# Weight-Dropped Long Short Term Memory Network for Stock Prediction with Integrated Historical and Textual Data

Thayogo, Antoni Wibowo, *Member, IAENG*

***Abstract*— Investors and traders need an accurate stock prediction model to help them make decisions. They can use deep learning models such as Long Short-Term Memory Network (LSTM). However, a weakness of LSTM is that it tends to overfit to the training data and have unstable results. To overcome this weakness, this paper proposes using Averaged Stochastic Gradient Descent and Weight-Dropping on an LSTM network (AWD-LSTM). The proposed model regularizes the network by weight-dropping with DropConnect and optimizes the training process using a Non-Monotically Triggered Averaged Stochastic Gradient Descent (NT-ASGD). Additionally, this paper tested with integrating historical data with textual data which was shown to be valuable by other studies. This paper evaluated six variants of the model with different regulizers, optimizers, and data. The results show that (1) DropConnect regulizer performed better than DropOut or No Drop; (2) Adam optimizer is better for stock prediction than NT-ASGD; (3) Adding textual data slightly increases performance; (4) The models were able to gain consistent profits in a market simulation; (5) A variant of the model outperformed a previous study in 4 out of 5 indexes.**

***Index Terms*—LSTM, weight-dropping, regularization, optimization, financial news, stock prediction**

## I. Introduction

THE stock market is a popular medium for investing and trading because of its high potential profits [1]. However, it requires a lot of time and expertise to make consistent profits in the market. A stock price prediction model is needed for a decision support system to help traders and investors make decisions. The state-of-the-art techniques uses deep learning models such as Long Short-Term Memory [2] and Gated Recurrent Unit [3] as they are able to predict non-linear patterns and scale with high amounts of data. However, these models tend to be complex and unstable [4], which are a problem because stock traders want fast and precise predictions. To reduce overfitting and to optimize the training process, a weight-dropped LSTM (AWD-LSTM) was proposed [5]. The model was initially used for language modeling, but the author stated that it can be applied to other sequence tasks.

Manuscript received September 2, 2019; revised April 22, 2020.

Thayogo is with Computer Science Department, Binus Graduate Program-Master of Computer Science, Bina Nusantara University, Indonesia (e-mail: thayogo@binus.ac.id).

Antoni Wibowo is with Computer Science Department, Binus Graduate Program-Master of Computer Science, Bina Nusantara University, Indonesia (e-mail: anwibowo@binus.edu).

Many stock prediction models only uses numerical market data such as historical price [6] and company ratios [7] as features, however progress in textual analysis allows using and mining additional features from textual data [8]. Using these textual features has been shown to improve accuracy in prediction [9].

The purpose of this paper is to propose using AWD-LSTM with additional features from financial news to predict stock prices. The model uses DropConnect which reduces overfitting by randomly dropping a subset of weights. The training is also optimized using a non-monotically triggered variant of averaged stochastic gradient descent (NT-ASGD) which requires less tuning of parameters. A simple sentiment dictionary is used to score the sentiment of daily financial headlines to add as a feature.

Six variants of the model was evaluated with different regulizers (No Drop, DropOut, or DropConnect), optimizers (Adam, NT-ASGD), and data (historical only, or historical + textual data). The results of this paper show an LSTM with DropConnect regulizer, Adam optimizer, and both historical and textual data performed best. When tested in a market simulation, the same model was able to get the highest profits. Finally, a comparison with a previous study show that the model outperformed in 4 out of 5 indexes even without using textual data.

## II. Related Works

This section discusses previous studies that are related to building a prediction model for stock prices. The first part discusses studies that focused on using historical data. The second part discusses studies that focused on using textual data. The third part discusses studies that integrated both historical and textual data. The summary of the related works are summarized in Table I , which compares the types of data and methodology of each works.

### *A. Prediction with Historical Data*

A study [6] used Genetic Algorithm (GA) optimized technical indicator tree -SVM based system to create intelligent stock recommender system with consistent profits. Focusing on multiple technical indicators, the study analyzed the underlying pattern of stock data to create ‘Trade’ or ‘No Trade’ recommendations. Among the three variants, a GA optimized technical indicator with feature selection performed best. Another study [10] took a different approach by recommending a group of stocks (a portfolio) based on

their correlations. By recommending a portfolio of stocks by with association rule mining and fuzzy logic, the results was able to surpass mutual fund returns.

Using financial ratios (book-to-market ratio, ROA, etc.) with neural networks, SVM, and random forest, a study [7] forecasts stock returns in the cross section in the Japanese stock market. The study compared different algorithms and showed that deep neural net performed best. Another research used neural networks to predict stock market prices [11]. The study proposed a combination of neural network and a Nonlinear Autoregressive Exogenous (NARX) model. The model was able to train quickly and effectively predict stock prices.

Recent studies uses more complex variants of neural networks such as Long Short Term Memory (LSTM) networks. A study [2] used uses financial market data with LSTM for stock predictions. The paper showed that LSTM outperforms previous models such as random forest (RF) and deep neural net (DNN). However, its accuracy still hovers around 54%. This shows the limit of using only historical data in predicting stock prices.

### *B. Prediction with Textual Data*

While historical data is still essential in predicting stock prices, many studies have found improvements in accuracy by augmenting it with textual data. A review paper on text mining for market prediction [8], showed that the most popular feature selection technique is Bag-of-Words. According to the paper, text mining can be improved by using better semantic techniques such as WordNet or specially customized dictionaries. For example, a study [3] created Stock2Vec as a financially-trained word embedding for its news analysis. SVM and Naive Bayes was the most popular machine learning models, whereas neural networks were still underused.

A review of the state-of-the-art models in sentiment analysis using Twitter data [12] showed that using twitter data is still unreliable at around 60% accuracy due to its unstructured contents. A more formally-written source of text such as financial news or documents should be more reliable. According to a review on using deep learning for sentiment analysis [13], a good approach on the document level is to use word embedding based on neural networks and using SVM for classification.

A paper [14] used deep learning for event-driven stock prediction. It extracted events from news and train them with a neural tensor network. Then, it used a deep convolutional neural network, to model the influence of the events on stock price. The results showed that event-embedded based document are better than discrete-events based methods. Another study [15] took a unique approach by using recommendation data from online stock communities such as ShareWise. By taking advantage of collective wisdom, the system was able to outperform market benchmarks. While the techniques used was relatively simple compared to other papers above, it showed that there are other creative data sources that can help predict stock prices.

### *C. Integrating Both Historical and Textual Data*

To use both historical and textual data together, researchers searched for ways to integrate them effectively. A paper evaluated an intraday stock recommendation system using market and textual data and integrated them to find joint patterns [9]. The goal is to bring different data sources together and create an "end-to-end" recommendation system. As there are many predictors available, the paper used GainSmarts to select the strongest features. Then, the data were trained using neural network. The best results was obtained when using market data, simple news item counts, categorization into business events, and calibrated sentiment scores as predictors.

To improve upon previous works, a paper used state-of-the art deep learning models while both using historical S&P 500 prices and news articles from Reuters and Bloomberg [3]. It also incorporated technical indicators such as Stochastic oscillator (%K), William (%R), Relative Strength Index (RSI). The paper contributed by proposing a two-stream Gated Recurrent Unit Network (TGRU). It also proposed Stock2Vec which is a sentiment dictionary that was specially trained on stock news. The steps included processing the articles, labeling them, embedding it with Stock2Vec, and finally implementing TGRU network on the dataset.

A study [16] proposed a model using LSTM and emotional analysis to predict stock prices. The emotional analysis worked by collecting public opinion on forum posts related to the Shanghai Composite Index and labeling their emotional tendency through manual labeling and a sentiment dictionary. A total of 15 input variables were used including technical data such as highest and lowest trading price of the day. The model was able to perform well by capturing the long-term dependence of the stock data.

A study [17] used both market and news headline data in an LSTM model to predict stock prices. The paper grouped 10 companies together to capture their correlations. Each news articles were represented using Paragraph Vectors and were concatenated for the 10 companies. If there were no news in a single time step, the vectors were filled with zeros. If there were multiple news in a single time step, the vectors were averaged. The stock prices were normalized to be between [-1,1]. When combining them together, the dimensions were reduced so that both side has a more balanced dimension. The results showed an improvement when using Paragraph Vectors and LSTM model.

### *D. Trends and Conclusion*

The trend in stock prediction is to incorporate more relevant data sources as predictors. Starting from simply using historical data, studies began using more sophisticated technical indicators [10]. Eventually, as text processing developed, news were used to augment the data resulting in increased accuracy [9]. Then, deep learning models improved prediction such as using LSTM for prices [2] and using better word processing techniques such as Word2Vec and Paragraph Vector.

To conclude, there are fewer studies that has integrated historical and textual data together, and these studies claimed to have increased performance after adding textual data. More studies need to combine both historical and textual data together to improve results. Additionally, there are no studies that have regularized and optimized LSTM with AWD-LSTM stock prediction. Therefore, this study contributes by improving LSTM with AWD-LSTM and integrating both historical and textual data.

TABLE I
RELATED WORKS

| Reference | Historical Data | Textual Data | Feature Extraction | Model | Measure Performance |
|---|---|---|---|---|---|
| [18] | Turkey ISE National 100 Index | - | - | ANN, SVM | Directional Accuracy |
| [19] | DJIA Index | Tweets | Profile of Mood States (POMS) | Self-Organizing Fuzzy Neural Networks | Positive and Negative Recall |
| [20] | US S&P 500 index | - | - | Fuzzy Clustering + Fuzzy NN | RMSE |
| [10] | Various indexes | - | - | Association Rule Mining | ROI, Precision, Rebalancing precision |
| [9] | 72 individual stocks | News | Bag-of-Words, Business Events, Sentiment | Neural Network, Stepwise Logistic Regression | Returns, Sharpe Ratio |
| [21] | 3 Latin-American indices | - | - | ANN + GARCH | RMSE, MSE, MAE, MAPE |
| [22] | China SSE | - | - | PCA + SVM | Accuracy classified by return rank |
| [14] | S&P 500 and 15 individual stocks | News Headline | Event Embedding | Convolutional Neural Network | Acc, MCC, Profit |
| [6] | 4 individual stocks | - | - | GA-optimized decision-tree-SVM | Precision, Recall, Accuracy, F-Measure |
| [23] | India BSE and CNX | - | - | KNN + SVM | MSE, RMSE, MAPE |
| [24] | 18 individual Stocks | Tweets | Topic-sentiment | SVM | Acc, MCC |
| [25] | China SSE and SZSE | - | - | IG+SVM+KNN | Directional Accuracy |
| [26] | Korea KOSPI 38 stock returns | - | - | DNN | NMSE, RMSE, MAE |
| [7] | MSCI Japan Index | - | - | SVR, RF, Ensemble | MSE, Return, Risk |
| [27] | Korean Stock Price Index | - | - | GA-optimized LSTM | MSE,MAE,MAPE |
| [3] | S&P 500 | News | Stock2Vec | TGRU | Accuracy, Precision, Recall |
| [11] | DJI | - | - | Neural Network + NARX | MSE |
| [16] | Shanghai Composite Index | Public Opinion from Memories | Sentiment Dictionary | LSTM with Emotional Analysis | MSE |
| [2] | S&P 500 | - | - | LSTM | Return, STD, Sharpe Ratio, Accuracy % |
| [28] | Multiple indices | - | - | Rough set + Wavelet Neural Network | RMSE, MAD, MAPE, CP, CD |
| [15] | 50 companies on Tokyo Stock Exchange | News Headlines | Paragraph Vector | LSTM | Trade Gains |

## III. Theory and Methods

### A. Deep Learning

The term deep learning is usually used to describe neural network with many hidden layers, hence a deep network. Formally, deep learning is defined as a class of machine learning algorithms that uses multiple layers of processing units where the next units uses the previous units as inputs [29]. These methods has been shown to give state-of-the art results in many fields such as computer vision, audio processing, and natural language processing.

Recurrent Neural Networks (RNN) are a type of neural networks containing loops so that one node can pass information into the other nodes. It is especially useful for sequencing problems as it is able to retain information for previous nodes. Long Short Term Memory (LSTM) is a class of RNN that is able to capture long-term dependencies.

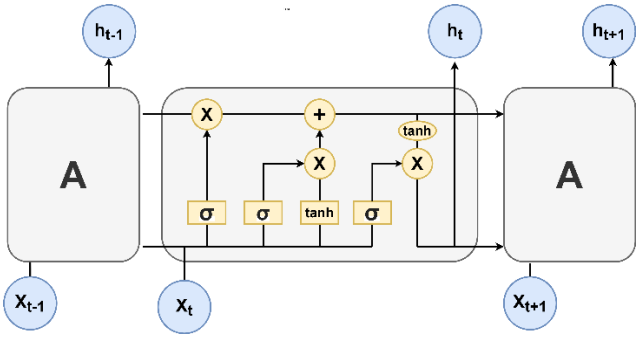


Fig. 1. LSTM Network, redrawn from [30]

Fig. 1 visualizes an LSTM network which was redrawn from [30]. Each nodes (A) takes both the input ($x_t$) and the memory from previous node ($c_t$) to calculate output ($h_t$). Information from previous sequences are recurrently used as input for the next, therefore it is useful for a time series problem. Within the node (A), LSTM calculates a hidden state, $s_t$, as follows:

$$i = \sigma(x_t U^i + s_{t-1} W^i)$$
$$f = \sigma(x_t U^f + s_{t-1} W^f)$$
$$o = \sigma(x_t U^o + s_{t-1} W^o)$$
$$g = \tanh(x_t U^g + s_{t-1} W^g)$$
$$c_t = c_{t-1} \sigma \circ f + g \circ i$$
$$s_t = \tanh(c_t) \circ o$$

Where $i$ is called the input gate, $f$ is the forget gate, and $o$ is the output gate. These gates have a value between 0 and 1 because of the sigmoid function. They will then be multiplied element-wise with a vector to define how much of that vector will go through the gate. The input gate controls how much of the newly computed state to let through. The forget gate controls how much of the previous state to let through. Lastly, the output gate controls how much of the internal state to expose to the external network. The $g$ is a candidate hidden state computed from the current and previous unit state, and $c_t$ is the internal memory of the unit which combines how much of the old and new state we want.

### B. Regularization and Optimization

Recurrent neural networks are prone to overfitting, therefore regularization techniques are formed to solve the overfitting problem and improve performance. A method of regularization involves weight-dropping such as Dropout and DropConnect. Fig. 2 is a redrawn illustration from [32] which shows the difference between No Drop, DropOut, and DropConnect networks. No Drop means there is no changes to the network. DropOut [31] reduces overfitting by preventing complex co-adaptions in the data through randomly dropping feature detectors on each training case. DropConnect [32] generalizes the idea of DropOut. Whereas Dropout sets randomly selected subset of activations to zero, DropConnect sets a randomly selected subset of weights to zero.

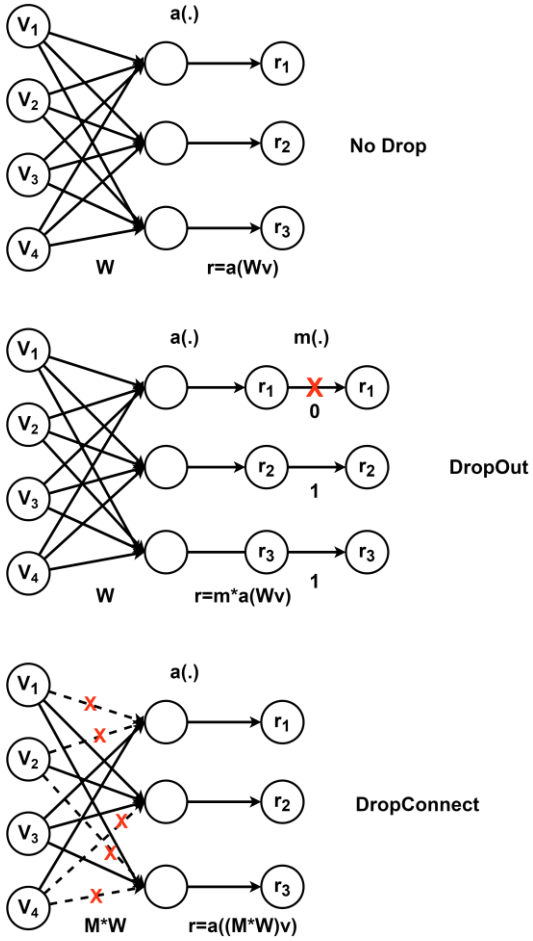


Fig. 2. Comparison of Weight Drop Networks, redrawn from [32]

A deep learning network usually trains itself through the use of backpropagation. However, there are several techniques to optimize the training process. The most basic training method for neural networks is the Stochastic Gradient Descent (SGD). SGD minimizes an objective function, which in this case is errors, iteratively in an incremental gradient descent with the following steps:

$$w_{k+1} = w_k - \gamma_k \widehat{\nabla} f(w_k) \quad (2)$$

A possible improvement to SGD is the Averaged Stochastic Gradient Descent (ASGD), which uses the average as a final solution to reduce noise. ASGD returns the following as a solution:

$$w = \frac{1}{K - T + 1} \sum_{i=T}^{K} w_i \quad (3)$$

Where $K$ denotes the total number of operations and $T<K$ is a user-specified averaging trigger. The goal of ASGD is to find the optimal point faster by averaging the final solution because normal SGD tends to fluctuate around the optimal solution.

### C. Textual Feature Extraction

The art of representing text as vectors has grown significantly these past few decades [8]. The traditional method is the Bag-Of-Words representation. This method represents each documents as a sparse vector with its dimension equal to size of the vocabulary. Each position in the vector represents a word as the number of times it occurr in the document. For example, the word 'brown' in the document 'a brown dog' would have a vector (0,0,1). Another method using word counts is the TF-IDF. The model improves upon Bag-of-words by giving weights to words based on its importance to a document:

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i}) \quad (4)$$

Where $tf_{i,j}$ is the frequency of term $i$ in document j, $df_i$ is the number of documents with term $i$, and $N$ is the total number of documents. The higher the word frequency in the entire corpus, the less valuable it is. The higher the word frequency in a single document, the more valuable it is. Therefore a word has high representational value of a document when it only appears distinctively in that document.

While traditional methods such as Bag-of-Words and TF-IDF are fast and simple, they ignore the word's position in the document. It does not capture the word's context, semantics, and relationship with each other. While not the most effective, bag-of-words and tf-idf representation is easy to understand and still provides reasonable performance. They are often the baseline method for many studies for word representations.

Word embedding, or distributed representations, is a method of representing words as vectors where each words has some dependence on other words. Contrary to the previous methods where each words' vector are independent, here each word's vector are dependent on the words around it. Word embedding can be compared to one another with cosine similarity. A popular ideal example of word embedding is where we can do: ("king"-"man") +"woman" = "queen".

Word2Vec by [33] is a word embedding technique that uses a shallow neural network. Word2Vec has two methods : Continuous Bag of Words (CBOW) and Skip Gram. The CBOW model accepts a context word as input and predicts a target word. The Skip Gram Model is the inverse of CBOW. It accepts a target word as input and predicts its context words. Along this process, the vector representation of each word is obtained.

## IV. Research Framework

The general process of the methodology of this study is shown in Fig. 3. Historical stock prices and news articles were collected.The news data were represented as numerical vectors through feature extraction. The textual features were then integrated with historical price and were matched in the same time dimension. Then, a deep learning model (AWD-LSTM) was used to learn the underlying patterns for prediction. Finally, the model's performance was evaluated. The evaluation process included creating six variants of the model and comparing them by cross-validation, market simulation, and comparison with previous study.

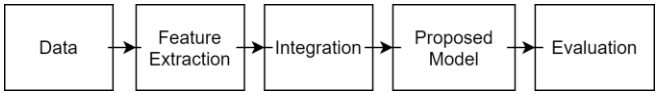


Fig. 3. Research Framework

## V. Methodology

### A. Data

Historical data was obtained from daily S&P 500 index for the year 2007-2018. The variables included date, opening price, and closing, highest price, and lowest price. The data excluded days where there are no stock trading activities (Weekends and holidays). In total there were 3018 entries of historical data. Text data were obtained from daily financial news from Reuters also in the year 2007-2018. There were a total of 8,551,440 news articles available. News data were processed by removing stop words, symbols, numbers, and punctuations. Furthermore the words were reduced to its base form by stemming and lemmatization. This converts words such as "helping" to "help" in order to reduce the amount of possible vocabularies. The final results were individual word tokens ready to be converted to vectors.

### B. Feature Extraction

A pre-made sentiment dictionary was used to extract features from each news headlines. Loughran and McDonald is a financial sentiment dictionary that have mapped common words found in financial news as Positive or Negative. For example, the word 'DECLINE' is represented as a negative word in the form of (0,1), whereas the word 'SURPASSED' is represented as a positive word in the form of (1,0). Words that were not found in the dictionary were represented as (0,0). Each news headline were represented as a sentiment score calculated from calculating Positives minus Negatives.

### C. Integration

To integrate the output text vectors with historical data, their time dimension has to be aligned. Each time point is a single day, excluding weekends and holidays where the stock market is closed. Historical data is available in every time point, however a single time point may have multiple or no news at all. Furthermore, a news article's effect may not be limited to only a single day. Therefore the integration method has to be carefully examined. Table II shows the integrated data, where *Predicted Price* is a function of *Historical Data* and *Textual Data*. The difference between *Predicted Price* and *Actual Price* will then be used as the performance measure of the model.

TABLE II
DATA INTEGRATION METHOD

| Time | Historical Data | Textual Data | Actual Price | Predicted Price |
|---|---|---|---|---|
| $t$ | $P_t$ | $S_t$ | $p_{t+1}$ | $\hat{p}_{t+1}$ |
| $t-1$ | $P_{t-1}$ | $S_{t-1}$ | $p_t$ | $\hat{p}_t$ |
| . | . | . | | |
| . | . | . | | |
| . | . | . | | |
| $t-n$ | $P_{t-1}$ | $S_{t-n}$ | $p_{t-n+1}$ | $\hat{p}_{t-n+1}$ |

Where $t$ is today, and $n$ is the amount of samples. $P_t$ represents a vector of historical prices, and $S_t$ represents the sentiment score. $p_{t+1}$ is tomorrow's actual price and $\hat{p}_{t+1}$ is the tomorrow's predicted stock price. Since $p_{t+1}$ is still unknown, the first row of the data will be excluded. Equation 4 and 5 shows how to obtain the vectors $P_t$ and $S_t$ respectively:

$$P_t = p_t, p_{t-1}, \dots, p_{t-60} \quad (3)$$

$P_t$ consists of a 60 days sliding window starting from price $p_t$ to $p_{t-60}$.Where $p_t$ is the price of time $t$. All price values were scaled to a value between 0 and 1 based on the minimum and maximum values. The sentiment score vector is obtained from the following:

$$S_t = s_t, s_{t-1}, \dots, s_{t-60} \quad (4)$$

$$s_t = Sum(n_{t,1}, n_{t,2}, \dots, n_{t,m}) \quad (5)$$

$S_t$ calculates the total score of sentiment $n_{t,1}$ to $s_{t,m}$. Where $n$ is the sentiment score of time $t$ and news $m$. The $m$ indexes the number of news in a day. A problem is that each data point only has information of the news published during that day. Furthermore, the method is forced to only include news data on days where the stock market opens, therefore ignoring news on weekdays and holidays. To solve this, news for days with no stock prices are averaged to the next day with stock price data. For example, the news score for Monday is the average of Saturday, Sunday, and Monday. With this method, news data on weekdays and holidays are also captured.

### D. Proposed Method (AWD-LSTM)

The implementation of AWD-LSTM is a combination of regularization through DropConnect, and optimization through NT-ASGD (a variant of ASGD). DropConnect regularizes a network by randomly setting a subset of weights to zero. Applying this to an LSTM network, the hidden-to-hidden weights at the input ($W_i$) , forget ($W_f$ ), output ($W_o$) gates and cell state ($W_g$) are randomly dropped. Therefore, the DropConnect Layer ($r$) is calculated as follows:

$$r = a((M * W)v) \quad (5)$$

Where $a$ is a non-linear activation function, M is a binary weight mask, W is the fully-connected layer weights for ($W_i$, $W_f$ , $W_o$ , $W_g$) and $v$ is each cell inputs. The operation is performed only once during the forward and backward propagation, thus making minimal impact on training speed. The result encourages smaller weights which simplifies the model and reduces overfitting. Fig. 4. shows how DropConnect is applied to an LSTM network. Each input features are processed through an LSTM network which is followed by a DropConnect layer. Some information are passed for the next recurrent LSTM network, while the result continues to the next set of layers. At the end of the layers, a dense layer combines the nodes to produce a single output.

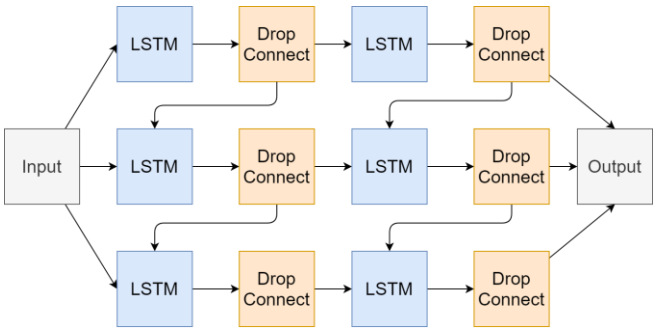


Fig. 4. Weight-Dropped LSTM Network

Other than reducing overfitting through weight-dropping, AWD-LSTM also uses a variant of Average SGD called NT-ASGD to optimize the training process. A weakness of Average SGD is that it has unclear tuning guidelines for T and for the parameter K. To solve this problem, the variant NT-ASGD is non-monotonically triggered as shown in the following algorithm by [5]:

**Inputs:** Initial point $w_0$, learning rate $\gamma$, logging interval $L$, non-monotone interval $n$.

Initialize $k \leftarrow 0, t \leftarrow 0, T \leftarrow 0$, `logs` $\leftarrow$ `[]`
**while** stopping criterion not met **do**
  Compute stochastic gradient $\hat{\nabla} f(w_k)$ and take SGD step (1).
  **if** $\mathrm{mod}(k, L) = 0$ and $T = 0$ **then**
    Compute validation perplexity $v$.
    **if** $t > n$ and $v > \min_{l \in \{t-n, \cdots, t\}}$ `logs[l]` **then**
      Set $T \leftarrow k$
    **end if**
    Append $v$ to `logs`
    $t \leftarrow t + 1$
  **end if**
**end while**
**return** $\frac{\sum_{i=T}^{k} w_i}{(k-T+1)}$

When the validation metric does not improve after multiple cycles, the algorithm triggers the averaging. The non-monotone interval hyperparameter, n, controls this. The hyperparameter $n$ will be set to 5 as recommended by Stephen et al. (2018). Therefore, after 5 failed attempts to improve the metric, the algorithm will switch back to ASGD. As a constant learning rate $\gamma$ is used, there is no need for further tuning.

### E. Evaluation

The evaluation process had three sets of experiments: Cross-Validation, Market Simulation, and Comparison with Previous Study. The first experiment compared six variants of the proposed model on a single data with cross-validation to find the best performing model. The second experiment tested the models on an automated trading system to simulate the profits gained in real-life conditions. Finally, the final experiment compared the models to a previous study [28] using the same data. The main metric for accuracy will be evaluated by Root Mean Square Error (RMSE). RMSE is calculated following [34]'s notation as:

$$RMSE_{fi} = \sqrt{\frac{1}{FN_i}\sum_{j=1}^{FN_i}(p_{t+1,j} - \hat{p}_{t+1,j})^2} \quad (6)$$

Where $RMSE_{fi}$ is the RMSE for fold $i$, $FN_i$ is the sample size of the testing data in fold $i$, $p_{t+1,j}$ is tomorrow's actual price, and $\hat{p}_{t+1,j}$ is tomorrow's predicted price. For cross-validation, the folds were averaged using Prediction Error Sum of Squares (PRESS), which is shown in Equation 6:

$$PRESS_{mi} = \frac{1}{4}\sum_{fi=1}^{4}(RMSE_{fi})^2 \quad (7)$$

Where $PRESS_{mi}$ is the PRESS for model $i$ and $RMSE_{fi}$ is the RMSE for fold $i$. Therefore, the best model will have the lowest PRESS value. Additionally, the training time of each models were evaluated as the number of seconds it takes to finish training the model.

## VI. Experiment Results

### A. Cross-Validation

Six variants of the proposed model were tested to find the model with the best performance. The first model followed a standard LSTM network. The second model added DropOut as a regulizer. The third model changed DropOut into DropConnect. The fourth model adds news data to Model 3. The fifth model used the NT-ASGD optimizer. Finally, the 6th model added news data to Model 5. The regulizers were set with a probability of 0.5. All experiments uses 4 LSTM layers + 1 Dense layer and 50 nodes. The training is repeated on 100 epochs with a batch size of 32.

All models were cross-validated with different amounts of training and testing data on the the S&P 500 index. The evaluation used k-fold cross-validation method that retains the time consistency by forward-chaining. Each fold will calculate its own RMSE, which will then be averaged into PRESS as the final score of each model. The folds are structured as follows:

- Fold 1: Training (2007-2010), Testing (2011-2012)
- Fold 2: Training (2007-2012), Testing (2013-2014)
- Fold 3: Training (2007-2014), Testing (2015-2016)
- Fold 4: Training (2007-2015), Testing (2016-2018)

Table III show the average result of each models. Table IV shows the individual results for each folds. Fig. 5 illustrates each model's loss per epochs during the training process. The results shows that an LSTM model with DropConnect (Model 3) performed better than with no regulizer (Model 1) and DropOut (Model 2). This shows that regularizing our LSTM model with DropConnect can successfully improve performance. However, regularizing with DropOut decreased the performance instead (Model 2). This shows that DropOut's method of setting subsets of activations randomly to zero causes too much loss of information compared to DropConnect's method of setting subsets of weights randomly to zero. Model 4 shows that adding news feature was able to increase performance slightly.

Using NT-ASGD as optimizer (Model 5 and 6) did not improve performance. The training time is also significantly longer when using NT-ASGD. This shows that Adam optimizer is still better suited for stock prediction and that NT-ASGD is more useful its original purpose in language modeling [5]. Adding news feature (Model 6) was still able to increase the performance slightly similar to Model 4.

TABLE III
Model Comparison

| No | Input | Model | Regulizer | Optimizer | Training PRESS | Testing PRESS | Training Time |
|---|---|---|---|---|---|---|---|
| 1 | Price | LSTM | No Drop | Adam | 11.7943 | 19.4588 | 493.9174 |
| 2 | Price | LSTM | DropOut | Adam | 17.9533 | 29.1330 | 616.8222 |
| 3 | Price | LSTM | DropConnect | Adam | 9.8852 | 17.0913 | 575.7456 |
| 4 | Price + News | LSTM | DropConnect | Adam | 8.8377 | 16.4755 | 1088.4994 |
| 5 | Price | LSTM | DropConnect | NT-ASGD | 30.1706 | 52.6467 | 702.1594 |
| 6 | Price + News | LSTM | DropConnect | NT-ASGD | 24.3464 | 50.5623 | 1301.1703 |

TABLE IV
CROSS VALIDATION RESULTS FOR EACH FOLDS

| Model 1 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 8.0118 | 16.3939 | 240.3699 |
| 2 | 13.3908 | 14.9683 | 429.5659 |
| 3 | 6.3158 | 19.8980 | 554.4083 |
| 4 | 19.4588 | 26.5747 | 751.3255 |
| Average | 11.7943 | 19.4588 | 493.9174 |

| Model 2 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 17.1768 | 27.0394 | 294.6913 |
| 2 | 18.6632 | 20.1073 | 519.6491 |
| 3 | 12.6905 | 33.5112 | 727.0733 |
| 4 | 23.2827 | 35.8742 | 925.8752 |
| Average | 17.9533 | 29.1330 | 616.8222 |

| Model 3 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 7.9628 | 16.2877 | 280.5763 |
| 2 | 13.4884 | 14.4246 | 485.2615 |
| 3 | 5.2980 | 19.3778 | 655.8619 |
| 4 | 12.7915 | 18.2750 | 881.2829 |
| Average | 9.8852 | 17.0913 | 575.7456 |

| Model 4 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 8.4319 | 15.1404 | 537.8242 |
| 2 | 9.1548 | 11.2653 | 881.1681 |
| 3 | 6.3398 | 19.0319 | 1303.0805 |
| 4 | 11.4243 | 20.4647 | 1631.9249 |
| Average | 8.8377 | 16.4755 | 1088.4994 |

| Model 5 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 24.8512 | 46.5905 | 366.6784 |
| 2 | 38.5754 | 43.6641 | 589.7876 |
| 3 | 16.9829 | 59.8609 | 834.8687 |
| 4 | 40.2729 | 60.4713 | 1017.3030 |
| Average | 30.1706 | 52.6467 | 702.1594 |

| Model 6 | | | |
|---|---|---|---|
| Fold | Training | Testing | Time |
| 1 | 23.2352 | 47.5598 | 676.1469 |
| 2 | 25.3609 | 34.7261 | 1089.3147 |
| 3 | 17.2101 | 58.2368 | 1505.5307 |
| 4 | 31.5794 | 61.7265 | 1933.6887 |
| Average | 24.3464 | 50.5623 | 1301.1703 |

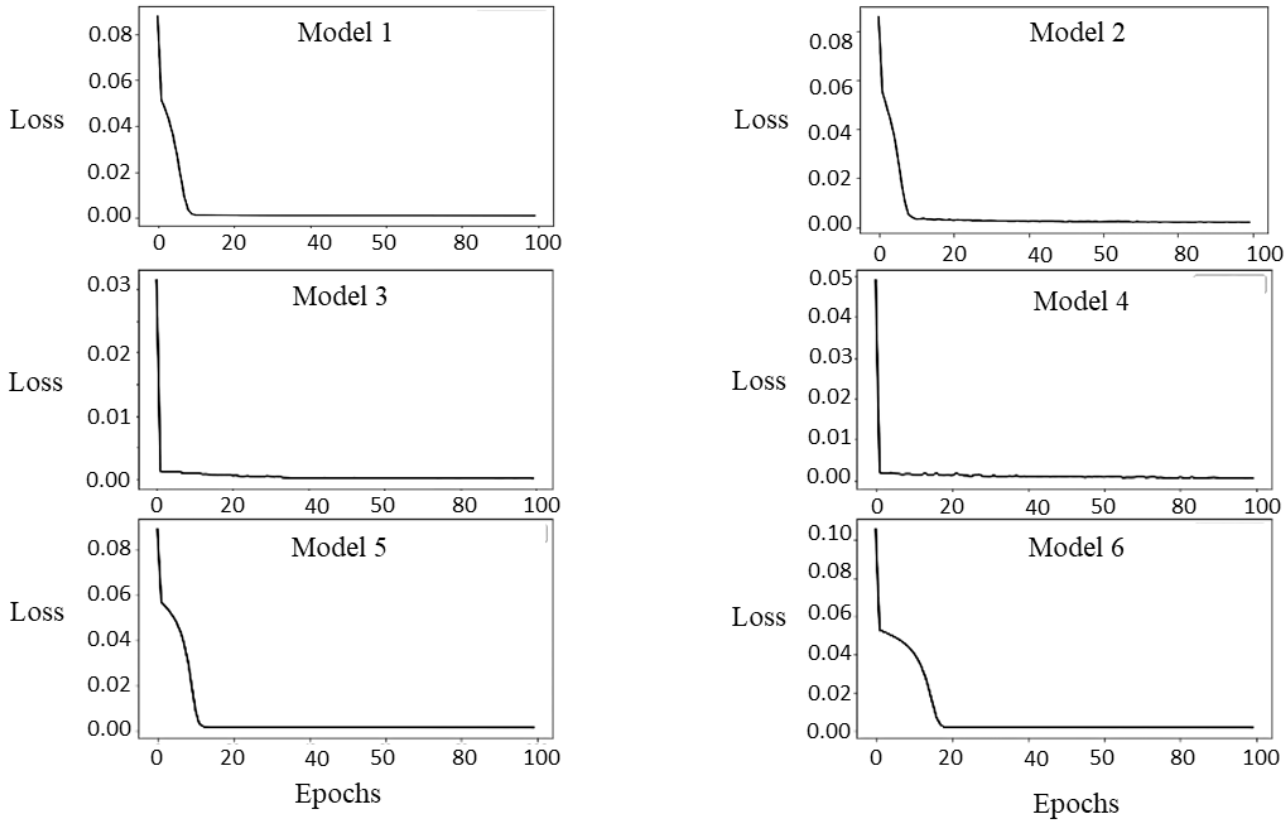


Fig. 5. Training Loss per Epochs

### B. *Market Simulation*

In this section, the models were tested on an automated trading system that imitates real-life trading scenario. The setup was inspired by the simulation done by [3]. The simulation used data from Fold 4 (training data from 2007-2015, and testing data from 2016-2018). A difference from [3]'s simulation is that instead of a binary buy/sell recommendation, our model outputs the next day's predicted stock price. To decide on a buy/sell recommendation, a cutoff value of 1% is used. If the predicted price is +1% or higher than today's price, the system recommends to buy. If it is -1% or lower, it recommends to sell. Otherwise, it recommends to hold. Each transaction was charged with a transaction cost of 0.25% of the trading amount. Each day was limited to a single buy or sell transaction to minimize transaction costs. The starting capital is set to USD 10,000. The trading rules are similar to [35] as follows:

- If you are not holding stocks and the system recommends to buy, then buy with all of your current money.
- If you are not holding stocks and the system recommends to sell, then do nothing.
- If you are holding stocks and the system recommends to buy, then do nothing.
- If you are holding stocks and the system recommends to sell, then sell all of your stocks.

Table V shows the Net Capital, Profits, and Return of each Models by the end of the simulation. Net Capital is the final capital after selling all stocks by the end of the simulation. Profits is the amount of money that the model gained starting from the original capital of 10,000. Return is the amount of profits expressed as a percentage of the original capital. Fig. 6 illustrates the growth of net capital for each models. While the results might seem impressive, note that any strategy would most likely yield a positive return as the data is already trending upwards. For reference, if an investor were to buy at the beginning of the simulation and hold until the end without any trading, they would already get a return of 39.61%. Therefore, it is more important to compare among models. The comparison showed that Model 4 has the highest return out of the 6 Models.

TABLE V
MARKET SIMULATION RESULT

| Model | Net Capital | Profits | Return |
|---|---|---|---|
| Model 1 | 24059 | 14059 | 140.59% |
| Model 2 | 19277 | 9277 | 92.77% |
| Model 3 | 24162 | 14162 | 141.62% |
| Model 4 | 24738 | 14738 | 147.38% |
| Model 5 | 16887 | 6887 | 68.87% |
| Model 6 | 16805 | 6805 | 68.05% |

### A. *Comparison with Previous Study*

The final set of experiments was done to compare this study with the results of a previous study [28]. The study was chosen because it also evaluated using RMSE and has results on several different indexes. The study used a Wavelet Artificial Neural Network on 5 different indexes: SSE Composite Index (SSE) from 04/10/2009 to 06/04/2004, Shanghai Shenzhen CSI 300 Index (CSI 300) from 02/03/2009 to 04/02/2014, All Ordinaries Index (AORD) from 04/01/2009 to 03/26/2014, Nikkei 225 Index (NIKKEI 225) from 03/15/2009– 05/25/2014, and Dow Jones Industrial Index (DJI) from 10.22.2009 to 07/18/2014.

Table IV compares this study's models with the previous study [26]. The models were compared on their RMSE on 5 different indexes. Model 4 and 6 was excluded from comparison as this study was not able to obtain news data which are relevant to these specific stock indexes. Since Reuters is mainly based on U.S. financial news, it would be out of context to apply it to the other indexes. The results show that Model 3 outperformed all models in 4 out of 5 indexes. It shows that our model was able to perform better than a previous study pretty consistently under the same conditions.

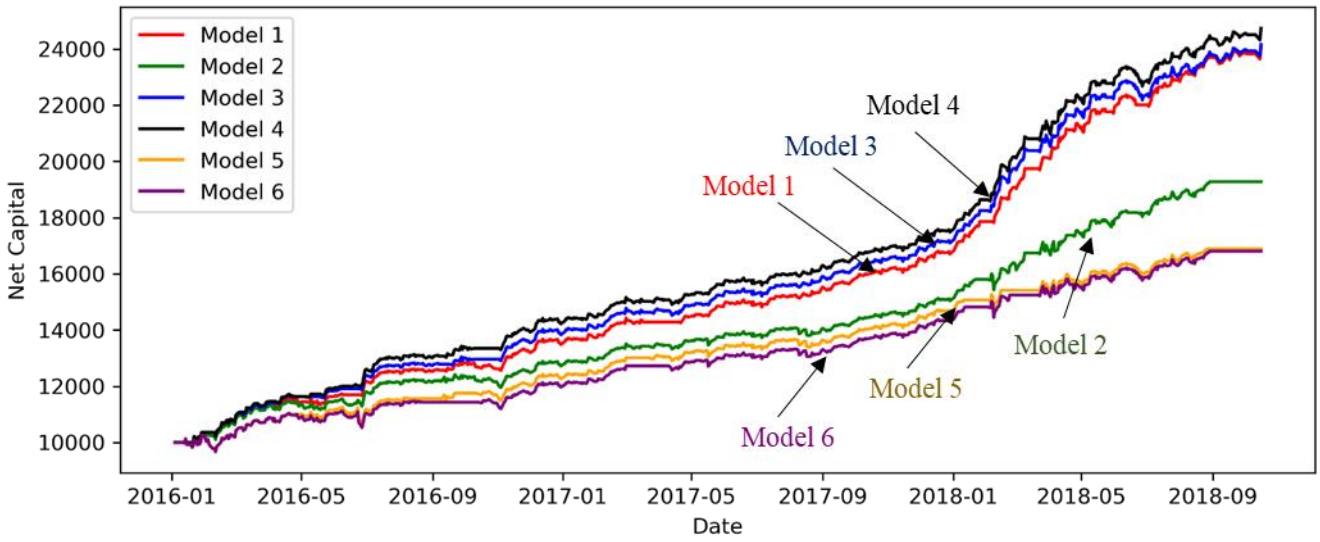


Fig. 6. Net Capital Growth for each Models

A possible reason our model was not able to perform well on the Nikkei 225 index is that it has the highest variance of returns compared to the other indexes. Table VII compares the standard deviation of each indexes. According to [36] , one advantage of Wavelet Neural Network is that it requires less training amount. With the limited amounts of training data in these 5 indexes, the Wavelet Neural Network was able to predict the stock's variance better than our model.

TABLE VI
RMSE ON DIFFERENT INDEXES

| Model | Stock Index | | | | |
|---|---|---|---|---|---|
| | **SSE** | **CSI** | **AORD** | **Nikkei** | **DJI** |
| Model 1 | 29.78 | 34.59 | 41.64 | 187.46 | 104.10 |
| Model 2 | 39.44 | 47.47 | 54.93 | 209.48 | 162.83 |
| Model 3 | 27.80 | 35.46 | 38.93 | 150.23 | 101.10 |
| Model 5 | 86.90 | 115.77 | 147.77 | 342.12 | 321.99 |
| [28] | 98.50 | 84.30 | 97.60 | 114.60 | 105.33 |

TABLE VII
STANDARD DEVIATION OF STOCK INDEXES

| Stock Index | Standard Deviation |
|---|---|
| SSE | 1.12% |
| CSI 300 | 1.34% |
| AORD | 0.72% |
| Nikkei 225 | 1.63% |
| DJI | 1.04% |

## VII. CONCLUSION

This paper proposed using DropConnect and a variant of Averaged Stochastic Gradient Descent on an LSTM model to reduce overfitting and increase accuracy. Additionally, the study also experimented with integrating both historical and textual data as input. This study experimented with 6 variants of the proposed model with different inputs, regulizers, and optimizer. The results show that the Model 4 performed better than the other models. Model 4 is an LSTM model with historical price and news input, DropConnect regulizer, and Adam optimizer. The model has the lowest RMSE and gained the highest profits in the market simulation.

DropConnect performed better than DropOut as a regulizer as it preserved more information while still simplifying the model. Adam optimizer is still more suitable for stock prediction than the NT-ASGD optimizer which was originally designed for language modeling. Adding news features was able to slightly increase performance. It is not as large as expected probably due to data and method limitations. When compared with a previous study using Wavelet Artificial Neural Network, Model 3 outperformed in 4 out of 5 indexes.

This paper has limitations namely the feature extraction technique for news is very simple due to time constraints in our experiments. In addition, the news headline was only used in text processing instead of the complete news. Further studies can also improve the model by tuning the hyperparameters of the training model automatically with evolutionary algorithms such as Genetic Algorithm.

## ACKNOWLEDGMENT

The authors would like to express a sincere gratitude to the anonymous reviewers for their valuable comments and suggestions to improve the quality of this manuscript. In addition, the authors would also like to thank Bina Nusantara University for supporting this research project.

## REFERENCES

[1] Damodaran, A. (2012). Investment valuation: Tools and techniques for determining the value of any asset (Vol. 666): John Wiley & Sons.

[2] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654-669.

[3] Minh, D. L., Sadeghi-Niaraki, A., Huy, H. D., Min, K., & Moon, H. (2018). Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access*, *6*, 55392-55404.

[4] Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

[5] Merity, S., Keskar, N. S., & Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

[6] Nair, B. B., & Mohandas, V. (2015). An intelligent recommender system for stock trading. *Intelligent Decision Technologies*, *9*(3), 243-269.

[7] Abe, M., & Nakayama, H. (2018). *Deep Learning for Forecasting Stock Returns in the Cross-Section.* Paper presented at the Pacific-Asia Conference on Knowledge Discovery and Data Mining.

[8] Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653-7670.

[9] Geva, T., & Zahavi, J. (2014). Empirical evaluation of an automated intraday stock recommendation system incorporating both market data and textual news. *Decision support systems*, *57*, 212-223.

[10] Paranjape-Voditel, P., & Deshpande, U. (2013). A stock market portfolio recommender system based on association rule mining. *Applied Soft Computing*, *13*(2), 1055-1063.

[11] Qeethara K. Al-Shayea, "Neural Networks to Predict Stock Market Price," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2017, 25-27 October, 2017, San Francisco, USA, pp371-377

[12] Zimbra, D., Abbasi, A., Zeng, D., & Chen, H. (2018). The state-of-the-art in Twitter sentiment analysis: a review and benchmark evaluation. *ACM Transactions on Management Information Systems (TMIS)*, *9*(2), 5.

[13] Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, e1253.

[14] Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). *Deep learning for event-driven stock prediction.* Paper presented at the Ijcai.

[15] Gottschlich, J., & Hinz, O. (2014). A decision support system for stock investment recommendations using collective wisdom. *Decision support systems*, *59*, 52-62.

[16] Qun Zhuge, Lingyu Xu, and Gaowei Zhang, "LSTM Neural Network with Emotional Analysis for Prediction of Stock Price," Engineering Letters, vol. 25, no.2, pp167-175, 2017

[17] Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). *Deep learning for stock prediction using numerical and textual information.* Paper presented at the Computer and

Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference.
[18] Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert systems with Applications*, *38*(5), 5311-5319.
[19] Mittal, A., & Goel, A. (2012). Stock prediction using twitter sentiment analysis. Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 15.
[20] Enke, D., & Mehdiyev, N. (2013). Stock market prediction using a combination of stepwise regression analysis, differential evolution-based fuzzy clustering, and a fuzzy inference neural network. *Intelligent Automation & Soft Computing*, *19*(4), 636-648.
[21] Kristjanpoller, W., Fadic, A., & Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications*, *41*(5), 2437-2442.
[22] Yu, H., Chen, R., & Zhang, G. (2014). A SVM stock selection model within PCA. *Procedia computer science*, *31*, 406-412.
[23] Nayak, R. K., Mishra, D., & Rath, A. K. (2015). A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices. *Applied Soft Computing*, *35*, 670-680.
[24] Nguyen, T. H., Shirai, K., & Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, *42*(24), 9603-9611.
[25] Chen, Y., & Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, *80*, 340-355.
[26] Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187-205.
[27] Chung, H., & Shin, K.-s. (2018). Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*, *10*(10), 3765.
[28] Lei, L. (2018). Wavelet neural network prediction method of stock price trend based on rough set attribute reduction. *Applied Soft Computing*, *62*, 923-932.
[29] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197-387.
[30] Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved From http://colah.github.io/posts/ 2015-08-Understanding-LSTMs/
[31] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
[32] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013, February). Regularization of neural networks using dropconnect. In *International conference on machine learning* (pp. 1058-1066).
[33] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
[34] Barnston, A. G. (1992). Correspondence among the correlation, RMSE, and Heidke forecast verification measures; refinement of the Heidke score. *Weather and Forecasting*, *7*(4), 699-709.
[35] Weng, Bin, Mohamed A. Ahmed, and Fadel M. Megahed. "Stock market one-day ahead movement prediction using disparate data sources." Expert Systems with Applications 79 (2017): 153-163.
[36] Wang, G., Guo, L., & Duan, H. (2013). Wavelet neural network using multiple wavelet functions in target threat assessment. *The Scientific World Journal*, 2013.

**Thayogo** was born in Jakarta, Indonesia on June 7, 1997. He has received his first degree of Economics with Finance concentration from University Pelita Harapan on 2014. He is currently pursuing a masters degree in Computer Science from Bina Nusantara University since 2018. Thayogo is currently working as a Programmer and Trainer for Data Science tools such as SAS, Python, and R at PT. Ganesha Cipta Informatika, Jakarta.

**Antoni Wibowo** (M'12) is a Member (M) of IAENG since 2012. He has received my first degree of Applied Mathematics in 1995 and master degree of Computer Science in 2000. In 2003, He awarded a Japanese Government Scholarship (Monbukagakusho) to attend Master and PhD programs at Systems and Information Engineering in University of Tsukuba-Japan. He completed the second master degree in 2006 and PhD degree in 2009, respectively. His PhD research focused on machine learning, operations research, multivariate statistical analysis and mathematical programming, especially in developing nonlinear robust regressions using statistical learning theory. He has worked from 1997 to 2010 as a researcher in the Agency for the Assessment and Application of Technology – Indonesia. From April 2010 – September 2014, he worked as a senior lecturer in the Department of Computer Science - Faculty of Computing, and a researcher in the Operation Business Intelligence (OBI) Research Group, Universiti Teknologi Malaysia (UTM) – Malaysia. From October 2014 – October 2016, he was an Associate Professor at Department of Decision Sciences, School of Quantitative Sciences in Universiti Utara Malaysia (UUM). Dr. Eng. Wibowo is currently working at Binus Graduate Program (Master in Computer Science) in Bina Nusantara University-Indonesia as a Specialist Lecturer and continues his research activities in machine learning, optimization, operations research, multivariate data analysis, data mining, computational intelligence and artificial intelligence.