 instances for validation, and 6000 instances for testing. We shall extract features with *CountVectorizer* from *sklearn*. Coupled with utilities from *nltk*, we have built a wide

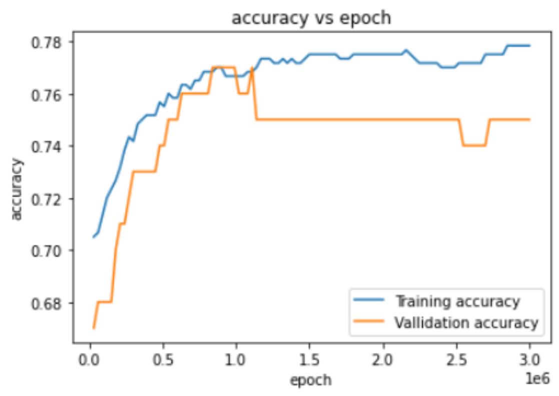


Figure 1: Performance of the Default GD with 2e-4 Learning Rate

range of feature extractors: binary *CountVectorizer*, unigram as well as bigram feature extraction, extractor that converts words to lowercase and one that does not. The complete combination of feature extractors we have tried are listed in Table 6. Moreover, we have experimented with *TfidfTransformer* from *sklearn*.

3 Results

For the first part, we have defined that $epsilon = 6.5 \cdot 10^{-3}$ for our logistic regression model. In this study, we define the point of convergence to be the point with the highest stable validation accuracy. What we mean by "stable" is that the point of convergence has to be able to sustain at that accuracy for a period of time.

3.1 The Baseline Model

We have chosen a learning rate of $2 \cdot 10^{-4}$ for our baseline model. By plotting the model's training and validation accuracy versus epochs, as shown in Figure 1, we find that the model converges at $epochs = 840,000$, with validation accuracy at 77% and test accuracy at 76.5%.

3.2 Impacts of MiniBatch

Secondly, we shall investigate the impacts of *MiniBatch*. Because it takes significantly longer to fit with *MiniBatch*, we have to constrain ourselves with a limited range of batch sizes: 16, 64, 128, 256. The plot of losses versus epochs, as well as that of norm versus epochs, are depicted in Figure 10. We have also recorded the trends of training and validation accuracy for each batch size, which can be viewed in the Appendix. Looking at the plot of validation accuracy, as shown in Figure 11, it is obvious that these models do not converge. Since all *MiniBatch* models do not converge, there is no definite way to measure their performance, yet we have observed that *MiniBatch* models rise to a certain accuracy and start to oscillate around that level after 500,000 epochs. Therefore, to measure their performance and as a way to gauge their degree of oscillations, we have calculated the mean of validation accuracy, as well as the *std* of validation accuracy, after 500,000 epochs, as shown in Table 1. Since all of these models do not converge, we do not apply these models on the test set.

Batch Size	Mean of Val. Accuracy	STD
16	72.69%	0.0416
64	74.10%	0.0326
128	74.66%	0.0214
256	74.42%	0.0222

Table 1: Mean of Validation Accuracy and Standard Error for Various Batch Sizes After 500,000 Epochs

3.3 Impacts of Momentum

In the third section, we want to know how momentum affects our model. We have picked 0.99, 0.96, 0.93, 0.9, 0.6, 0.3 to test for momentum. Similar to what we have done in subsection 3.1, we have measured the number of epochs to converge, as well as the maximal validation accuracy, in Table 2. We have likewise plotted training and validation accuracy versus epochs, which is included in the Appendix. The performance of these models on the test data is shown in Table 4.

Momentum	Convergence Epoch	Val. Accuracy
0.99	714,000	77%
0.96	714,000	77%
0.93	714,000	77%
0.90	714,000	77%
0.60	714,000	77%
0.30	714,000	77%
0	840,000	77%

Table 2: Convergence Speed and Validation Accuracy for Various Momentum

3.4 Impacts of Momentum on Mini-Batch

We have applied momentum to the smallest and largest batch size we have tried, which are 16 and 256 respectively. Again, all 12 models do not converge, and their plots of accuracy versus epochs are listed in the Appendix. Therefore, we have calculated their mean and *std* of validation accuracy, as listed in Table 5. Out of the 12 combinations, $momentum = 0.9$ gains the highest mean validation accuracy for mini-batch size 16, and $momentum = 0.99$ works the best for batch size 256. The former reached 74.61% mean validation accuracy on validation set, and the latter arrived at 75.19%. Since all of these models do not converge, we do not apply these models on the test set.

3.5 Application on Fake News

As the training corpora is huge, we have followed the recommendation from *sklearn* to use "sag" as the solver for the Logistic Regression model. Also, we have set the maximum number of iterations to be 1000. In the first step, we have tried many preprocessing settings without *TfidfTransformer*, and the performance is listed in Table 6. Out of the 16 combinations, we notice that the one with $stemming = False, binary = True, lowercase = False$ works the best in terms of both validation and test accuracy. We then utilized this setting and combined the vectorizer with *Tfidf-Transformer*, the results of which is recorded in Table 7. Also, we have tried the effects of *class_weights* parameter of the Logistic Regression model with that feature extractor; the results is recorded in Table 8.

4 Discussion and Conclusion

From subsection 3.2, it is obvious that mini-batch can oscillates dramatically. All of the models with mini batch do not converge. As shown in *Fig.*12 13 14 15, we can see that the smaller the batch size, the more intense the variation in validation accuracy. Also, all of the mini-batch models can reach validation accuracy higher than 77% before 840,000 epochs, as shown in Table 3. Oscillations are not surprising as models with mini-batch use merely a portion of the training set to update weights each batch, and the data used may be skewed, resulting in extreme gradients. Therefore, it is anticipated that the larger the batch size, the smoother the accuracy and loss curve. For the same reason, it is normal for mini-batch models to reach higher validation accuracy due to its chaotic nature.

Overall, we believe it is impossible to pick one configuration that works the best, because none of them converges.

For the third part, as depicted in *Fig.*16 17 18 19 20 21, along with Table 2 and Table 4, momentum does not influence the final performance, yet it successfully improve the convergence speed from 840,000 to 714,000. One way to understand this is that by including gradients from the past, the model may be descending on a smoother path, and thus reduce the time to converge.

In the last section for part 1, we have seen the effects of including momentum to *MiniBatch* models. As shown by graphs in subsection 4.4, we can see that by adding momentum there are fewer oscillations in terms of validation accuracy. Also, by comparing the *std* of validation accuracy in Table 1 and Table 5, we can see that the introduction of momentum can drastically reduce oscillations. As discussed previously, momentum takes into account gradients from previous updates. As such, extreme gradients, which exists in mini-batch, can be averaged.

Again, we cannot conclude which setting works the best, because models with batch size 16 and 256 do not converge. However, we can see that a momentum of 0.3 has successfully improved the mean validation accuracy of batch size 16 by 0.82%, and a momentum of 0.99 has reduced the *std* of validation accuracy for batch size 256 by 73%.

In the second part, as depicted in Table 6, unigram-bigram feature extractors bring a significant improvements to unigram feature extractors. Also, by comparing the effects of stemming, we have found that stemming actually lowers the final performance across all combinations. Also, converting counts into binary values can improve the performance. As we may observe, converting words into lowercases does not any impact when stemming is applied, and it brings negative effects when there is no stemming. Based on the pattern we have found, we believe that $stemming = False, binary = True, lowercase = False$ brings the best feature extractor. When combining this feature extractor with *TfidfTransformer*, we can see that the transformer brings down the performance dramatically, as shown in Table 7. Finally, we tested the effects of weight balancing with the *class_ weight* parameter. Again, as shown in Table 8, a balanced class weight performs worse than our original model. Overall, the best model we have found utilizes the feature extractor setting listed before and has achieved 78.3% accuracy on the test set.

Statement of Contributions

Zeying Tian is responsible for codes.

Ao Shen is responsible for designing experiments and plot graphs.

Kevin Li is responsible for the report write up.

References

[1] R. D. Joshi and C. K. Dhakal, "Predicting Type 2 Diabetes Using Logistic Regression and Machine Learning Approaches," International Journal of Environmental Research and Public Health, vol. 18, no. 14, p. 7346, 2021.

[2] K. Dhandhania, "End-to-end Data Science example: Predicting diabetes with logistic regression," Towards Data Science, 24-May-2018. [Online]. Available: https://towardsdatascience.com/end-to-end-data-science-example-predicting-diabetes-with-logistic-regression-db9bc88b4d16. [Accessed: 27-Oct-2021].

Appendix

4.1 diabetes Feature Distribution

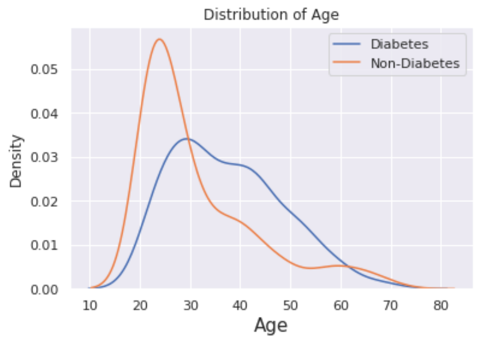


Figure 2: Distribution of Age

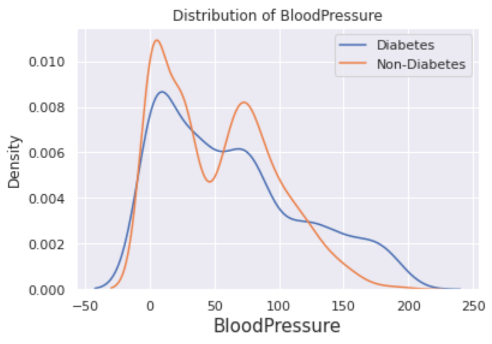


Figure 3: Distribution of blood pressure

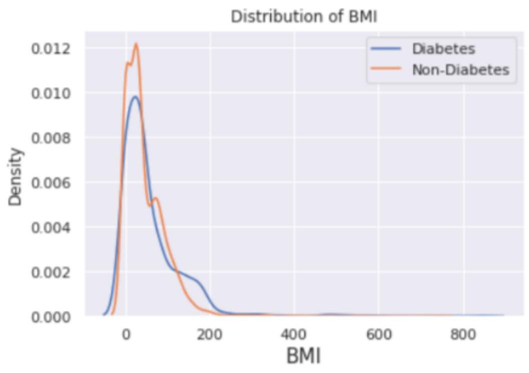


Figure 4: Distribution of blood pressure

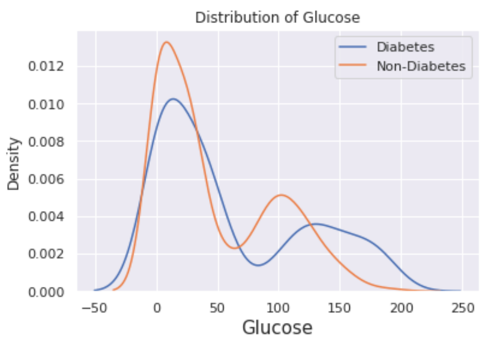


Figure 5: Distribution of glucose

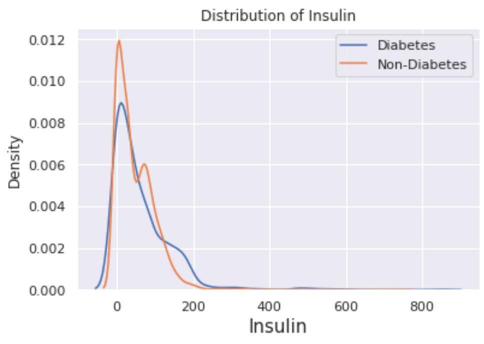


Figure 6: Distribution of Insulin

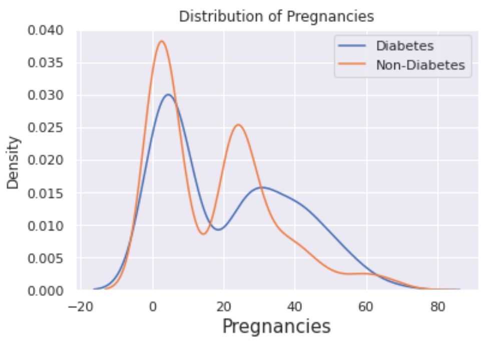


Figure 7: Distribution of pregnancies

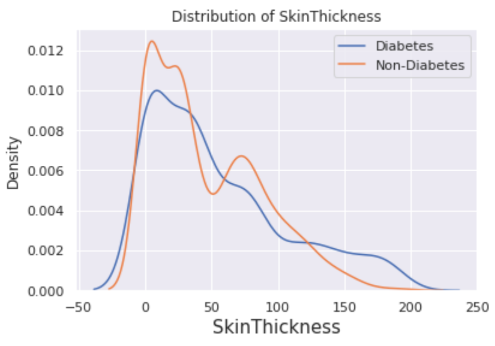


Figure 8: Distribution of skinthickness

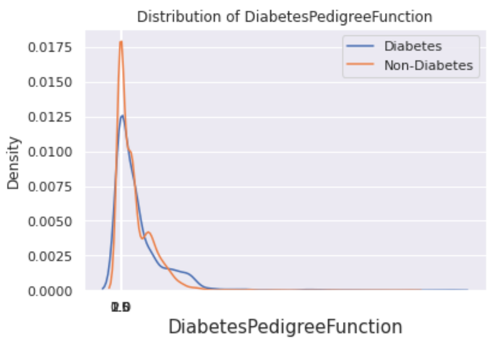
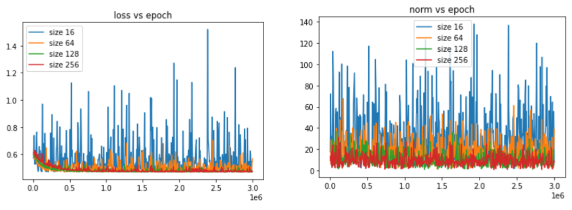


Figure 9: Distribution of diabetefunction

4.2 Impacts of MiniBatch

Batch Size	Epochs
16	114,000
64	120,000
128	120,000
256	216,000

Table 3: The Smallest Epochs for Various MiniBatch Models to Reach Validation Accuracy Greater than 77%



(a) Loss vs. Epochs **(b)** Norm vs. Epochs

Figure 10: Loss and Norm with Various Batch Sizes

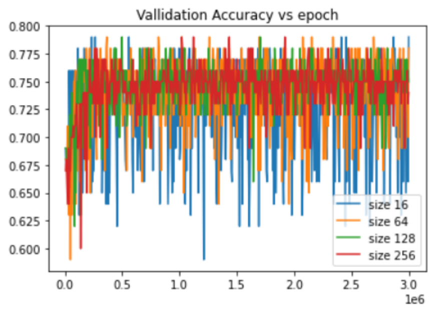


Figure 11: Validation Accuracy for Various Batch Sizes

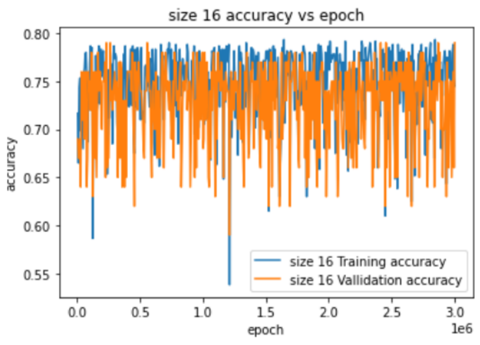


Figure 12: Validation and Training Accuracy for Batch Size 16

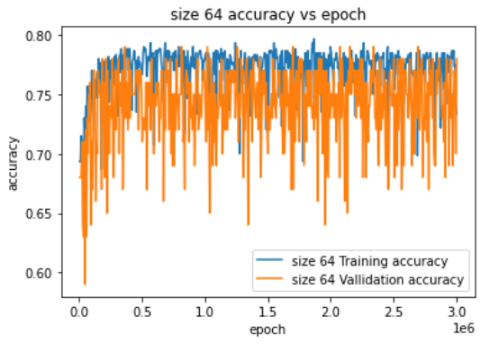


Figure 13: Validation and Training Accuracy for Batch Size 64

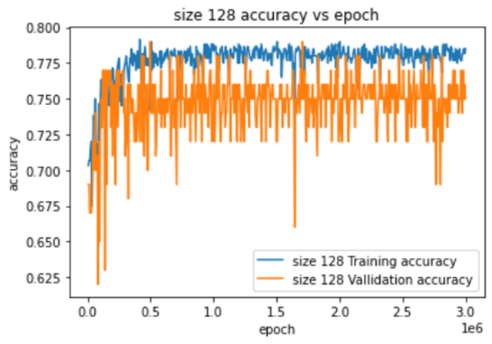


Figure 14: Validation and Training Accuracy for Batch Size 128

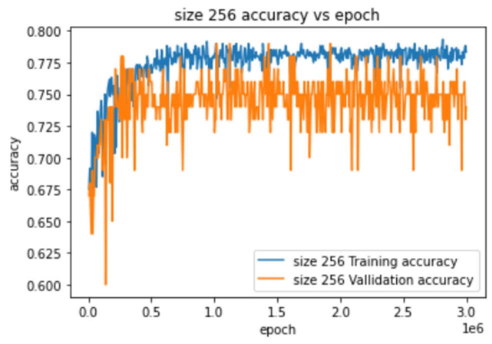


Figure 15: Validation and Training Accuracy for Batch Size 256

4.3 Impacts of Momentum

Momentum	Convergence Epoch	Test Accuracy
0.99	714,000	76.5%
0.96	714,000	76.5%
0.93	714,000	76.5%
0.90	714,000	76.5%
0.60	714,000	76.5%
0.30	714,000	76.5%
0	840,000	77%

Table 4: Convergence Speed and Test Accuracy for Various Momentum

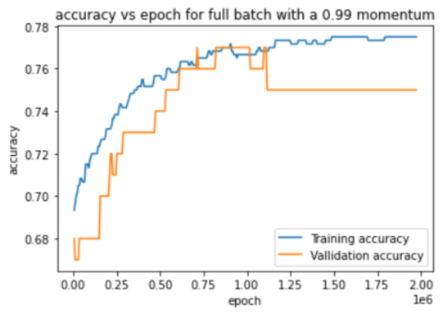


Figure 16: Validation and Training Accuracy for Momentum 0.99

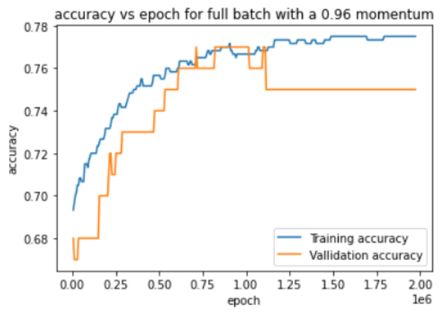


Figure 17: Validation and Training Accuracy for Momentum 0.96

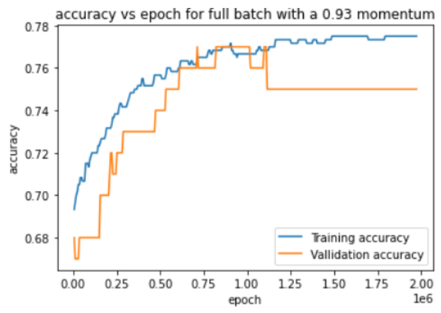


Figure 18: Validation and Training Accuracy for Momentum 0.92

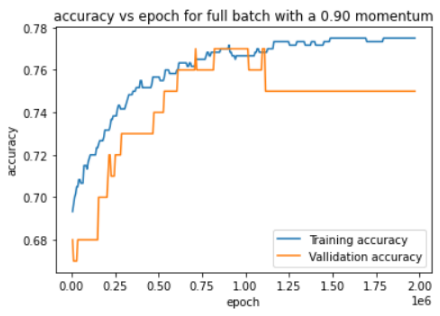


Figure 19: Validation and Training Accuracy for Momentum 0.90

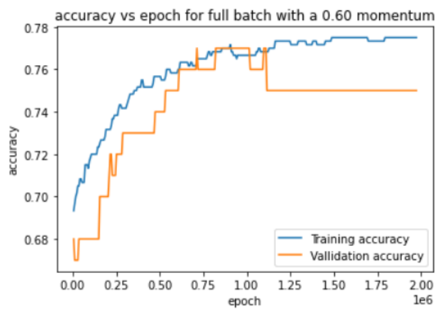


Figure 20: Validation and Training Accuracy for Momentum 0.60

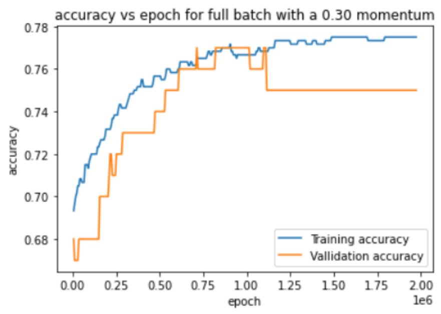


Figure 21: Validation and Training Accuracy for Momentum 0.30

4.4 Impacts of Momentum on Mini-Batch

Batch Size	Momentum	Mean Val. Acc.	STD
16	0.99	74.59%	0.0240
16	0.96	74.46%	0.0245
16	0.93	74.52%	0.0273
16	0.90	74.61%	0.0268
16	0.60	74.22%	0.0315
16	0.30	73.51%	0.0342
256	0.99	75.19%	0.00591
256	0.96	75.09%	0.00589
256	0.93	75.06%	0.00592
256	0.90	75.18%	0.00600
256	0.60	75.07%	0.00599
256	0.30	74.93%	0.0113

Table 5: Mean of Validation Accuracy and Standard Error for Various Momentum with Batch Size 16 and 256 After 500,000 Epochs

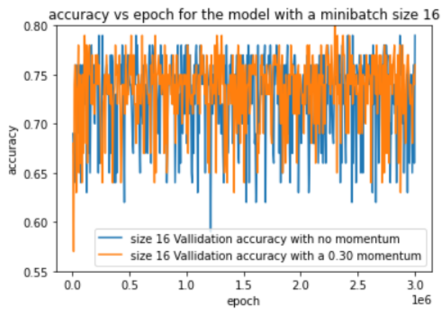


Figure 22: Validation Accuracy for Batch Size 16 with and without Momentum 0.30

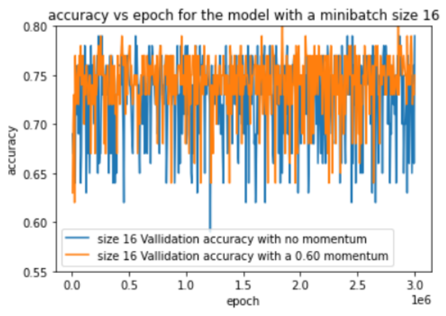


Figure 23: Validation Accuracy for Batch Size 16 with and without Momentum 0.60

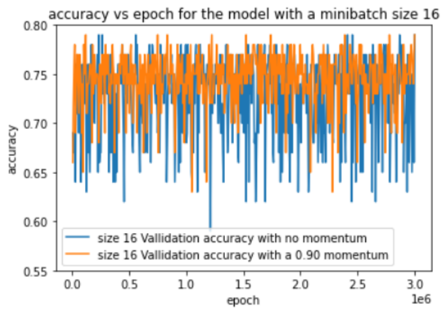


Figure 24: Validation Accuracy for Batch Size 16 with and without Momentum 0.90

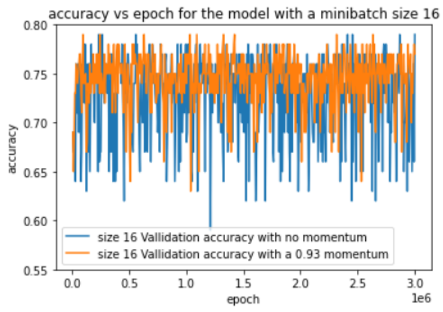


Figure 25: Validation Accuracy for Batch Size 16 with and without Momentum 0.93

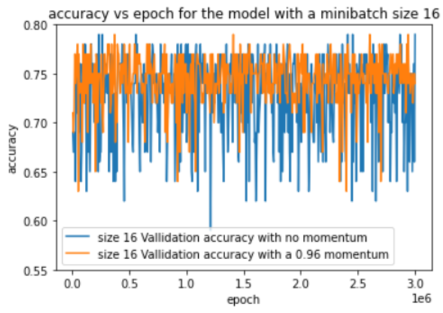


Figure 26: Validation Accuracy for Batch Size 16 with and without Momentum 0.96

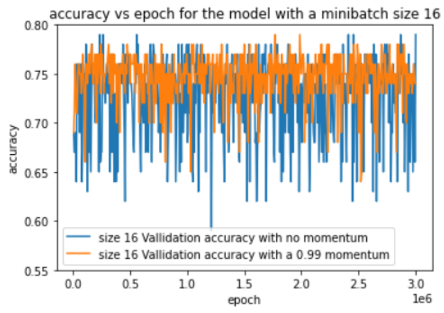


Figure 27: Validation Accuracy for Batch Size 16 with and without Momentum 0.99

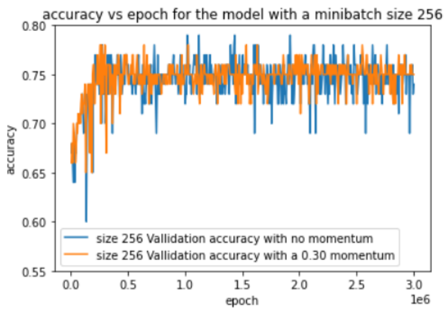


Figure 28: Validation Accuracy for Batch Size 256 with and without Momentum 0.30

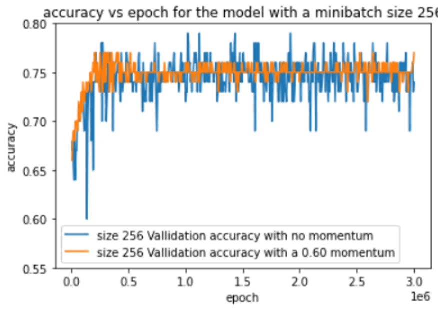


Figure 29: Validation Accuracy for Batch Size 256 with and without Momentum 0.60

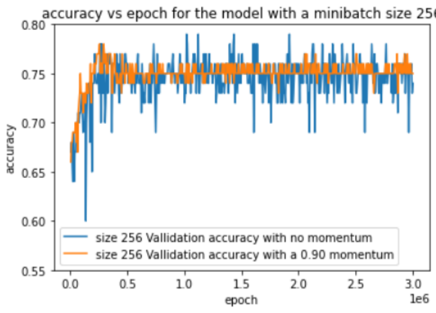


Figure 30: Validation Accuracy for Batch Size 256 with and without Momentum 0.90

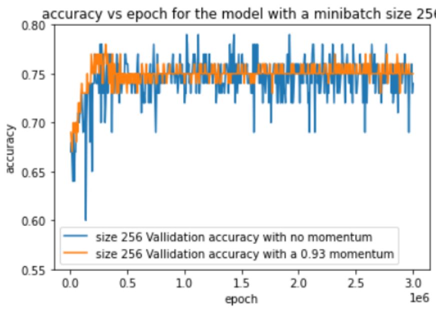


Figure 31: Validation Accuracy for Batch Size 256 with and without Momentum 0.93

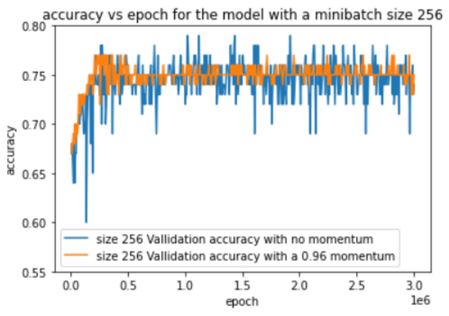


Figure 32: Validation Accuracy for Batch Size 256 with and without Momentum 0.96

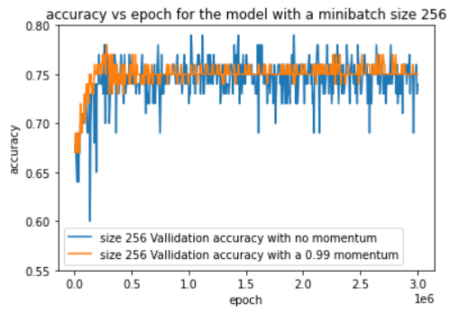


Figure 33: Validation Accuracy for Batch Size 256 with and without Momentum 0.99

4.5 Fake News Dataset

Stemming	Binary Vectorizer	Use Lowercase	Ngram	Val. Acc.	Test Acc.
True	True	True	Uni	74.15	73.67
True	True	True	Uni-bi	77.75	77.43
True	True	False	Uni	74.20	73.67
True	True	False	Uni-bi	77.85	77.43
True	False	True	Uni	72.95	72.07
True	False	True	Uni-bi	75.15	75.13
True	False	False	Uni	72.90	72.03
True	False	False	Uni-bi	75.15	75.13
False	True	True	Uni	74.55	73.33
False	True	True	Uni-bi	78.25	77.20
False	True	False	Uni	75.85	75.17
False	True	False	Uni-bi	77.95	78.33
False	False	True	Uni	74.65	72.83
False	False	True	Uni-bi	76.20	75.63
False	False	False	Uni	74.75	72.90
False	False	False	Uni-bi	76.90	75.90

Table 6: Validation and Test Accuracy for Various Preprocessing Options

Smooth IDF	Sublinear TF	Val. Acc.	Test Acc.
True	True	73.05	72.57
True	False	73.05	72.57
False	True	72.75	72.43
False	False	72.75	72.43

Table 7: Validation and Training Accuracy for Logistic Regression Model using CountVectorizer with binary=True

Balance Weights	Val. Acc.	Test Acc.
False	78	78.2
True	77.65	77.67

Table 8: Effects of Balance Weights