Techincal Analysis Pattern Recognition Using Convolutional Neural Networks

TREVOR MARTIN ROCHESTER INSTITUTE OF TECHNOLOGY

INDUSTRIAL AND SYSTEMS ENGINEERING ROCHESTER, NY TMM4038@RIT.EDU

Abstract—Technical analysis is a method of using candlestick chart patterns to attempt to predict the price movement of a tradable asset, such as a stock or cryptocurrency. Many traders identify patterns manually due to the subjective nature of technical analysis. Using convolutional neural networks, the manual subjective pattern recognition process can be simulated and would be more generalized than hard coded or mathematical methods. The developed model identifies patterns with a 66% accuracy when compared to the hard-coded method. While this accuracy is not high enough to measure the level of generalization, this model could be used as a stock screener using automated pattern recognition.

I. INTRODUCTION

Candlestick patterns have been used for centuries to predict price direction *(Morris, 2006)*. Since their introduction into Western culture in 1991, there have been studies on the effectiveness of candlestick patterns of predicting future price trends. Most notably, a study by Bulkowski found that candlestick patterns can predict price movement between 50 and 60% of the time *(Bulkowski, 2012)*. 50% is essentially random, but there are some specific patterns Bulkowski highlights, such as the bullish engulfing and bullish harami, that perform better than random, at 58% and 57% respectively. Both patterns are bullish continuation signals, meaning that they occur during an uptrend, and indicate the uptrend will continue. The performance measure is how often the uptrend continues after the pattern has been identified. Bulkowski is a prominent figure in a group of traders who base their trades primarily on candlestick patterns, which is a trading method known as Technical Analysis.

The goal of this project is to use machine vision to simulate the manual detection of candlestick patterns. Currently, it is possible to recognize these patterns using hardcoded methods, but technical analysis can be subjective, so these models do not always capture every occurrence of a pattern *(Lo, Mamaysky, & Wang, 2000)*. This subjectivity is based on an individual's visual interpretation of the chart patterns and inspired this project. The Convolutional Neural Network (CNN) derives from the process that occurs in the human eye to interpret visual signals *(Fukushima, 1980)*. Since their inception CNN's have been the top choice for many machine vision problems.

There is a lot of existing research using CNNs for pattern recognition, with a subset of that research being in the field of stock trading. CNNs and long short-term memory (LSTM) are regularly combined as they have some complimentary characteristics. CNNs are useful for extracting effective features from image data, while LSTM is useful for finding interdependence in time series data. A group of researchers used a CNN-LSTM model to predict the stock closing price on the next day *(Lu et al., 2020)*. This research used R^2 as a success measure and produced a value of 0.9646. This CNN-LSTM model was the best of the models analyzed and was recommend for use when forecasting next day closing prices.

Much of the work in this paper builds off research around using deep learning to recognize stock chart patterns *(Velay & Daniel, 2018)*. This research utilized both CNN and LSTM models to detect chart patterns on both candlestick charts and curve charts. The LSTM model produced 97% accuracy, while the CNN produced 73% accuracy using the candlestick charts. The CNN results with the candlestick charts were 3% better than the curve charts. The previous research looked at charts with 30-minute candlesticks, as opposed to daily candlesticks. It also looked at longer term patterns such as bearish flags and double bottoms, which can be 20+ candles long. Achieving similar CNN accuracy on daily candlesticks with the highlighted two-candle patterns of bullish engulfing and bullish harami would be a success. However, it is mentioned that the accuracy of 73% is not enough to compare the generalization of the model to the hard-coded method, which is something to keep in mind.

II. METHODOLOGY

To conduct the analysis, the first necessity was stock market data. Four stock tickers were used in this analysis in order to provide a sufficient amount of data. These tickers are AAPL, F, MSFT, and SPY. The data for these tickers was collected using Wall Street Journal historical price data, and contains the open, high, low, and close (OHLC) prices for every day from January 1st, 1997, to December 31st, 2020. All four tickers have been trading since before 1997, and there are 6041 trading days in the chosen interval. With four tickers, this gives 24164 days of OHLC data. This is smaller than the MNIST handwritten integers dataset that is used for an introductory to CNNs, which contains 60000 images. However, the 24164 days should be a large enough dataset to work well with.

Candlestick charts were generated from the OHLC values using matplotlib. An example is provided in *Figure 1*. When the close price is above the open price the candle is green, and when the close price is below the open price the candle is red. The charts have 10 candles on them, which represent the open and close prices for 10 days. Charts were created for all four stocks starting at each day in the interval, which resulted in the creation of 24120 charts. This was done to both lengthen the dataset as well as to assist the CNN in learning patterns invariable of location. For example, if the pattern was always on the left, the CNN would only expect the pattern to be on the left, when it is possible for the pattern to be present anywhere on the chart.

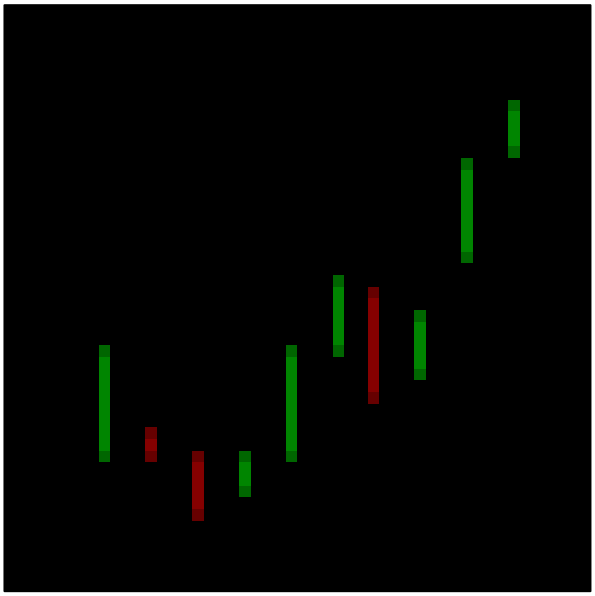

Figure 1: Candlestick Chart Example

The candlestick charts created were intentionally made to be low resolution with a black background to limit both the file size and complexity of the image. Limiting the file size was helpful when storing 24120 images, but the more important piece was limiting the complexity. Standard practice when using CNNs is to manipulate the input image data to simplify it without losing key features that make the images identifiable. One popular example is using CNNs to identify cats and dogs. In this example, instead of heavily reducing the resolution of the image, the image is input in grayscale to reduce complexity since cats and dogs can be discerned in grayscale nearly as easily as in full color. For the candlestick charts, grayscale does not work because the red and green candles are important identifiers for the pattern. However, the red and green candles are still easily identified in a low-resolution image, so that solution works nicely for this use case. It is also important to note that the black background helps to simplify the image for input into the CNN. The 'border' around the chart is constant as well, meaning the 10 candles will always appear centered both horizontally and vertically on the image. There will never be a candle near the edge of an image, so the model will always know where to look. This concept is well discussed in the MNIST introduction to CNNs, as the handwritten integers are centered and oriented properly to minimize the variation in model inputs.

With the candlestick charts created and ready to use in the CNN as the features, the next requirement is labels. For this, the images were classified into four groups: bullish engulfing *(Figure 2)*, bullish harami *(Figure 3)*, both patterns, and no pattern. The pattern presence was determined using a hard coded method based on the mathematical definitions of these patterns *(Morris, 2006)*. This is the simplest way to label 24120 days of data and is a standard practice for implementing the known truth in technical analysis *(Velay & Daniel, 2018; see also Lu et al., 2020)*. One issue with this method is that there is not a known truth when it comes to technical analysis.

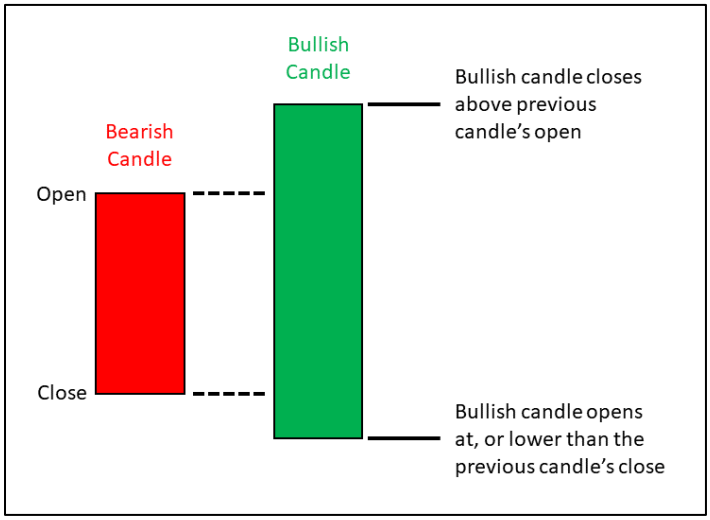

Figure 2: Bullish Engulfing Pattern Description

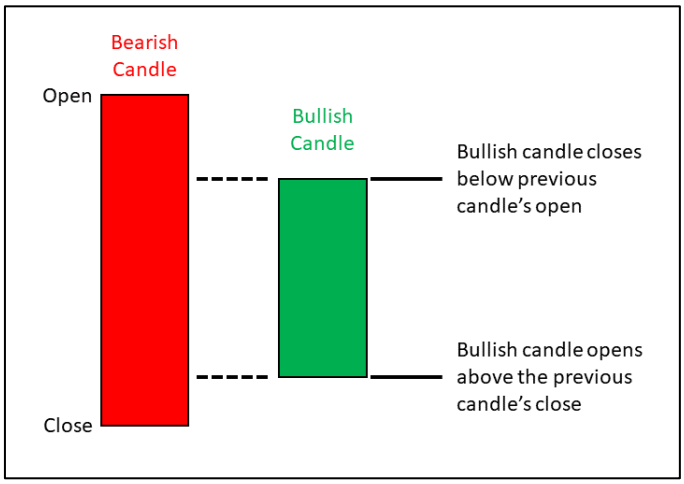

Figure 3: Bullish Harami Pattern Description

Once the features and response were prepared, the data was shuffled, and then split into a training set and a validation set. 80% of the data was used for training, with the other 20% left for validation. In both the training and validation sets, there were approximately 48% of images containing no pattern, 20% of images containing a bullish engulfing, 26% of images containing a bullish harami, and 6% of images containing both patterns. This is a sufficient distribution of data between classifiers, as the CNN will not be able obtain a high accuracy by guessing there is no pattern for every image. It is also important the both the training and validation sets contain close to the same proportions of each identifier, as it would be undesirable for the model to learn that approximately 20% of the images contain a bullish engulfing, and then when presented with new data that is 35% for example.

The creation of a CNN involves making some decisions about architecture. The main building block of a CNN is a convolutional layer, which consists of filters that help the CNN learn. The network learns filters that activate when a specific type of feature is detected in a portion of the input *(Géron, 2017)*. In this case, the CNN uses filters to learn if a pattern is detected on a specific part of the chart. After each convolutional layer, an activation function is used, and most commonly this is the ReLU function. The ReLU effectively removes negative values from an activation map by setting them to zero *(Romanuke, 2017)*. This adds non-linearity to the decision function which helps improve network accuracy.

Another element of CNN architecture is the pooling layer. Pooling is a form of non-linear down-sampling, which helps with generalization. To the CNN, the exact location of a feature is less important than the rough location relative to other features. Max pooling is the most common type of pooling, and as with all types of pooling, helps reduce the number of parameters *(Figure 4)*. This reduction of parameters also helps combat overfitting. After some combination of convolutional and pooling layers, a CNN will have a flattened, fully connected layer, as present in a typical artificial neural network. Finally, there is a loss layer which penalizes the model for incorrect identifications.

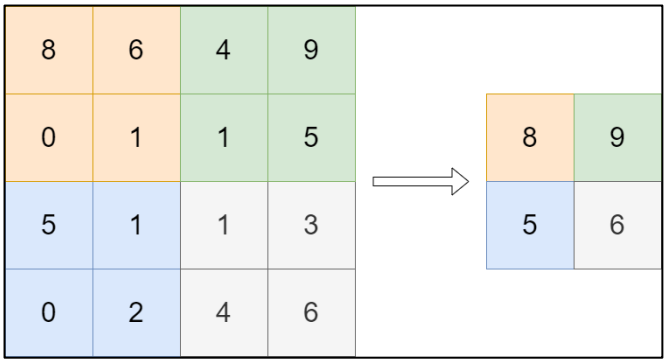

Figure 4: Max Pooling Example

The CNN architecture used for this model can be seen in *Figure 5*. This model uses a convolutional layer, followed by a pooling layer. The pooling layers use max pooling with a 2x2 filter and a stride of 2, as shown in the example in *Figure 4*.

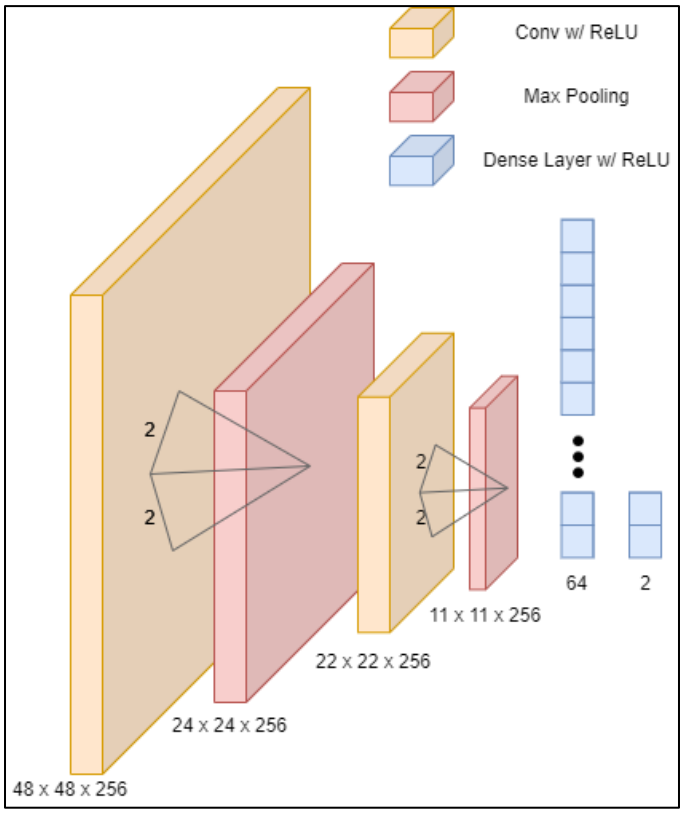

Figure 5: CNN Architecture Used

This combination of convolutional layer and pooling layer repeats, and then is flattened into a dense layer, and then finally a loss layer. This loss layer uses the Sigmoid activation function because it is the most useful for multi-

class classification. The input of this model is the candlestick charts, as well as the label (bullish engulfing [1,0], bullish harami [0,1], both patterns [1,1], and no pattern [0,0]). Since this model uses the sigmoid loss function, the output of this model is 2 probability values for each of the patterns. If for a single chart the model outputs [0.05, 0.82], it would mean that the model is 5% confident there is a bullish engulfing, and 82% confident there is a bullish harami present in the image. The chart would be classified as bullish harami. If the actual label is [0,1], this would be a success, and if it is anything else, it would be a misclassification. The measure of success of this model is the accuracy at which it classifies the images, as a percentage of correctly classified images vs all observed images.

This model uses the Adam optimizer, a learning rate of 0.001, a beta 1 decay rate of 0.9, and a beta 2 decay rate of 0.999. These are the default hyperparameters associated with the Adam optimizer and are recommended for use to those who are new to implementing CNNs.

III. Results and Discussion

The results of this model nearly met the established hypothesis of 73% accuracy derived from previous research. With 10 epochs, the training accuracy of the model was 69.84%, with a validation accuracy of 64.32% *(Figure 6)*. This falls slightly short of the desired 73% but shows some promise. Some tweaking of the hyperparameters of this model such as the optimizer, learning rate, and decay rates could potentially improve this model to meet or exceed the 73% accuracy shown in prior work.

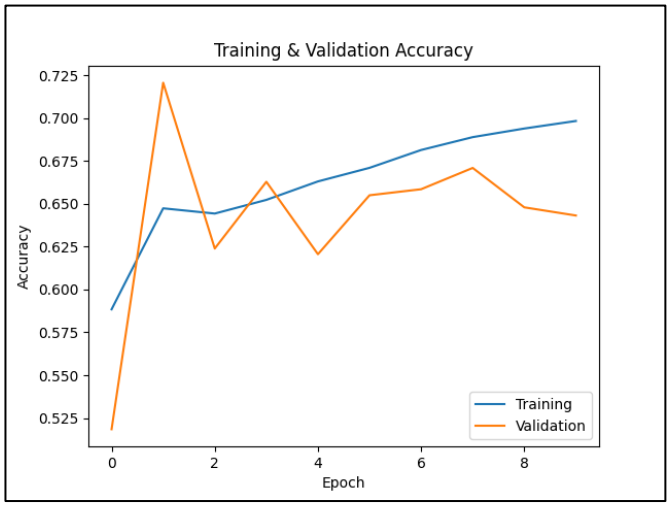

Figure 6: Training Accuracy

Over the course of 10 epochs, the accuracy of both the training and validation sets are somewhat correlated. The same cannot be said for the loss. The training and validation loss values begin to diverge after 2 or 3 epochs *(Figure 7)*. This is a classic sign of overfitting. The training loss is being minimized by the model, and it is beginning to memorize the training data. This leads to a decrease in the training loss, but an increase in validation loss. Overfitting prohibits the model from accurately predicting new data and should be avoided or corrected.

Figure 7: Training Loss

There are a few ways to counteract overfitting when working with CNNs. The easiest method is to look at the model when validation loss is the lowest, which is usually where the training and test loss diverges. In this case, that is with 5 epochs and a validation loss of 0.391. At this point, the model has a training accuracy of 65.24 % and a validation accuracy of 66.29%. This model does a better job of generalizing the data, with a better validation accuracy than the model at 10 epochs. Another way to counteract overfitting would be to introduce dropout layers to the model, which effectively nullify the contribution of some neurons towards the next layer at random. This prevents all neurons in a layer from synchronously optimizing their weights. Dropout layers are frequently used with small datasets since they are easier for a neural network to memorize but they are also useful with large datasets such as this one to combat overfitting.

A difficult part of interpreting the results of this model is that the accuracy is compared to the hard coded, mathematically identified pattern. Ideally, the network would be able to learn from a dataset of patterns identified by a group of renowned technical analysis traders. This would allow the model to learn to identify patterns that do not perfectly fit the mathematical definition, but still classify as an example of a pattern in the eyes of traders. However, a perfect dataset like this does not exist, so the hardcoded method is the next best choice.

The lack of a perfect dataset also limited the scope of this project, as there are more historically successful

patterns than those that were analyzed. Bulkowski has a rank of small patterns with respect to upward breakout performance, and bullish engulfing and bullish harami rank 6th and 10th on that list respectively. However, these two patterns are relatively simple to identify with a hard coded method compared to the top performing pattern, downside weekly reversal. This pattern looks at weekly charts and is much more subjective in nature than the bullish harami and bullish engulfing. This type of pattern highlights the impact of the lack of a ground truth when it comes to technical analysis pattern recognition.

IV. Conclusion

The results of this modeling process are promising but are a few improvements away from being useful enough to use for technical analysis trading. With an accuracy of less than 70%, this model produces large quantities of both false negatives and false positives. In the case of a trader, false negatives are missed trading opportunities. False positives waste a trader's time looking for a pattern that is not there or waste a traders money trading on a bullish pattern that was not truly present. One improvement to this model would be to minimize false positives by penalizing the model for incorrectly predicting a pattern when it is not present.

This large number of false negatives and false positives also makes it difficult to quantify the ability of this model to generalize compared to the hard coded method. With a smaller quantity of false positives, they could be individually inspected. It is possible that the model is identifying some instances of patterns that are not captured using mathematical methods, but there is no way to know for sure with so many false positives. This would be a high priority improvement as this generalization is a key reason that one would use CNNs for this task.

Another area for improvement with this model is to introduce the overfitting counteraction measures that were previously discussed. Adding dropout layers would improve the model's ability to generalize and reduce the memorization of training data that is taking place. This would improve the validation loss and allow for more epochs to take place before the model begins detrimentally overfitting.

A possible next step for this research would be to investigate the model's ability to identify other technical analysis patterns. Some tweaking would be needed to incorporate new patterns depending on the length of the pattern, but otherwise the model would most likely have similar or slightly worse performance. One issue with other patterns is that they are far less prevalent than the patterns used in this research. The most effective bearish pattern according to Bulkowski, called Three Black Crows, had 3 occurrences in the selected stocks since 1997. This extremely small sample size would most likely cause the model to never predict the presence of this pattern. To incorporate many different patterns, some data manipulation may be necessary to create enough training data to have success.

Similarly, another step would be to use this model to identify patterns with different candle frequencies. Some traders will look at the charts with weekly candlesticks, daily candlesticks (used in this paper), and hourly candlesticks. This would just require access to the data at different frequencies and would be as simple as changing the data source. The model input would still be the 10-candle chart, so the transition should theoretically be seamless, but would need further testing to verify. If this model were to have similar performance on other candlestick frequencies, it could be used by all types of traders, from swing traders to day traders.

If this model were improved enough to be trusted and identify patterns that are not found using hard coded methods, there would be many use cases for stock traders. The simplest use case would be to use this model as a stock screener to identify possible trades. The trader could input as many tickers as they are interested in trading, and the model would generate charts and be able to identify bullish or bearish patterns. The relevant charts could be displayed to the trader for them to make the final trading decision.

With enough trust in the model and technical analysis patterns, this ideal model could be used in a framework that automatically triggers trades. Ideally that framework includes some form of sentiment analysis to capture news, as well as an LSTM model to work with forecasting the raw time series data. If the patterns from the CNN, forecast from the LSTM, and output from the sentiment analysis are all bullish, a purchase order could be placed.

This project could be utilized for a variety of applications if the current shortcomings are addressed. This would require further research into limiting the overfitting, and would require higher quality input data, which may also have something to do with the overfitting.

V. WORKS CITED

Bulkowski, T. N. (2012). Encyclopedia of Candlestick Charts. Germany: Wiley.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193–202.

Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. United States: O'Reilly Media.

Lo, A.W., Mamaysky, H. and Wang, J. (2000), Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. The Journal of Finance, 55: 1705-1765.

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-Based Model to Forecast Stock Prices. Complexity, 2020, 1–10.

Morris, G. L. (2006). Candlestick Charting Explained: Timeless Techniques for Trading Stocks and Sutures. Ukraine: McGraw-Hill Education.

Romanuke, V. (2017). Appropriate Number and Allocation of RELUS in Convolutional Neural Networks. Research Bulletin of the National Technical University of Ukraine “Kyiv Politechnic Institute,” 0(1), 69–78.

Velay, M., & Daniel, F. (2018). Stock Chart Pattern recognition with Deep Learning. Artificial Intelligence Department of Lusis.