FLORIDA STATE UNIVERSITY

IMPROVED STOCK PRICE MOVEMENT CLASSIFICATION USING NEWS ARTICLES

BASED ON EMBEDDINGS AND LABEL SMOOTHING

By

LUIS VILLAMIL

A Project Report submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Master of My Science

2022

Luis Villamil defended this project report on April 28, 2022.

The members of the supervisory committee were:

Dr. Xiuwen Liu

Professor Directing Traffic

Dr. Shayok Chakraborty

Committee Member

Dr. Zhenhai Duan

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the project report has been approved in accordance with university requirements.

TABLE OF CONTENTS

ABSTRACT

Stock price movement prediction is a challenging and essential problem in finance. While it is well established in modern behavioral finance that the share prices of related stocks often move after the release of news via reactions and overreactions of investors, how to capture the relationships between price movements and news articles via quantitative models is an active area research; existing models have achieved success with variable degrees. In this paper, we propose to improve stock price movement classification using news articles by incorporating regularization and optimization techniques from deep learning. More specifically, we capture the dependencies between news articles and stocks through embeddings and bidirectional recurrent neural networks as in recent models. We further incorporate weight decay, batch normalization, dropout, and label smoothing to improve the generalization of the trained models. To handle high fluctuations of validation accuracy of batch normalization, we propose dual-phase training to realize the improvements reliably. The experimental results show significant improvements, achieving average accuracy of 80.7% on the test set, which is more than 10.0% absolute improvement over existing models. The ablation studies show batch normalization and label smoothing are most effective, leading to 6.0% and 3.4% absolute improvement, respectively on average.

CHAPTER 1

INTRODUCTION

One of the principles forming the foundation of modern finance is the Efficient Market Hypothesis (EMH), the theory that, in an efficient market, the stock price fully reflects all available, relevant information [7]. This poses a problem for anyone attempting to make predictions about stock prices since all the information necessary for prediction is already part of the price, so no financial advantage can be reliably produced [25].

Despite this, it has been shown in several contexts that text-based information such as Twitter [12], news articles [8], and even Reddit [2] could be used to accurately predict future price movements. For instance, Bollen et al. achieved 87.6% accuracy when predicting up and down movements of stock prices from Twitter [2]. One has to look no further than the recent incident with "r/WallStreetBets" on Reddit in order to find an example where sentiment from a text-based internet resource played a significant role in future price movements. In another example, Li et al. [12] described was a 2013 fake tweet about Barack Obama being injured, which caused the Dow Jones Industrial Average to drop 100 points within two minutes. Therefore, it is clear the stock market does move in response to media such as news articles.

However, capturing the complex dependencies between the two systems is a difficult problem. Such a model would need to efficiently compute the relevant news article features in addition to generalizing across all stocks and news articles well. One way in which to represent those complex dependencies is through the use of *Natural Language Processing* (NLP) techniques[6]. In recent years, there has been a revolution in such techniques, which in some cases resulted in neural networks being able to outperform humans at the same task. For instance, NLP has consistently outperformed humans for the past three years on the Stanford SQuAD 2.0, a reading comprehension assessment, with F1 scores as high as 93.2% compared with 89.4% for humans [23].

This project builds upon the work of Du and Tanaka-Ishii [6], who used such NLP techniques to create embeddings of the stock prices and news articles. They then applied those embeddings to the binary classification problem of whether the stock price will move up or down, through the use of a Gated Recurrent Unit (GRU) neural network. The model of this project applied deep learning techniques such as batch normalization, dropout, weight decay, and label smoothing to

improve the performance of a bidirectional GRU (Bi-GRU) neural network with the same settings. The contents of the report go as followed. First, Chapter two goes over the related work of this project. What methods were used to solve this problem and gives background into the techniques used in this solution. Chapter three describes the classification problem in detail, as well as giving a model overview and how the embeddings were generated by Word2Vec and BERT, two NLP techniques. Chapter four describes the dataset, evaluation metrics, and the model's performance on that dataset during the experiments. Finally, Chapter five addresses potential problems with the model and its potential impact, and suggest several improvements for the future.

CHAPTER 2

RELATED WORK

Stock price movement prediction has become am increasingly studied topic among the NLP researchers working on understanding the volatile behavior of financial markets. With the recent development of natural language processing, Automatic sentiment analysis has become highly accurate.

Several studies have developed a number of techniques that tried to predict the stock price movement based on sentiment. For example, Bollen et al. utilized OpinionFinder [27] to find the positive and negative sentiment of tweets, and GPOMS to measure six different human moods from the same tweets in order to predict the closing values from the Dow Jones Industrial Average (DJIA)[2]. While these type of features can be used to classify sentiment of a text, it is now possible to extract useful information from news articles and other text based contexts in the form of continuous vectors. More formally, these fixed-length vectors are called embeddings of the texts. These embedding vectors allow numerical computations as they are expected to contain rich semantic information extracted from the contexts [17].

Word2Vec is an algorithm designed to create trained word embeddings. These embeddings help define meaning to the words by producing a mathematical representation of the word which can be used to capture relationships with other words using similarity functions like dot product, or cosine similarity. Word2Vec has been done in two algorithms, Skip-gram or Continous Bag of Words (CBOW) [18]. In the Skip-Gram model, it uses a set of sentences (corpus) to predict the neighbors (or contexts) of an input word. In the CBOW method, if predicts the word from the contexts words. The introduction of Word2Vec enabled new NLP models to be developed for sequences like ELMo, an application of bi-directional LSTM[20], using recurrent neural networks (RNN) like Long Short-term memory (LSTM) or Gated Recurrent Units (GRU). LSTM is a type of RNN that uses input and output gates to address the vanishing gradient problem [9]. GRU is another type of RNN similar to LSTM but with only two gates, reset gate and update gate, simplifying and speeding up the training [3].

With the recent introduction of transformers, many tasks that were considered challenging have become more feasible. For instance, within the last five years, NLP methods involving transformers

have surpassed humans on the precision and recall, and exact matching aspects of reading comprehension questions [23]. These are given by the F1 and EM scores, respectively, on the Stanford SQuAD dataset. The majority of methods performing better than humans utilize a transformer system known as BERT. Bidirectional Encoder Representations from Transformers (BERT)[4], is a technique that uses transformer models [26] to classify an entire piece of text instead of word by word. This means that the vectors that are output from BERT have been encoded to represent the context of the sentence. Hence utilizing transformers allows for contextualized representations of sentencing meanings that allow models to learn easier.

Du and Tanaka-Ishii[6] built from Hu et al.[10], one of the first works that applied the attention mechanism to news-driven stock price movement classification, by utilizing Word2Vec and BERT to create a dual representation of the headlines of articles to classify the price of a stock utilizing a bi-directional GRU and using a multilayer perceptron for the binary classification. However, they did not explore how deep learning techniques such as batch normalization and label smoothing can further improve generalization. Batch normalization reduces the internal covariance shift of the model through a normalization step that fixes the distribution of layer inputs, thus reducing the dependence of gradients on the scale of the parameters or of their initial values [11]. Label smoothing replaces one-hot encoded (0 or 1) targets with soft targets during calculation of the loss. These soft targets are a linear combination the original labels with a uniform distribution[19], and were first shown by Szegedy et al. [24] to significantly improve model generalization.

CHAPTER 3

THE STOCK PRICE MOVEMENT CLASSIFICATION MODEL

This section will explain the price movement classification problem, briefly explain how the NLP techniques were used to create the news article embeddings and give an overview of the proposed model.

3.1 Price Movement Classification Problem

For the closed price p_t on day t where $t \in \{1, 2, ..., T\}$, and T is the number of trading days in the considered time period [6], the target class y_t was found by:

$$y_t = \begin{cases} 1, & p_t \geq p_{t-1} \\ 0, & p_t < p_{t-1} \end{cases} \tag{3.1}$$

Just like Du and Tanaka-Ishii, the problem is treated as a binary classification problem. For the time window $[t - s + 1, ..., t - 1, t]$ where s is the window size, the time window is considered around day t instead of $t-1$. This is due to not having enough timely information to know whether the price or the article happened first [6].

3.2 NLP to Embeddings

To get the vectors at the word level, the CBOW version of Word2Vec was utilized[1] due to this version having faster training and slightly better accuracy for frequent words [18]. The model was trained with the data from the news article headlines and then, for each headline i the Word2Vec model was used to create word embeddings w_k of dimension 60. All the word embeddings in the headline were further transformed into key vectors n_i^K of dimension 60, using Term Frequency Inverse Document Frequency score (TFIDF) γ_k [6]. The key vector is defined by:

$$n_i^K = \frac{\sum_k \gamma_k w_k}{\sum_k \gamma_k}. \tag{3.2}$$

---

[1]the gensim library provides a vast number of machine learning model implementations, including Word2Vec. available at: https://radimrehurek.com/gensim/models/word2vec.html

TFIDF is a statistic value that gives unique words in a document more importance by computing the number of times the word appears in a document and the number of documents that the word appears in a collection [15]

Figure 3.1: Visualization of word clusters related to stocks of interest: First, the set of companies of interest is sorted based on the market sector they belong to, then it is ordered again in clusters of similar words to each company using t-SNE [13] to reduce the dimensions of the vectors and visualize them. The vectors that belong to companies in the same sectors tend to be around the same space, thus strengthening the idea that embeddings can be used to find feature similarities of data.

Equation (2) captures the headlines as word-level embeddings. However it is not able to get the context of the headline itself. For example, negation could change the meaning of a sentence with one word. To capture the meaning in a more context sensitive way, a BERT encoder as a service[2] was used to output vectors of 1024. The embedding output from the BERT is achieved through the self-attention mechanism in the BERT achieved through transformers [4] [26]. Then, the dimensions of the context vector were reduced to 256 by using principal component analysis (PCA)[6]. This minimizes the number of parameters in the neural network, and thus, creates the value vectors n_i^V from the headlines of the articles.

Every news article n_i was then transformed into a pair of vectors (n_i^K, n_i^V). Utilizing Word2Vec for the word-level "Key" vector n_i^K, and BERT for the context-level "Value" vector n_i^V. To save

---

[2]at the time of this writing, the encoder has changed names to Clip. Implementation can be found here https://github.com/jina-ai/clip-as-service

computation time, these were computed once and stored into a dataset where, for each day t, there is a pair of sets of key/value vectors $N_t^K = \{n_i^K\}_t$ and $N_t^V = \{n_i^V\}_t$ respectively.

The trained stock embeddings were provided by the authors of the original paper [6][3]. Let each stock embedding be s_j where $j = 1, 2, ..., J$, and J=50 (number of stocks picked from R&B dataset). The stock embeddings are then combined with the pair of vectors of each article to output what the original authors called the *market vector*. The procedure to generate the *market vector* is done using the *text feature distiller* process[6].

The process starts by calculating the attention score for every article n_i of day t for stock s_j. the operation is a dot product, as it is the simplest operation that can be done between the two vectors, but the authors suggested an improvement to the method would be to utilize more complex operations in this step. The attention score is defined by:

$$score_{i,j} = n_i^K \cdot s_j, \tag{3.3}$$

The weight of each article i with respect to stock j is found by using a softmax activation function on the attention score:

$$\alpha_i^j = Softmax(score_{i,j}), \tag{3.4}$$

Finally, the *market vector* m_t^j for day t with respect to stock j is computed using a weighted sum of the article weights and the context vectors:

$$m_t^j = \sum_{n_i^V \in N_t^V} \alpha_i^j n_i^V \tag{3.5}$$

The *market vector* is computed for every trading day in the window s with respect to stock j. The window is then assembled as a sequence of market vectors:

$$M_{[t-s+1,t]}^j = [m_{t-s+1}^j, m_{t-s+2}^j, ..., m_{t-2}^j, m_{t-1}^j, m_t^j], \tag{3.6}$$

which is used as the input of the price movement classifier with the target class y_t for day t. The best window size s for this problem is 5. A quick search reveals that distribution of the correct labels for the five day sequence to fairly even between an upward stock price movement (55%) and downward stock price movement (45%).

---

[3]The newest version of the code, is able to generate embeddings without using the originals

3.3   Model Overview

The proposed model takes as input a sequence of market vectors, $M_{[t-5,t]}^j$ with respect to stock j that are used to classify the price movement[6]. The model then outputs the classification value $\hat{y}_t$. The model has a *bidirectional Gated Recurrent Unit*, Bi-GRU layer, which takes in $M_{[t-5,t]}^j$ and an initial hidden state h_0, they both construct a bi-directional encoded vector h_t^O:

$$h_t^O = GRU(M_{[t-5,t]}^j, h_0) \tag{3.7}$$

where t is the trading day we want to classify. the output of the GRU, h_t^O, is then input into a batch normalization layer [11] and a dropout layer. After the dropout layer, the classifier estimates the probability by[6]:

$$\hat{y}_t^j = softmax(MLP(dropout(batchnorm(h_t^O)))) \tag{3.8}$$

where MLP is a fully connected layer used to predict the binary classification. The model then utilizes cross-entropy to calculate the loss, described by:

$$l = -\frac{1}{T}\sum_{t=1}^{T}(y_t^j log(\hat{y}_t^j) + (1 - y_t^j)log(1 - \hat{y}_t^j)), \tag{3.9}$$

Just like it was mentioned in Du et al.[6], A problem of stock movement classification is that single stock classification does not provide enough data to achieve a good performance. In order to address this, A classifier is trained across all the stocks by sharing one classifier [6]. This allows for a more generalized model which avoids the overfitting issue with small sample sizes.
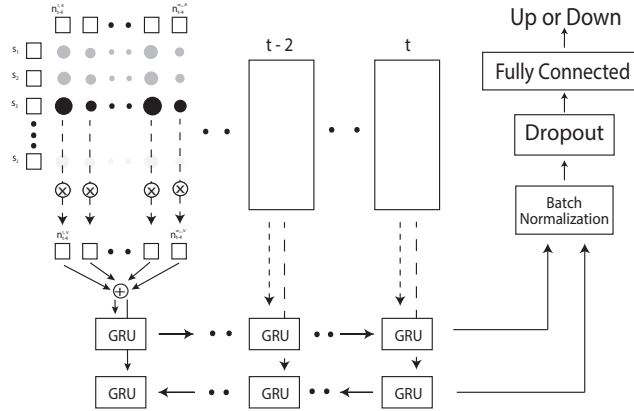

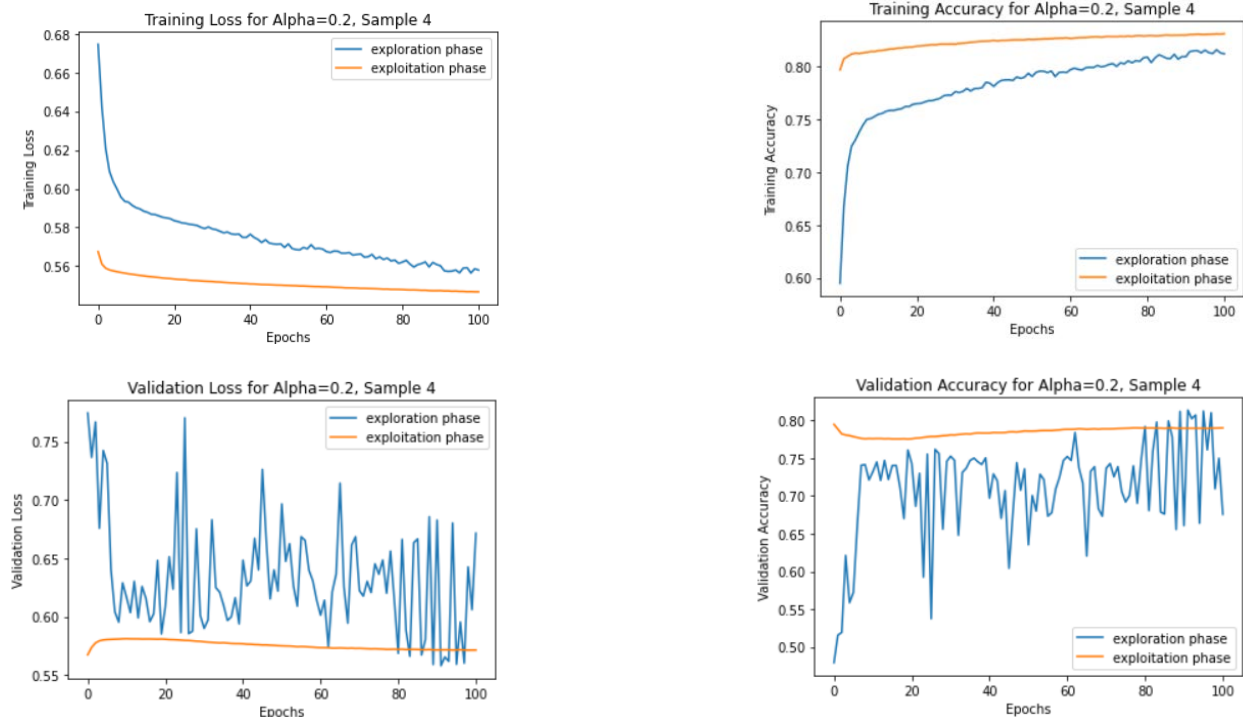Figure 3.2: The system diagram of the proposed deep learning model.

Figure 3.3: Example Loss and Accuracy for Training and Validation Sets for α=0.2

3.3.1 Training Details

The model was trained in two phases with distinct learning rates: an *Exploration Phase* and an *Exploitation Phase*. We call this method a *Dual-Phase Training*.

The exploration phase used a learning-rate of 0.0001, while the exploitation phase used a learning-rate of 0.000001. The model was trained for 100 epochs in the exploration phase. Then, model with the best validation set performance was trained for an additional 100 epochs in the exploitation phase. Figure 3.3 shows an example of the model's performance across the epochs during both phases.

A mini-batch of 64 samples was used during both training and validation. The dropout was set to 0.2. Since this is a classification problem into either "stock price increase" or "stock price decrease" with softmax output, cross-entropy was used as the loss function. The Adam optimizer was used to train the model with weight decay equal to 0.000001 and the learning rates as above.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1   The Dataset

The only public available dataset was the *Reuters & Bloomberg* (R&B) dataset [1]. This dataset contains 552,909 articles combined for a total of 2,605 days or 1,794 trading days. Yahoo Finance was used to generate market data for the 50 stocks that were mentioned in 100 or more different news articles [6]. The market data is then preprocessed to get a set of trading days per sample, $N_{[t-s+1,t]}$, where s is the size of the time window. Each sample consists of a tuple $(N_{[t-s+1,t]}, y_t)$.

The original paper proposed using a threshold to calculate ambiguity in a sample using a the log-return between two consecutive days [6]. The threshold is applied for $[t-1, t]$ in each sample. After discarding the samples with ambiguity, the total number of samples was 32,204. This samples are used to access the vectors of each article per day and compute the market vector with the stock embeddings provided by Du et al.[6].

For each iteration of training, the dataset was randomly split into 60% training, 20% validation, and 20% test. This was done using the "sklearn" function "train_test_split" with the "shuffle" parameter set to "True." Hence, for each new iteration of training, the training, test, and validation sets all may have had different "s"-day windows as individual samples compared to previous training iterations.

4.2   Evaluation Metrics

The standard measurement of accuracy (Acc) and Matthews Correlation Coefficient (MCC) were used to evaluate stock price movement classification. These two metrics are commonly used in related papers to analyze the performance of their models. Acc is defined by:

$$Accuracy = \frac{Number of Correct Predictions}{Total Number of Predictions Made} \tag{4.1}$$

Let a confusion matrix containing the number of samples classified as true positive (tp), false positive (fp), true negative (np), and false negative (fn). MCC is calculated as:

$$MCC = \frac{(tp * tn) - (fp * fn)}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{4.2}$$

[28]

4.2.1 Classification Results

The proposed model achieved an average 80.7% test accuracy over 10 models and 0.631 average MCC score. This result is much higher than the results from the previous methods as seen in Table 4.1.

Model:	Test Accuracy	σ	MCC
Proposed Model	**0.807**	0.962	**0.631**
Du and Tanaka-Ishii [6] model	0.688	1.67	-
Ding et al. [5] model	0.651	-	0.436

Table 4.1: Results from test set evaluations comparing proposed model with previous models

In order to evaluate its capability to predict the market price, an additional prediction model was made by using $[t - s + 1, t - 1]$ and predicting for day t. In order to show that there is no data leak, the prediction model is trained with the classification data and the proposed classification model is trained with a prediction dataset. Table 4.2 shows the results of each run, where it is clear that the prediction model achieves satisfactory results.

Model:	Test Accuracy	MCC
Prediction model with prediction set	0.765	0.557
Classification model with classification set	0.817	0.644
Prediction model with classification set	0.453	0.407
Classification model with prediction set	0.452	0.496

Table 4.2: Results from test set evaluations comparing prediction with classification
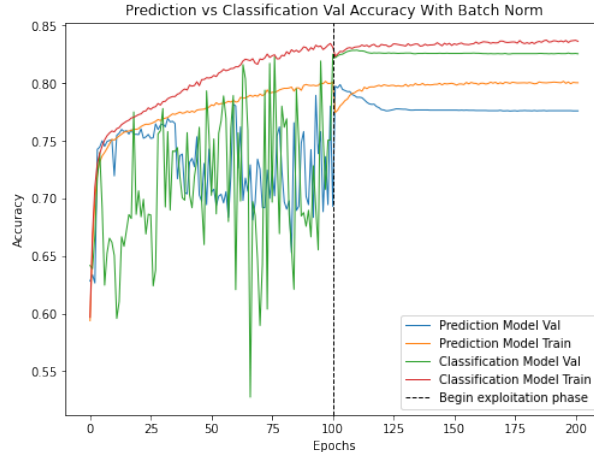
11

Figure 4.1: This figure shows that the proposed model is able to do prediction as well as classification

4.3 Label Smoothing

When dealing with a supervised learning problem with a softmax output, the labels are usually one-hot encoded as vectors of 0's and 1, with 1 indicating the correct label. The cross-entropy loss between the output and correct label vectors is then minimized.

As Szegedy et al. [24] initially showed, label smoothing regularization can be used to significantly improve model performance during classification, and has since gained widespread popularity. Label smoothing is a technique whereby the one-hot encoded labels are adjusted by $y_k^{LS} = y_k(1 - \alpha) + \alpha/K$ for the k-th class when calculating the cross-entropy loss[19]. Here, K is the number of classes, y_k and y_k^{LS} are the labels and modified labels respectively, and α is the label smoothing parameter. In our case, $K = 2$ since the two classes are either a distinct increase in stock price or a distinct decrease in stock price. As Szegedy et al.[24] showed, label smoothing regularization can have a significant impact on model performance and we found the same thing here. Table 4.3 shows the test set accuracy when applying different α's to the one-hot encoded labels. The model was trained with the same dual-phase training method described above with the same parameters. Five different run were cared out, with the training, validation, and test sets all shuffled, and the results averaged.

α:	0	0.05	0.1	0.2	0.3	0.4	0.5
Run 1	0.774	0.794	0.800	0.790	0.787	0.793	0.780
Run 2	0.768	0.781	0.790	0.816	0.804	0.817	0.800
Run 3	0.781	0.780	0.791	0.801	0.804	0.803	0.793
Run 4	0.742	0.774	0.794	0.791	0.798	0.790	0.786
Run 5	0.786	0.774	0.794	0.791	0.798	0.790	0.786
Avg.	0.770	0.780	0.792	0.804	0.798	0.803	0.789

Table 4.3: Accuracies from test set evaluation using label-smoothing with different alphas.

Label smoothing resulted in an increase in the accuracy of the model on the test set by 2-3% for $\alpha \in [0.1, 0.2, ...0.4]$. This is an increase of over 10% compared with the test set accuracy achieved by Du and Tanaka-Ishii (68.8%)[6]. However, as the α increased towards 1, the accuracy began to drop off. This was expected since as $\alpha \to 1$, $y_k^{LS} \to 0.5$ for both classes, making them indistinguishable to the model.

4.4 Ablation Study

An ablation study was done to investigate the impact of several components on the performance of the model by removing each component on each run. The label smoothing parameter was set to 0.2 as the model performed the best on the test set with that α. The same dual-phase training method as above was used with the same parameters. This was done across five replications with shuffled training, validation, and test sets, and the results averaged.

Component Removed	None	Weight Decay	Dropout	Batch Norm	Batch Norm + Dropout
Run 1	0.790	0.781	0.794	0.745	0.741
Run 2	0.816	0.807	0.806	0.742	0.743
Run 3	0.801	0.811	0.800	0.740	0.745
Run 4	0.824	0.799	0.825	0.746	0.742
Run 5	0.791	0.802	0.800	0.729	0.746
Avg.:	0.804	0.800	0.805	0.744	0.743

Table 4.4: Accuracies from test set evaluation after removing selected component(s) with alpha=0.2.

As Table 4.4 illustrates, setting both the weight decay and dropout to 0.0 did not have much impact on the model's ability to predict stock price movement for the test set. In fact, removing the dropout actually cased a marginal increase in performance. However, removing the batch

normalization layer from the model caused the test set accuracy to drop drastically by about 6%. Removing both the batch normalization layer and dropout together had approximately the same effect as removing only the batch normalization layer.

CHAPTER 5

DISCUSSION

5.1   Stock Embeddings Relation to Data

A valid concern for this problem is that the trained stock embeddings that were provided by [6] were trained with the same data that we used to train our own model. In order to address this, a random noise value was introduced to the stock embeddings whose scale was one fifth the standard deviation of the embeddings and trained the model with a dataset created from this new embeddings with the dual phase training method and same parameters. The noise model accuracy resulted in 80.5%, and with an MCC score of 0.63.

Model:	Test Accuracy	MCC
Proposed Model	0.807	0.631
Noise Model	**0.811**	**0.639**
Diff.	0.004	0.008

Table 5.1: Results from test set evaluation comparing proposed model with noise model.

To test the dependency of the data, the model was tested with different window sizes as shown in Figure 5.1. These results show that the system is intensive to the number of days as long as there is a context.
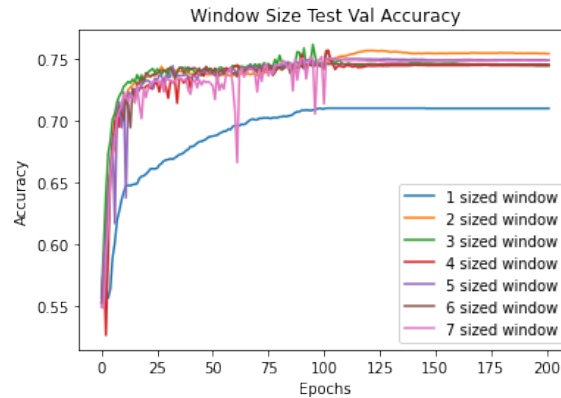

Figure 5.1: Validation Accuracy on different window sizes without Batch Normalization

5.2   Validation Set Fluctuations

It is clear from Figure 3.3 that during the exploration phase when the learning rate is high (0.0001), the accuracy and loss on the validation set fluctuate wildly. However, the validation accuracy and loss when the batch normalization is removed becomes more stable, as seen in Figure 5.2.
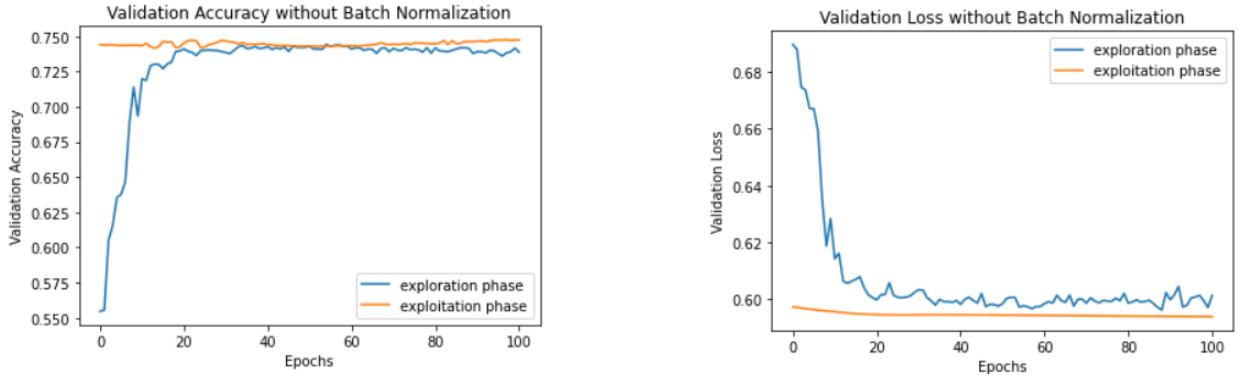

Figure 5.2: Validation Loss and Accuracy for $\alpha = 0.2$ with Batch Normalization Removed

this may be related to the internal statistics that batch normalization tracks across each epoch, but more investigation is necessary to confirm this. However, it is clear from Table 4.4 that performing batch normalization significantly increases the model's accuracy on the test set. This indicates that batch normalization is indeed having the intended effect of improving the model's generalization[11].

5.3   Implications of Efficient Market Hypothesis (EMH)

Perhaps the study is inconsistent with the Efficient Market Hypothesis (EMH), which states that prices in an efficient market fully reflect all available, relevant information[7]. Once patterns are discovered, they provide an advantage only for a limited time, as rational traders adjust their behavior to compensate for this new information, the pattern will be destroyed[25].

However, in recent years, there has been much criticism of the EMH from proponents of behavioral finance[14]. One of the foundational assumptions of economics is that individuals behave rationally when exposed to incentives. This is clearly not always the case and is the basis of behavioral finance[22]. Therefore, since the news and other new information does have an effect on the stock price, this may place the market in so-called "over-reaction and under-reaction states,"

which may be taken advantage of by knowledgeable investors[21]. It is also logical to assume that when relationships between economic forces are more complicated, such as between the news and stocks, different economic actors may interpret those patterns differently, or fail to recognize them at all. Indeed, problems of incomplete information have been studied by economists for decades[16], and the presence of complete information is another assumption at the foundation of classical economics models such as the EMH [7]. Hence, since this assumption is not incredibly realistic, if the model was able to uncover more information about how the news and stocks are related than Du and Tanaka-Ishii[6] through different regularization techniques, the model could provide additional financial advantage.

CHAPTER 6

CONCLUSION

By modifying the neural network architecture and introducing further generalization techniques, the model was able to significantly improve the performance of the model first introduced by Du and Tanaka-Ishii [6], from 68.8% to 79.3% test accuracy. Smoothing the one-hot encoded labels during loss calculation further increased performance to a maximum accuracy of 80.4%. An ablation study was able to confirm the significant impact that batch normalization had on model performance, while the effects of introducing weight decay and a dropout layer were much less pronounced.

Since the same embeddings that were trained by Du and Tanaka-Ishii's model were used, it may be possible to increase the test set accuracy even further by implementing a joint training of the stock and news embeddings together with the network.

The accuracy could also increase by making the classification problem more realistic. One simple way to do this would be to expand the classes from "up" and "down" to "up", "down", and "no change". This would involve reincorporating the stocks with little price movements back into the dataset that were originally removed by Du and Tanaka-Ishii [6].

Additionally, abandoning one-hot encoding altogether in favor of *Soft Label Assignment* (SLA) might lead to further accuracy improvements. As first described by Alishahrani et al. [1], SLA in this case would involve estimating the probability distribution of the stock price movement classifiers for each stock. Then the stocks would be assigned probabilities for each classifier for a given window during the calculation of the loss.

BIBLIOGRAPHY

[1] Ali Alshahrani et al. "Optimism/Pessimism Prediction of Twitter Messages and Users Using BERT with Soft Label Assignment." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

[2] Johan Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." In: *Journal of computational science* 2.1 (2011), pp. 1–8.

[3] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *NAACL-HLT*. 2019.

[5] Xiao Ding et al. "Deep Learning for Event-Driven Stock Prediction." In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 2327–2333. ISBN: 9781577357384.

[6] Xin Du and Kumiko Tanaka-Ishii. "Stock Embeddings Acquired from News Articles and Price History, and an Application to Portfolio Optimization." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 3353–3363. DOI: 10.18653/v1/2020.acl-main.307. URL: https://aclanthology.org/2020.acl-main.307.

[7] Eugene F. Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work." In: *The Journal of Finance* 25.2 (1970), pp. 383–417. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2325486.

[8] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu. "The Predicting Power of Textual Information on Financial Markets." In: *IEEE Intell. Informatics Bull.* 5.1 (2005), pp. 1–10.

[9] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation*. 1997.

[10] Ziniu Hu et al. *Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction*. 2019. arXiv: 1712.02136 [cs.SI].

[11] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[12] Qing Li et al. "Web media and stock markets: A survey and future directions from a big data perspective." In: *IEEE Transactions on Knowledge and Data Engineering* 30.2 (2017), pp. 381–399.

[13] Laurens van der Maaten. "Learning a Parametric Embedding by Preserving Local Structure." In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 384–391. URL: https://proceedings.mlr.press/v5/maaten09a.html.

[14] Burton G Malkiel. "The efficient market hypothesis and its critics." In: *Journal of economic perspectives* 17.1 (2003), pp. 59–82.

[15] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 1999. URL: http://nlp.stanford.edu/fsnlp/.

[16] Robert C Merton et al. "A simple model of capital market equilibrium with incomplete information." In: (1987).

[17] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality." In: *NIPS*. 2013.

[18] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space." In: *ICLR*. 2013.

[19] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. "When does label smoothing help?" In: *arXiv preprint arXiv:1906.02629* (2019).

[20] Matthew Peters et al. "Deep Contextualized Word Representations." In: *NAACL-HLT*. 2018.

[21] Nader Rezaei and Zahra Elmi. "Behavioral Finance Models and Behavioral Biases in Stock Price Forecasting." In: *Advances in Mathematical Finance and Applications* 3.4 (2018), pp. 67–82. ISSN: 2538-5569. DOI: 10.22034/amfa.2019.576127.1118. eprint: http://amfa.iau-arak.ac.ir/article_545661_d70add5bc6d4778c54bd99e2fd44e311.pdf.

[22] Jay R Ritter. "Behavioral finance." In: *Pacific-Basin Finance Journal* 11.4 (2003). Selected papers from the Fourteenth Annual APFA/PACAP/FMA Finance Conference, pp. 429–437. ISSN: 0927-538X. DOI: https://doi.org/10.1016/S0927-538X(03)00048-9. URL: https://www.sciencedirect.com/science/article/pii/S0927538X03000489.

[23] *Stanford SQuAD 2.0 Results*. URL: https://rajpurkar.github.io/SQuAD-explorer/.

[24] Christian Szegedy et al. "Rethinking the inception architecture for computer vision." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[25] Allan Timmermann and Clive W.J. Granger. "Efficient market hypothesis and forecasting." In: *International Journal of Forecasting* 20.1 (2004), pp. 15–27. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(03)00012-8. URL: https://www.sciencedirect.com/science/article/pii/S0169207003000128.

[26]   Ashish Vaswani et al. "Attention is all you need." In: *NIPS*. 2017.

[27]   Theresa Wilson et al. "OpinionFinder: A system for subjectivity analysis." In: *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*. 2005, pp. 34–35.

[28]   Yumo Xu and Shay B Cohen. "Stock movement prediction from tweets and historical prices." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1970–1979.