Financial Trading with Deep Convolutional Neural Networks: Apple Stock

Gerend Christopher[1], Kevin Sean Hans Lopulalan[2]

[1,2]Institut Teknologi Bandung

Email: [1]gerendc@gmail.com, [2]kevinsean.lopulalan@gmail.com

Abstract

The application of computational technology in financial trading has grown significantly in popularity. Deep Learning, particularly through Convolutional Neural Networks (CNNs), has gained traction in image processing and is now being adapted for financial analysis. This study develops an algorithmic trading model for Apple stock data using a 2D CNN approach. Time-series data is transformed into 2D images by incorporating 14 technical indicators. Each indicator generates 16 days of sequential data, forming a 16×16 pixel image. These images are labeled as *Buy*, *Sell*, or *Hold* based on stock movement patterns.

Kata Kunci: *Deep Learning, Algorithmic Trading, Convolutional Neural Networks, Technical Analysis, Apple Stock*

I. Introduction

In today's era of globalization and increasing financial market complexity, a profound understanding of stocks, financial trading, and technical indicators has become essential for both academics and practitioners in finance. The stock market, one of the most actively traded financial instruments worldwide, remains a primary focus for investors, analysts, and researchers. To manage risk and optimize returns, market participants rely on diverse tools and strategies, including the use of technical indicators.

Technical indicators are analytical tools that assist traders in making buy/sell decisions. These indicators analyze historical price and trading volume data to identify market patterns, trends, and momentum. Commonly used technical indicators include the Relative Strength Index (RSI), Moving Averages (MA), and Money Flow Index.

However, as markets grow more complex and data availability expands, market participants are increasingly adopting advanced approaches like **algorithmic trading**.

Algorithmic trading employs computer algorithms to automate trading decisions based on predefined rules. Among these approaches, **deep learning** has gained significant traction. Deep Learning—a subset of machine learning—utilizes multi-layered artificial neural networks to process and interpret complex data. In stock trading, Deep Learning can identify intricate and abstract market patterns that may be imperceptible to human analysts or traditional methods. By leveraging Deep Learning, researchers and practitioners can develop sophisticated predictive models to forecast future stock movements or identify profitable trading opportunities. This approach integrates **technical analysis** with **artificial intelligence** to create more adaptive and efficient trading strategies.

The development of Deep Learning algorithms, such as *Convolutional Neural Networks (CNNs),* has been applied to stock trading systems. By incorporating technical indicators into CNNs, algorithmic trading in real-world markets demonstrates enhanced performance and effectiveness.

II. Methodology

II.1 Data

This study utilizes Apple Inc. stock data obtained from Yahoo! Finance. The time series data covers the period from January 1, 2000 to December 31, 2023. Figure 1 presents a sample of the stock price movement during this timeframe.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2000-01-03	0.936384	1.004464	0.907924	0.999442	0.846127	535796800
1	2000-01-04	0.966518	0.987723	0.903460	0.915179	0.774790	512377600
2	2000-01-05	0.926339	0.987165	0.919643	0.928571	0.786128	778321600
3	2000-01-06	0.947545	0.955357	0.848214	0.848214	0.718098	767972800
4	2000-01-07	0.861607	0.901786	0.852679	0.888393	0.752113	460734400
...	...	...	...	...	...	...	...
6032	2023-12-22	195.179993	195.410004	192.970001	193.600006	193.353287	37122800
6033	2023-12-26	193.610001	193.889999	192.830002	193.050003	192.803986	28919300
6034	2023-12-27	192.490005	193.500000	191.089996	193.149994	192.903839	48087700
6035	2023-12-28	194.139999	194.660004	193.169998	193.580002	193.333298	34049900
6036	2023-12-29	193.899994	194.399994	191.729996	192.529999	192.284637	42628800

Figure 1 AAPL Stock Price

The dataset contains 7 variables: *Date, Open, High, Low, Close, Adjusted Close*, and *Volume*, comprising 6,037 daily records from January 1, 2000 to December 31, 2023. Following preprocessing to derive actionable technical indicators, the data was partitioned into:

- 60% training data

- 20% validation data
- 20% testing data

II.2 *Technical Indicators*

This study employs 14 technical indicators, including:

1. Relative Strength Index (RSI)

   The Relative Strength Index (RSI) is a momentum oscillator that measures the historical strength and weakness of stock prices. The RSI value ranges between 0 and 100, where:

   - Values above 70 typically indicate overbought conditions
   - Values below 30 typically indicate oversold conditions

   The RSI is calculated as follows:

$$RSI_{step\ one} = 100 - \frac{100}{1 + \frac{average\ gain}{average\ loss}}$$

$$RSI_{step\ two} = 100 - \frac{100}{1 + \frac{(Previous\ Average\ Gain * period) + current\ gain}{(Previous\ Average\ Gain * period) + current\ loss}}$$

2. William %R

   Williams %R is a momentum-based technical analysis indicator that identifies overbought and oversold conditions in stock prices. The indicator oscillates between -100 and 0, where:

   - **Values below -80** typically indicate **oversold** conditions
   - **Values above -20** typically indicate **overbought** conditions

   The Williams %R is calculated as follows:

$$R = \frac{\max(\text{high}) - \text{close}}{\max(high) - \min(low)} * (-100)$$

3. Money Flow Index (MFI)

The Money Flow Index (MFI) is a technical indicator that combines price and volume data to identify overbought or oversold conditions in an asset. The MFI can also detect divergences that signal potential trend reversals.

$$Money\ Flow\ Index = 100 - \frac{100}{1 + Money\ Flow\ Ratio}$$

$$Money\ Flow\ Ratio = \frac{14\ Period\ Positive\ Money\ Flow}{14\ Period\ Negative\ Money\ Flow}$$

$$Raw\ Money\ Flow = Typical * Volume$$

$$Typical\ Price = \frac{High + Low + Close}{3}$$

4. Rate of Change (ROC)

   The Rate of Change (ROC) is a momentum oscillator that measures the percentage change in price between the current price and the price from a specified number of periods ago.

$$ROC = \frac{(Latest\ Close - Previous\ Close)}{Previous\ Close} * 100$$

5. Chaikin Money Flow (CMF)

   The Chaikin Money Flow (CMF) is a technical analysis indicator that measures the volume of money flow over a specific time period. The CMF value ranges between -1 and 1. A CMF value approaching -1 indicates increasing selling pressure, whereas a value approaching 1 signifies increasing buying pressure.

$$Multiplier = \frac{((Close - Low) - (High - Close))}{(High - Low)}$$

$$Money\ Flow\ Volume\ (MFV) = Volume * Multiplier$$

$$21\ Period\ CMF = \frac{21\ Period\ Sum\ of\ MFV}{21\ Period\ Sum\ of\ Volume}$$

6. Chande Momentum Oscillator (CMO)

   The Chande Momentum Oscillator (CMO) is a momentum-based technical indicator similar to the Relative Strength Index (RSI). The CMO ranges between -100 and 100. A CMO value above 50 is interpreted as the stock price being in the overbought region, while a value below -50 indicates that the stock is in the oversold region. Here, S_u represents the sum of momentum on up days, and S_d

represents the sum of momentum on down days. The formula for calculating the CMO is as follows:

$$CMO = 100 * \frac{S_u - S_d}{S_u + S_d}$$

7. Simple Moving Average (SMA)

   The Simple Moving Average (SMA) is the arithmetic mean of prices over a specified period. The formula for calculating the SMA is as follows:

$$SMA(M, n) = \frac{1}{n} \sum_{k=a+1}^{a+n} M(k)$$

8. Exponential Moving Average (EMA)

   The Exponential Moving Average (EMA) is a moving average of prices over a specified period that places greater emphasis on more recent data. In other words, the weights assigned to the most recent days are higher than those of the earlier days. The formula for calculating the EMA is as follows:

$$EMA(M, t, \tau) = \tau\, EMA(M, t-1, \tau) + (1-\tau)M(t)$$

9. Weighted Moving Average (WMA)

   The Weighted Moving Average (WMA) is a moving average of prices over a specified period that applies specific weights to each data point, similar to the Exponential Moving Average (EMA). While the EMA assigns exponentially decreasing weights, the WMA applies linearly decreasing weights. The formula for calculating the WMA is as follows:

$$WMA(M, n) = \frac{Sum\ of\ Weighted\ Averages}{Sum\ of\ Weight}$$

10. Hull Moving Average (HMA)

    The Hull Moving Average (HMA) is a type of moving average designed to reduce the lag typically associated with the SMA, EMA, and WMA. The formula for calculating the Hull Moving Average is as follows:

$$HMA(M, n) = WMA\left(\left(2 * WMA\left(\frac{n}{2}\right) - WMA(n)\right), sqrt(n)\right)$$

11. Triple Exponential Average (TRIX)

The Triple Exponential Average (TRIX) is an Exponential Moving Average (EMA) designed to filter out insignificant or unimportant price movements. The formula for calculating the Triple Exponential Average is as follows:

$$TRIX = \frac{EMA3(i) - EMA3(i-1)}{EMA3(i-1)}$$

$$EMA3(i) = EMA(EMA2, N, i)$$

$$EMA2(i) = EMA(EMA1, N, i)$$

$$EMA1(i) = EMA(Price, N, i)$$

12. Commodity Channel Index (CCI)

The Commodity Channel Index (CCI) is an indicator that compares the current price to the average price over a specified time period. Typically, CCI values range between -100 and 100. The formula for calculating the CCI is as follows:

$$CCI = \frac{Typical\ Price - 20\ Period\ SMA\ of\ TP}{.015 * Mean\ Deviation}$$

$$Typical\ Price\ (TP) = \frac{High + Low + Close}{3}$$

13. Detrended Price Oscillator (DPO)

The Detrended Price Oscillator (DPO) is a technical indicator used to remove price trends in an effort to estimate the length of price cycles from peak to peak or trough to trough. The formula for calculating the DPO is as follows:

$$DPO = Close\left(\frac{Period}{2} + 1\right) - SMA(Period)$$

14. Directional Movement Indicator (DMI)

The Directional Movement Indicator (DMI) is a technical indicator that measures the strength and direction of a trend. The DMI consists of three components: the Average Directional Index (ADX), the Plus Directional Indicator (+DI), and the Minus Directional Indicator (−DI). DMI values typically range between 0 and 100. The formula for calculating the DMI is as follows:

$$+DI = 100 * EMA\left(\frac{+DMI}{Average\ True\ Range}\right)$$

$$-DI = 100 * EMA\left(\frac{-DMI}{Average\ True\ Range}\right)$$

$$ADX = 100 * EMA\left(Absolute\ Value\ of\left(\frac{+DI - -DI}{+DI + -DI}\right)\right)$$

II.3 *Data Labelling*

Daily stock price data is manually labeled as Buy, Sell, or Hold based on peak and trough points within a *sliding window*. A Buy label is assigned if the price is at a local minimum, a Sell label is assigned if the price is at a local maximum, and the remaining points are labeled as Hold. Algorithm 1 illustrates the data labeling process.

```
Algorithm 1. Pelabelan Buy, Sell, atau Hold
Input: Data, kolom harga, dan windowSize
Output: label
Steps:
1.  While (counterRow < numberOfDaysInData)
2.       counterRow ++
3.       if (counterRow > windowSize)
4.             windowBeginIndex = counterRow – windowSize
5.             windowEndIndex = windowBeginIndex + windowSize – 1
6.             windowMiddleIndex = (windowBeginIndex + windowEndIndex)/2
7.             for (i = windowBeginIndex; i <= windowEndIndex; i++)
8.                   number = closePriceList.get(i)
9.                   if (number < min)
10.                        min = number
11.                        minIndex = closePriceList.indexOf(min)
12.                  if (number > max)
13.                        max = number
14.                        maxIndex = closePriceList.indexOf(max)
15.            if (maxIndex == windowMiddleIndex)
16.                  label = "Sell"
17.            elif (minIndex == windowMiddleIndex)
18.                  label = "Buy"
19.            else
20.                  label = "Hold"
```

II.3 Model

After the labels have been assigned, the stock data becomes labeled data with response and predictor variables, which can be modeled using machine learning. In this case, the type of machine learning to be modeled is *supervised learning*. Supervised learning trains the model using labeled data. This process involves two main phases: *training* and *predicting*. During the training phase, the model is trained to map the relationship between inputs and outputs so that it can make accurate predictions when given new, unseen data.

In the predicting phase, the trained model is used to make predictions on data that has not been previously encountered.

In this task, the model creation for stock prediction also involves supervised learning for regression, where the model is ultimately tasked with predicting a continuous value for the stock.

Furthermore, deep learning, a subfield of machine learning that uses multiple layers to understand complex data, will be employed. The deep learning architecture used in this task is the *Convolutional Neural Network* (CNN). CNN applies data partitioning into grids or matrices in deep learning. For instance, time series data can be considered as a 1D grid over specific time intervals, while image data is a 2D grid of pixels. In simple terms, a Convolutional Neural Network is a Neural Network that uses convolution instead of matrix multiplication at least in one of its layers.(Goodfellow, Bengio, & Courville, 2016). Adapun struktur dan komponen utama dalam CNN adalah sebagai berikut.

1. *Convolutional Layers*

   Convolutional layers consist of:

   a. *Filter* or *kernel*

      The filter is a small matrix with a fixed size, used to slide over or convolve the entire data in order to extract features from the data.

   b. *Stride*

      Stride is the number of pixels by which the filter moves at each step.

   c. *Padding*

      Padding refers to the additional pixels added to the edges of the data to control the size of the output.

2. *Activation Function*

   The activation functions to be used are ReLU for the hidden layers and softmax for the output layer. The ReLU and softmax functions are defined by the following equations

$$ReLU(x) = max(0, x) \quad \ldots(1)$$

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad \ldots(2)$$

3. *Pooling Layers*

In CNNs, the pooling layer is a component used to reduce the dimensionality of the feature map produced from the convolutional layer. The purpose of pooling is to reduce the number of parameters and computations in the network, while retaining important information in the *feature map*. The type of pooling used in this task is *max pooling*. Max pooling retains the maximum value from each block used in the pooling layer. This helps to downsample the feature map while preserving the most significant features from each region of the input data.

4. *Fully Connected Layers*

   Deep learning consists of components called *neurons* for each of its layers. The architecture of deep learning can vary greatly depending on the requirements of the task. One common type of architecture used in CNNs is *fully connected layers*. In this architecture, every neuron in the previous layer is connected to every neuron in the next layer. This fully connected structure allows the model to learn highly complex patterns in the data, which are then passed through an activation function suited to the data's needs. The primary task of fully connected layers is to perform high-level reasoning, such as classification. These layers help in decision-making by aggregating the learned features and transforming them into the final output.

5. *Dropout Layers*

   To prevent the phenomenon of overfitting, where the model becomes too specific to the training data, CNNs use a *dropout layer*. The dropout layer is a regularization technique in which neurons are randomly selected to be ignored during the training process. This means that the contribution of those neurons to the formation of the model is temporarily removed during *forward propagation*, and the updated weights will not be applied to those neurons during *backward propagation*.

II.6 Evaluasi Model

In the machine learning process, the *training phase* is where the machine learns the data along with its labels in order to build a model that captures the underlying patterns within the data. The *testing phase* is the stage where the quality of the model is evaluated to determine whether it has truly understood the fundamental structure of the data. In practice, the model needs to be evaluated using several specific metrics, which are as follows:

1. Accuracy

Accuracy is the percentage ratio used to measure how often the model makes correct predictions in alignment with the labels. The higher the accuracy, the better the model is at recognizing and correctly predicting the data.

2. Confusion matrix

   A confusion matrix provides an overview of the model's performance by showing how many correct and incorrect predictions were made for each class. A confusion matrix is generally represented as follows:

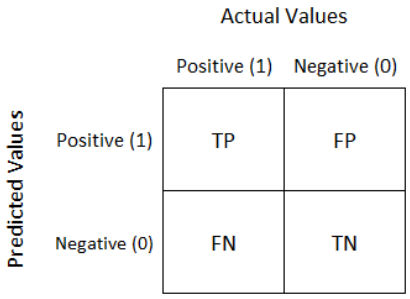

Figure 2 Confusion Matrix[1]

3. Skor F1

   The F1 score is the harmonic mean of *precision* and *recall*. F1 is a good measure of a model's accuracy when there is an imbalance in the number of samples between different classes. The formulas for precision and recall are as follows:

$$Precision = \frac{TP}{TP + FP} \quad \ldots(3)$$

$$Recall = \frac{TP}{TP + FN} \quad \ldots(4)$$

   di mana

   - TP (*true positive*) : the number of correct positive predictions
   - FP (*false positive*) : the number of incorrect positive predictions
   - FN(*false negative*) : the number of incorrect negative predictions

   From the two equations above, the F1 score can be formulated as follows:

$$F1\ Score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad \ldots(5)$$

[1] Narkhede, S. (2021, June 15). Understanding confusion matrix. Medium. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

The F1 score ranges between 0 and 1. The higher the score, the better the model's performance is considered. A high F1 score indicates that the model has a good balance between its ability to identify positive class instances (precision) and its ability to find all the positive class instances (recall). Additionally, a high F1 score suggests that the model is effectively handling both false positives and false negatives, making it particularly useful in situations where class imbalance exists.

4. *Cohen's Kappa*

   Cohen's Kappa is a statistical measure used to evaluate how well two different categorical data sets agree. It is calculated using the observed agreement (P_0) and the expected agreement (P_e) in the following formula:

$$\kappa = \frac{P_0 - P_e}{1 - P_e} \quad \ldots(6)$$

   di mana

   - P_0: is the observed agreement, which is the proportion of times the raters (or models) agree.
   - P_e: is the expected agreement, which is the proportion of times the raters (or models) would be expected to agree by chance.

   Cohen's Kappa ranges from -1 to 1, where:

   - A value of 1 indicates perfect agreement.
   - A value of 0 indicates no agreement beyond chance.
   - A value less than 0 indicates less agreement than expected by chance.

III. Result

From the daily stock price data of AAPL, indicators are applied to process the adjusted close data with the goal of creating images that can be used in CNN. The calculation of these indicators helps the model better recognize patterns in the stock prices. Then, in section II.3 **Data Labeling**, the data will be labeled with:

- 0 : *Sell*
- 1: *Buy*
- 2: *Hold*

After that, the data will undergo preprocessing to generate new variables derived from the technical indicators. A snippet of the data is shown in **Figure 3**.

	Date	Open	High	Low	Close	Adj Close	Volume	rsi_6	rsi_7	rsi_8	...	dmi_15	dmi_16	dmi_17	dmi_18	dmi_19	dmi_20	dmi_21	dmi_22	dmi_23	Labels
30	2000-02-15	1.029018	1.070871	1.028460	1.062500	0.899512	485744000	72.696038	76.835396	79.793674	...	34.590150	36.285777	37.939623	39.545141	41.098363	42.597202	44.040927	45.429768	46.764635	0
31	2000-02-16	1.051339	1.054688	1.001116	1.018973	0.862662	378246400	48.159258	58.655606	64.345229	...	34.054820	35.653502	37.227624	38.767524	40.266775	41.721319	43.128830	44.488239	45.799386	2
32	2000-02-17	1.028460	1.031250	1.010045	1.025670	0.868332	289497600	50.313836	51.898792	61.359582	...	33.706805	35.196867	36.680701	38.145393	39.581938	40.984206	42.348190	43.671449	44.952689	2
33	2000-02-18	1.023438	1.030134	0.989955	0.993304	0.840931	233441600	45.938282	42.442376	44.199185	...	33.381991	34.768772	36.165950	37.557826	38.933145	40.283948	41.604724	42.891786	44.142805	2
34	2000-02-22	0.983259	1.044085	0.952567	1.016183	0.860300	422296000	51.509486	52.871429	48.850815	...	32.626843	33.967725	35.324437	36.681188	38.026413	39.351744	40.651227	41.920735	43.157519	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6032	2023-12-22	195.179993	195.410004	192.970001	193.600006	193.353287	37122800	17.202025	43.670309	51.136163	...	20.569973	19.334348	18.231729	17.245871	16.379213	15.640682	14.972541	14.364833	13.809174	0
6033	2023-12-26	193.610001	193.889999	192.830002	193.050003	192.803986	28919300	14.443066	16.212729	41.828941	...	20.953421	19.713465	18.599896	17.598986	16.714012	15.954134	15.264906	14.636839	14.061832	0
6034	2023-12-27	192.490005	193.500000	191.089996	193.149994	192.903839	48087700	17.358911	16.032872	17.702440	...	21.311306	20.068887	18.946407	17.932484	17.031190	16.251914	15.543349	14.896481	14.303505	0
6035	2023-12-28	194.139999	194.660004	193.169998	193.580002	193.333298	34049900	29.985646	22.870982	21.175828	...	20.455621	19.375184	18.377922	17.461660	16.636358	15.915457	15.253537	14.644279	14.081905	0
6036	2023-12-29	193.899994	194.399994	191.729996	192.529999	192.284637	42628800	9.330794	23.599730	18.922148	...	19.656982	18.724836	17.842877	17.016993	16.262307	15.595823	14.977527	14.403542	13.869940	0

Figure 3 The data snippet after preprocessing.

The class distribution is shown as follows.

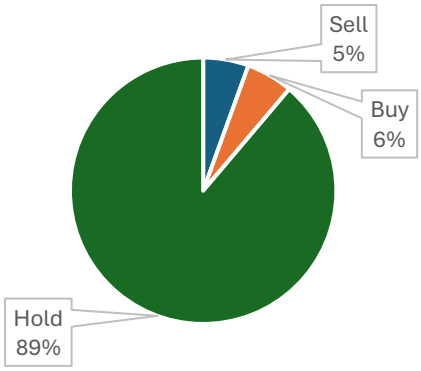

Figure 4 Class Distribution

The new data will then be processed using CNN. First, the model is given input data of size (16×16). The model then applies 2 convolutional layers. The first convolutional layer, with a filter size of (3×3) and a stride of 1, produces an array of size (14×14×32). This is followed by a dropout layer of 20%. The second convolutional layer, with a filter size of (3×3) and a stride of 1, produces an array of size (12×12×64). This is followed by a dropout layer of 30%. **Figure 5** shows a sample representation of the CNN output.

Next, the model proceeds with the fully connected layer, which consists of 2 layers and 2 dropout layers. The first layer has 128 neurons with a ReLU activation function, followed by a 20% dropout, and the second layer has 64 neurons with a ReLU activation function, followed by a 20% dropout. Finally, the output layer is configured with a size of 3 and a softmax activation function. The complete CNN architecture can be seen in **Figure 6**.

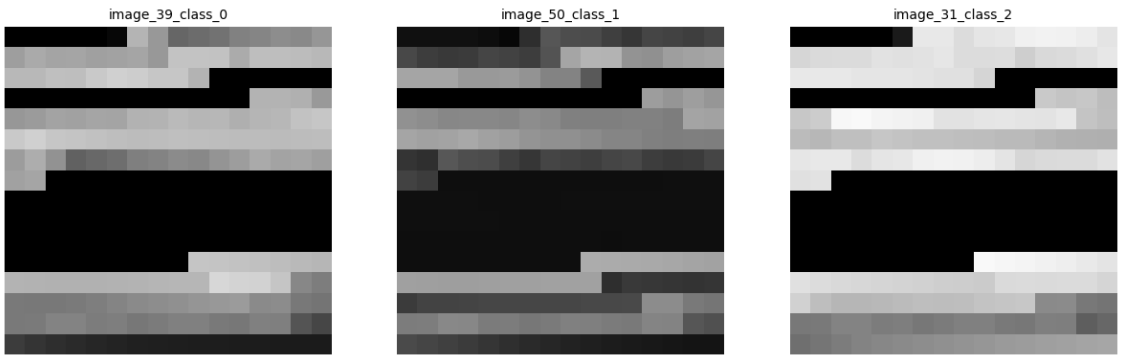

Figure 5 Sample representation of CNN in a 16 x 16 pixel image. From left to right: Sell, Buy, Hold.

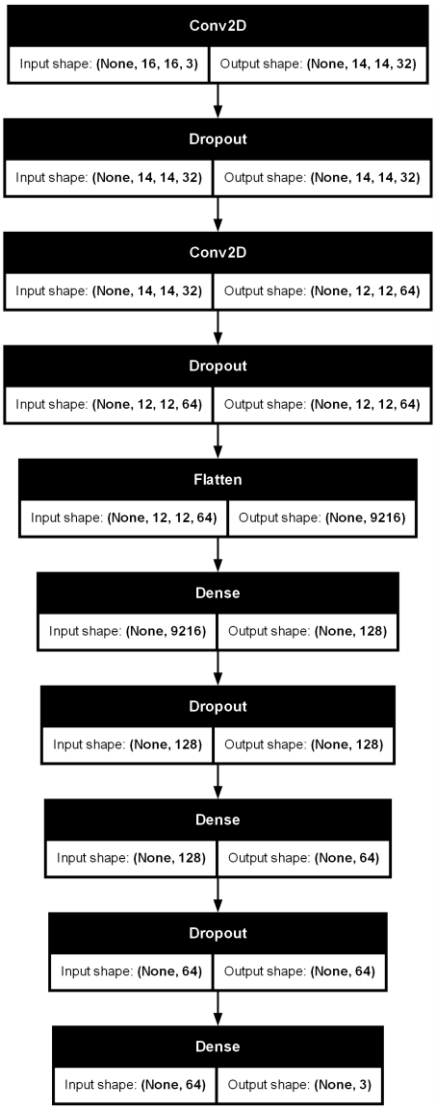

Figure 6 CNN Architecture

IV. Discussion

The simulation was conducted with a batch size of 80, 400 epochs, and a learning rate of 0.001, using the Adam optimization algorithm. Training was performed on the training data, and the cross-validation (CV) data showed that the model was able to learn patterns

in the data with a training accuracy of 0.8514 and a validation accuracy of 0.6664. The training loss at the final epoch was 0.2136, while the validation loss was 1.1462. Finally, the F1 score for training was 0.8514, and for validation, it was 0.6652. The plot of the training process can be seen in **Figure 7**.

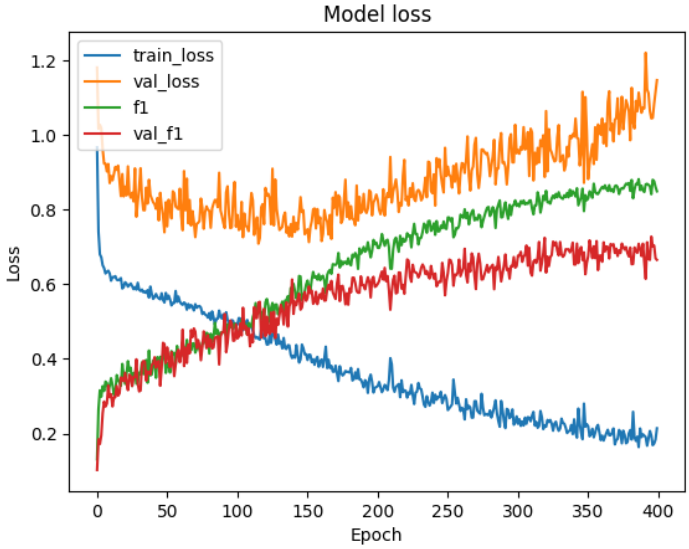

Figure 7 Data Training Plot

From the training results, the confusion matrix is obtained as follows.

Tabel 1 Confusion Matrix onto the Test Set

	Prediction			
Actual		*Sell*	*Buy*	*Hold*
	Sell	15	2	49
	Buy	0	37	31
	Hold	146	170	752

Tabel 2 Evaluation onto the Test Set

Total Akurasi: 0.67			
	Sell	Buy	Hold
Recall	0.23	0.54	0.70
Precision	0.09	0.18	0.90
F1-Score	0.13	0.27	0.79

Note that in **Figure 7**, the learning curve, including both accuracy and loss, is fluctuating. However, as the number of epochs increases, there is a trend for each metric. As the epochs increase, both the training accuracy and validation accuracy of the model continue to

improve. However, as the number of epochs increases, the gap between training accuracy and validation accuracy also widens. On the other hand, as the number of epochs increases, the training loss decreases, but the validation loss does not follow the same trend and actually increases as the epochs grow. These two phenomena indicate that the model has overfitted. One explanation for this is that although we have used weights for model fitting, the stock data is highly imbalanced, as shown in **Figure 4**. The significant imbalance in the data is one of the factors contributing to overfitting. This imbalance has the implication that the minority class data is more likely to contain more noise relative to the larger class, which often goes unnoticed during the model's computation.

V. Epilog

V.1 Conclusion

In this study, the author built a Convolutional Neural Network (CNN) model for predicting stock price movements using technical indicators. From the simulation, it was concluded that the CNN model can be used to predict stock prices with a relatively high accuracy. However, stock data is practical data in the field of economics that is difficult to predict. One of the impacts is the imbalance in the stock data for the "sell", "buy", and "hold" behaviors. This causes the CNN model created to overfit the stock data.

V.2 Suggestion

This modeling is performed for the highly imbalanced data distribution for the "sell", "buy", and "hold" labels. While this approach may be quite practical in daily life, it can lead to overfitting of the model. One way to address this issue is to explore methods that prevent overfitting, such as augmenting the data to make the minority class more balanced in terms of quantity.

VI. Reference

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Narkhede, S. (2021, June 15). *Understanding confusion matrix*. Medium. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Nayak, A. (2020). Stock trading with CNNS: Time series to image conversion. Medium. https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Applied Soft Computing, 70, 525-538.

VII. Contribution

PIC	Contribution
Gerend Christopher	• Writing Python code • Running model simulation • Writing abstract, introduction, and methodology • Creating part of the Presentation Deck
Kevin Sean Hans Lopulalan	• Writing Python code • Running model simulation and hyperparameter tuning • Writing part of the methodology • Analyzing results and discussion • Creating part of the Presentation Deck