 doi:10.20944/preprints202104.0515.v2

Convoluted Stock Market: Using Convolution Operation for Stock Market Prediction

Yigit Alparslan, Ethan Jacob Moyer, and Edward Kim

Drexel University, Philadelphia PA 19104, USA
{ya332, ejm374, ek826}@drexel.edu

Abstract. Many studies in the current literature annotate patterns in stock prices and use computer vision models to learn and recognize these patterns from stock price-action chart images. Additionally, current literature also use Long Short-Term Memory Networks to predict prices from continuous dollar amount data. In this study, we combine the two techniques. We annotate the consolidation breakouts for a given stock price data, and we use continuous stock price data to predict consolidation breakouts. Unlike computer vision models that look at the image of a stock price action, we explore using the convolution operation on raw dollar values to predict consolidation breakouts under a supervised learning problem setting. Unlike LSTMs that predict stock prices given continuous stock data, we use the continuous stock data to classify a given price window as breakout or not. Finally, we do a regularization study to see the effect of L_1, L_2, and *ElasticNet* regularization. We hope that combining regression and classification shed more light on stock market prediction studies.

1 Introduction

In a developed country with a free-market economy, the stock market is at the forefront of its economic growth and prosperity. Generally, stock markets are composed of buyers and sellers that trade ownership of publicly listed companies, also known as shares, during its hours of operation [20] [19]. The ability to predict trends in the stock market yield large gains as an analyst can anticipate when a stock will fall and they should sell or when a stock will rise and they should buy [19]. Even this simple classification is difficult to make as the stock market is terribly complex and seemingly unpredictable.

The complexities of a stock market are rooted in the many actions that can be undertaken at any given moment. All of these actions are influenced by often untraceable, immeasurable factors [12]. Since a representative feature set is imperative for a well-defined data science problem (i.e. predicting whether a stock will rise or fall), researchers anticipate difficultly in exploring stock market

All code is open-sourced on GitHub.

 doi:10.20944/preprints202104.0515.v2

prediction algorithms [14]. All is not lost as patterns tend to reoccur over the history of a stock [16]. For this reason, the most common metrics for analyzing stock patterns are past and present stock prices. In some cases volume of a stock, or the number of transactions at any given time, is used as a metric because large volumes often correlate with large shifts in stock behavior [3]. However, there is little evidence to support that this metric improves predictions [21].

While predicting raw stock prices is challenging, it may be more effective to instead predict certain well-defined annotated stock patterns in the stock market. One such pattern is a consolidation breakout.

Although the stock market is criticized at times to be an inaccurate measure of economic growth, long term trends of positive stock growth are often correlated with positive economic growth [2]. Overreaction is one of the major criticisms of the stock market and its trends [7]. Therefore, a buy-and-hold strategy of investing, one in which a certain amount of stock share is purchased in the past and not sold until the present, will be our comparative baseline.

We hope that our study sheds more light into the field of the stock market prediction to outperform a buying and holding strategy.

This research is organized so that section 1 introduces the concept of stock predictions and section 2 explores what has been done in this field. section 3 explains the dataset and the models in this study. We report the results in section 4 and conclude the study in section 5 and section 6 with summarizing what we have done in this study and discussing where the research might go in the future.

2 Related Work

Neural networks are among one of the most common techniques used to form predictions on the stock market. More specifically, convolution neural networks and Long Short-Term Memory (LSTM) networks are among of the most popular neural network architectures employed in stock market predictions [11] [5]. Convolution is especially interesting to researchers as it can recognize relationships in the data often unapparent through conventional data analysis [6]. These networks are composed of an input layer, an output layer, and multiple hidden layers. On the other hand, LSTM networks are attractive because of their ability to recognize long-term trends. By using internal units, each with an input gate, an output gate, and a forget gate, LSTM network can store states in the data until they are no longer required for prediction [10]. These models have been used in combination for stock price and stock direction prediction in which both a CNN and an LSTM networks have been applied in a generalized adversarial network (GAN) [22].

In literature, there is no short supply of stock market prediction algorithms. [13][1]. In this paper, we adopt a gradient descent algorithm with a convolution objective function to predict stock pattern and behavior on raw stock price data instead of pattern matching on the stock chart images. We combine regression through convolution and classification through gradient learning. We hope this

 doi:10.20944/preprints202104.0515.v2

effort of ours in the paper serves as a catalyst in the literature of predicting stock market.

3 Methodology

We are in search of the most optimal vector, $h(t)$, that perfectly describes the relative importance of the previous n features (sampled at the given sampling rate) leading up the classification of a stock pattern as a consolidation breakout.

3.1 Assumptions

Market action determines stock behavior : Although changes in the stock market can be effected by changes in the general economy, such a qualification would pose an array of seemingly unambiguous metrics. Therefore, we simply limit the feature set to the history of stock behavior for any given stock represented with percent changes in price per share.

Trends exist in stock market data as stock patterns :

In the history of a stock, distinct stock behavior occurs and can be classified. The classes marking the distinct behavior range from rising wedge to double bottom to consolidation break [17] [15] [9]. The latter will be the focus of this work as its behavior is most distinct and can be most easily annotated. These stock patterns are identified in sliding window segments in the history of a stock by analyzing the change between the close price point of a stock. By predicting percent change between stock prices instead of the raw price itself, any identified pattern can be generalized to stocks at different price points.

Certain trends tend to reoccur : While there are effectively an infinite number of stock patterns in the history of a given stock, some tend to reoccur more than others. The stock patterns that reoccur the most frequently are of the most interest to financial analysts as they are the ones that are most likely to be anticipated.

Trends can be generalized at different time scales : Because of the availability of the data, we only evaluate this method at a single, non-granular sampling rate of one day. However, we do generalize our method to other time scales based on the notion that stock patterns exist at different time scales. For instance, a consolidation break at one time scale can be adapted to apply at another. This adaptation may come in a few forms, including a change in the values stored in the most optimal $h(t)$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

3.2 Convolution Operation

In order to form classifications on our input signal, we first propose the convolution operation as our objective function. Below is the convolution operation:

$$x(t) * h(t) = \int_{-\infty}^{+\infty} x(t-\tau)h(\tau)\mathrm{d}\tau \tag{1}$$

We utilize this function because it measures the degree of similarity between two signals in continuous time. As we are working with reported digital signals, the continuous time operation can easily be adapted for discrete time with constant intervals using a convolution sum:

$$x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x(n-k)h(k) \tag{2}$$

Effectively, this operation measures the amount of overlap between two signals in the time domain. This is most relevant to this study because we aim to classify signals most similar to a representative set of consolidation breakouts as such.

The two signals relevant to this study are

1. $x(t)$ stock price data represented in percent change and
2. $h(t)$ the weights that we can learn to predict a breakout.

3.3 Gradient Descent

Using a gradient descent algorithm, we can learn the weights stored in $h(t)$ by calculating the negative derived of our currently predicted vector through sampling the signal in constant time intervals. In other words, given a stock price window, we attempt to learn a set of weights that we can use to classify that window as breakout or not. Below is the derivative of the convolution sum function:

$$\frac{dh}{dn} = \frac{d}{dn}(f * g[n]) = \sum_{k=-\inf}^{\inf} f'[n-t]g[n] = f' * g \tag{3}$$

This derived definition can be used in code as it can be seen 1.

Initialize weights We generate a tail-heavy chi square distribution to initialize the weights as it can be seen in Figure 1 because we assume that for any given stock percent change prediction the most recent days have the most influence.

$$\tilde{\chi}^2 = \frac{1}{d}\sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

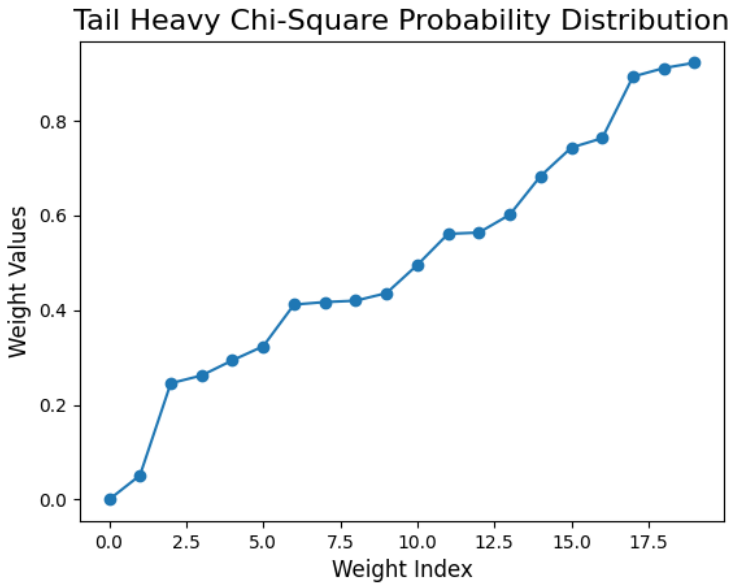


Fig. 1: Tail-heavy chi square distribution is generated to initialize the weights. The tail side has large values since breakout detection happens at the tail side

Algorithm 1 Performs gradient descent algorithm.

procedure LEARN_WEIGHTS($stock\_prices$, $nepoch$, $learning\_rate$, $momentum$)
    $n \leftarrow length(stock\_prices)$
    $x\_train, y\_train, x\_test, y\_test \leftarrow splitData(stock\_prices)$
    $delta\_weights \leftarrow zeros()$
    $weights \leftarrow generateChiDistr()$
    while $epoch < nepoch$ and $diff > thresh$ **do**
        $loss \leftarrow 0$
        for i in $range(n)$ **do**
            $loss \leftarrow loss + convolve(x\_train[i], weights)$
        $y\_pred \leftarrow logistic(x\_train * weights)$
        # diff(y_pred, prepend=[0]) means taking the
        # difference between each consecutive element.
        $delta\_weights \leftarrow -learning\_rate/n * t(x\_train) * convolve(y\_train, diff(y\_pred, prepend = [0])) + \lambda * 2/n * ||weights_i||^p + momentum * delta\_weights$
        # lambda is the regularization hyperparameter.
        # p can indicate whether the norm is l1, l2 or l1+l2
        $weights \leftarrow weights + delta\_weights$
    return $weights$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

3.4 Hyperparameters

The gradient descent uses 1000 epochs. The learning rate is set to be 0.05. Momentum is set to 0.001. Lambda value (λ) determines the regularization amount is set to 1e-14, which is always between 0 and 1. The weights are normalized at epoch to be between 0 and 1.

3.5 Regularization Study

L_1 (or Lasso) regularization and L_2 (or Ridge) regularization and L_1+L_2 (Elastic Net) regularization are applied on the weights to penalize getting large weight values.

If we define $f(x) = x \cdot w$, i.e. the approximation of y as characterized by an unknown vector of parameters (weights), w given an input x, we can then define $R(f)$ as $||w||_2^2$ for the case of L2 regularization e.g. Ridge regression, $||w||_1$ in the case of L1 regularization e.g. Lasso, and $(\alpha||w||_1 + (1-\alpha)||w||_2^2), \alpha \in [0,1]$ for Elastic Net. The L2 penalizes large values of w, while the L1 norm drives some of the coefficients exactly to zero, enforcing sparsity.

For below equations, L represents a model that assigns labels when given input x to predict y while optimizing over a loss function f,

i) L_1 (Lasso) We assume that sparsity must be introduced to the model to prevent neuron activation [18] and drive some values in the weights vectors to 0.

$$L(f, \mathbf{x}, y) = L_{function}(f, \mathbf{x}, y) + \lambda \sum_{f}^{n}{}_{i=1}|w_i|$$

ii) L_2 (Ridge) We prevent neuron activation by penalizing large values in the weights vector. penalizes large values of w

$$L(f, \mathbf{x}, y) = L_{function}(f, \mathbf{x}, y) + \lambda \sum_{f}^{n}{}_{i=1}w_i^2$$

iii) L_1+L_2 (Elastic Net) L_1+L_2 is a combination of Lasso and Ridge regularization.

$$L(f, \mathbf{x}, y) = L_{function}(f, \mathbf{x}, y) + \lambda \sum_{f}^{n}{}_{i=1}|w_i| + (1-\lambda) \sum_{f}^{n}{}_{i=1}w_i^2$$

Finding an optimal solution by optimizing for L_0 is non-linear. We approximate the L_0 to L_1 witht the assumption that the signal is sparse. [8][4]. However, we should note that using L_1 norm as proxy for L_0 doesn't guarantee optimal solution and depends on our function f. If we can prove that f is linear, we can guarantee the existence of optimal solution. If we fail to prove that f is linear, we fail to guarantee optimal solution when we approximate L_1 norm as proxy for L_0.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

Approximating L_1 norm as proxy for L_0 In this study, function f is a convolution operation on two signals 3.2. The two signals are x(t) the stock price action and h(t) the filter (weights that we are trying to learn). There are two operands, so the convolution operation is a binary operation since convolution is only defined between two functions, x(t) the stock price action and h(t) the filter. We have to prove that f is linear on two components. We say a function f is linear if

1. f(x+y) = f(x) + f(y)
2. f(ax) = af(x)

We know that h(t) is approximated to be a tail-heavy chi square distribution as it can be seen in section 3.3. Chi square distribution is not linear. So, we fail to prove linearity on the first component and therefore fail to guarantee optimal solution when we approximate L_1 norm as proxy for L_0.

Additionally, one cannot know whether x(t) is approximated to a function, let alone a linear one. x(t) is the stock price and if such accurate function approximation existed, it would defeat the purpose of this study and the stock market prediction literature. In our study, we assume history repeats itself as it can be seen in subsection 3.1, but this assumption is not enough to prove that there exists an x(t) that can determine stock price action deterministically. Therefore, we fail to prove linearity on the second component and therefore fail to guarantee optimal solution when we approximate L_1 norm as proxy for L_0.

3.6 Data

We use daily adjusted close prices for the following tickers from start date 2002-02-13 to end date 2021-02-12.

Index:

1. SPY

Technology Dataset:

1. AAPL
2. MSFT
3. TSM
4. NVDA
5. XLK

Health Care Dataset:

1. JNJ
2. UNH
3. NVS
4. ABT
5. XLV

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

Finance Dataset:

1. BRK-B
2. GS
3. JPM
4. BAC
5. XLF

The stock universe is chosen so that the stocks meet the following requirements:

1. At least $200B market cap
2. Average volume per day is over 1M
3. Current volume is over 1M

Due to missing stock price data for the chosen period, Visa and MasterCard stocks had to be dropped and be replaced with BAC and GS stocks. BAC and GS **donot** meet the "at least $200B market cap requirement".

3.7 Trading and Back-testing Methods

For our experiments, shorting a stock is allowed. That is, during backtesting, we can sell shares that we donot own in the hopes that stock price will go down and we will buy at a lower price. Additionally, only instruments that are allowed to trade in this paper are stock share securities. Option contracts are not allowed to trade. Additionally, we can enter a trade while we already have open trades just like day trading scenarios in real-life.

Strategy #0: Buy and Hold Strategy. The Buy and Hold strategy is used as a benchmark to for our subsequent strategies. Our Buy and Hold strategy represents buying one single share at the first date of the time period we studied on the market close and selling the share at the market close on the last date. We report the profits as percentage (%) amounts.

Strategy #1: Simple Consolidation Breakout Strategy We employ the following algorithm to detect a consolidation breakout.

This algorithm allows to look at a window of stock price, and predict whether the stock price will breakout up or down. The window size that we look at is 20 days. That is, we look at the stock price at market close for 20 days and enter a trade at the end of the window. We exit the trade after 3 days.

Strategy #2: Convolution Strategy In this strategy, we convolve weights that we learn via gradient descent with the stock price and classify whether the input is a breakout or not and treat the resulting signal as a loss function to optimize over a certain number of epochs. The intuition is that we can learn some set of weights that would classify a window of stock price as breakout our

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

Algorithm 2 Detects consolidation breakouts

```
procedure DETECT_SINGLE_CONSOLIDATION_BREAKOUT(close_prices)
    last_close_price ← close_prices[−1]
    if close_prices[: −1].max() + self.tolerance ≤ last_close_price : then
        # means our prediction is up
        return "bullish"
    else if close_prices[: −1].max()self.tolerance ≥ last_close_price : then
        # means our prediction is down
        return "bearish"
    else:
        # means our prediction is sideways
        return "neutral"
```

not. We learn the weights iteratively and our loss function is the amount of overlap between two signals: x(t) to represent stock price and h(t) to represent filter–weights. We are trying to minimize this loss. In other words, we are trying to optimize the amount of overlap between a signal that potentially indicates a breakout and another signal that is trying to classify whether the breakout exists. We only predict bullish breakouts.

4 Experiment Results and Evaluation

Results are shown in Table 1. Profitable trade count is the sum of profitable long and short trades counts. Unprofitable trade count is the sum of Unprofitable long and short trades counts. Total trade count is the sum of profitable and unprofitable trade counts. Success rate is the ratio of profitable trade count to total in terms of (%). Returns is calculated based on overall profit at the end of the backtesting simulation in (%). Buy & Hold will have N/As except the last column since there is only one single trade. Strategy 1 (section 3.7) is based on detecting consolidation breakouts.Strategy 2 is gradient-based learning via convolution operation. (section 3.7). Test accuracy of the model is only applicable to strategy 2 since only strategy 2 has a "trained model" based on learned weights via gradient descent.

4.1 Lessons Learned

Here, we define major lessons learned.

Strategy #0: Buy & Hold Buy and hold is a simple and profitable method and almost always outperforms

1. the SPY benchmark
2. 2 of all cases of strategy #1
3. all cases of strategy #2

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

Table 1: Results of each strategy separated by stock sector. Five stocks are chosen from technology, finance and health care sectors, the chosen stocks are largest mega-cap high-volume stocks. SPY is also added as a benchmark as well as their select sector funds such as XLF, XLV, XLK. For each sector, buy and hold returns are compared against strategy #1: consolidation breakout and strategy #2: gradient learning via convolution

Universe	Stock	Strategy	Profitable	Unprofitable	Total	Success Rate	Returns (%)
Benchmark	**SPY**	**Buy & Hold**	N/A	N/A	N/A	N/A	313.17
		Strategy 1	1629	1876	3505	46.48%	**464.14**
		Strategy 2	0	2	2	0%	-1.75
Tech	**AAPL**	**Buy & Hold**	N/A	N/A	N/A	N/A	**134.74**
		Strategy 1	795	811	1606	49.50%	128.47
		Strategy 2	0	1	0	0%	0
	MSFT	**Buy & Hold**	N/A	N/A	N/A	N/A	224.81
		Strategy 1	1448	1488	2936	49.32%	**391.64**
		Strategy 2	0	0	0	0%	0
	TSM	**Buy & Hold**	N/A	N/A	N/A	N/A	132.69
		Strategy 1	954	938	1892	50.42%	**189.23**
		Strategy 2	0	2	2	0%	-0.25
	NVDA	**Buy & Hold**	N/A	N/A	N/A	N/A	591.21
		Strategy 1	1633	1581	3214	50.81%	**1244.07**
		Strategy 2	0	0	0	0%	0
	XLK	**Buy & Hold**	N/A	N/A	N/A	N/A	**120.26**
		Strategy 1	1067	1074	2141	49.84%	50.89
		Strategy 2	0	2	2	0%	-1.31
Health	**JNJ**	**Buy & Hold**	N/A	N/A	N/A	N/A	**131.12**
		Strategy 1	1580	1631	3211	49.21%	118.51
		Strategy 2	0	2	2	0%	-0.138
	UNH	**Buy & Hold**	N/A	N/A	N/A	N/A	317.24
		Strategy 1	1634	194	3578	45.67%	**767.15**
		Strategy 2	0	2	2	0%	-0.02
	NVS	**Buy & Hold**	N/A	N/A	N/A	N/A	**72.93**
		Strategy 1	1430	1545	2975	48.07%	55.16
		Strategy 2	0	2	2	0%	0.543
	ABT	**Buy & Hold**	N/A	N/A	N/A	N/A	**112.09**
		Strategy 1	1192	1208	2400	49.67%	99.34
		Strategy 2	2	0	2	100%	0.36
	XLV	**Buy & Hold**	N/A	N/A	N/A	N/A	**96.02**
		Strategy 1	1140	1244	2384	47.82 %	73.83
		Strategy 2	0	2	2	0%	0.14
Finance	**BRK-B**	**Buy & Hold**	N/A	N/A	N/A	N/A	190.54
		Strategy 1	1919	1898	50.28	49.72 %	**317.42**
		Strategy 2	0	2	2	0 %	0.880
	GS	**Buy & Hold**	N/A	N/A	N/A	N/A	235.36
		Strategy 1	2156	2142	4298	50.16 %	**579.17**
		Strategy 2	0	2	2	0 %	-3.85
	JPM	**Buy & Hold**	N/A	N/A	N/A	N/A	121.95
		Strategy 1	1581	1686	3267	48.39 %	**160.61**
		Strategy 2	0	0	0	0 %	0
	BAC	**Buy & Hold**	N/A	N/A	N/A	N/A	**13.34**
		Strategy 1	1240	1291	2531	48.99 %	-105.85
		Strategy 2	0	2	2	0 %	-0.408
	XLF	**Buy & Hold**	N/A	N/A	N/A	N/A	**17.123**
		Strategy 1	808	801	1609	50.22 %	-82.86
		Strategy 2	0	2	2	0 %	-0.305

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

Strategy #1: Consolidation Breakout This strategy outperforms buy and hold strategy in almost all cases except BAC and JPM. One should note that BAC results in a negative return from buy and hold and BAC is not a mega cap stock unlike all others.

Strategy #2: Convolution based Breakout Convolution based Breakout underperforms all other benchmarks and simulations. This can be attributed to many factors: The gradient learning based on convolution operation resulted in a very sensitive model to the hyperparameters. This sensitivity to hyperparameters means a small change in the hyperparameters **donot** change the model predictions. Different regularization amounts added to the weights didn't change the model accuracies and predictions. Model accuracy resulted after applying L_1, L_2 and $L1_L2$ regularizations did not differ until the 8th digit after the decimal. Due to these issues, optimizing the model that we achieved during strategy 2 (section 3.7) via hyperparameters was difficult. That model also resulted in extreme bias towards one class (there are two classes: breakout or not) and the predictions resulted in losing money. As it can be seen from loss functions in section 8, the loss values per epoch for gradient learning based on convolution operato in (strategy 2) decrease over each epoch. This proves that the model is performing the gradient descent correctly. This is a major achievement since it proves that the convolution operation can be used as a loss function during gradient learning, - a novel discovery that is yet explored in the current literature.

5 Conclusion

In this paper, we investigate the field of predicting stock price action. Unlike computer vision models that look at the image of a stock price action, we explore using convolution operation on raw dollar values to predict consolidation breakouts under a supervised learning problem setting. Unlike LSTMs that predict stock prices given continuous stock data, we use the continuous stock data to classify a given price window as breakout or not. We propose buy and hold as benchmark and outline two strategies: strategy #1 where we trade based on consolidation breakouts outperforms and strategy #2 where we trade breakouts based on learned weights after convolution based gradient learning. Strategy #1 outperforms SPY and all other buy and hold benchmarks whereas strategy #2 underperforms significantly. Finally, we do a regularization study to see the effect of L_1, L_2, $L_1 + L_2$ and $ElasticNet$ regularization on the gradients learned in strategy #2. We hope that combining regression and classification shed more light on stock market prediction studies.

6 Future Work

In this paper, we limited the back-testing simulations to only trading stock shares and not option contracts. However, there exist option contracts, which

are security instruments designed to capture profit when the stock price moves very large amount (either up or down) from a predicted stock price. In the future, we can look into including selling and buying option contracts when detecting consolidation breakouts. Including option contracts to our back-testing studies in the future might be worthwhile to investigate in the search of outperforming buy & hold strategy.

Additionally, an extension to this paper would be to include the classification of a variety of different stock patterns beyond just consolidation breakout. In term, this problem would turn into a multi-class classification problem as opposed to the simple binary classification problem explored in this work. Similarly, one can explore different parameters in characterizing and annotating different types of a given stock pattern, such as consolidation breakout. A profit analysis of this change in stock pattern parameterization would offer some insight into the optimal definition of a given stock.

7 Acknowledgements

We would like to thank Drexel Society of Artificial Intelligence members, especially Jeff Winchell, Chaim Rahmani, Shesh Dave, Justine Goheen and Adam Dunlop due to their attention and support throughout various phases in our research.

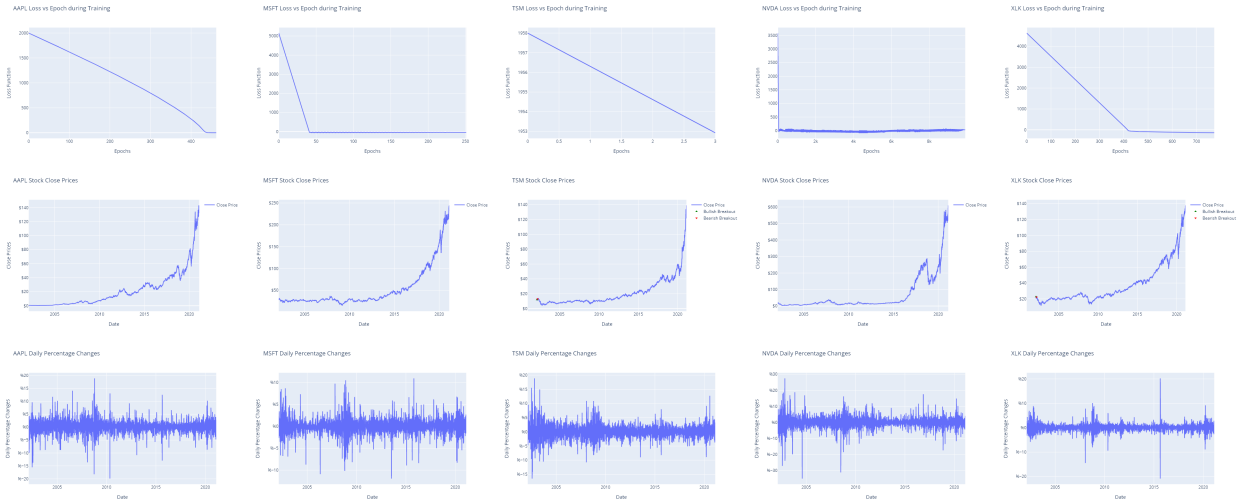
References

1. Alparslan, Y., Kim, E.: Extreme volatility prediction in stock market: When gamestop meets long short-term memory networks (2021)
2. Barro, R.J.: The stock market and investment. The review of financial studies **3**(1), 115–131 (1990)
3. Campbell, J.Y., Grossman, S.J., Wang, J.: Trading volume and serial correlation in stock returns. The Quarterly Journal of Economics **108**(4), 905–939 (1993)
4. Candes, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Processing Magazine **25**(2), 21–30 (2008). https://doi.org/10.1109/MSP.2007.914731
5. Chen, K., Zhou, Y., Dai, F.: A lstm-based method for stock returns prediction: A case study of china stock market. In: 2015 IEEE international conference on big data (big data). pp. 2823–2824. IEEE (2015)
6. Chong, E., Han, C., Park, F.C.: Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications **83**, 187–205 (2017)
7. De Bondt, W.F., Thaler, R.: Does the stock market overreact? The Journal of finance **40**(3), 793–805 (1985)
8. Feng, M., Mitchell, J.E., Pang, J.S., Shen, X., Wächter, A.: Complementarity formulations of l0-norm optimization problems. Industrial Engineering and Management Sciences. Technical Report. Northwestern University, Evanston, IL, USA (2013)
9. Harris, L.: A transaction data study of weekly and intradaily patterns in stock returns. Journal of financial economics **16**(1), 99–117 (1986)

10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
11. Hoseinzade, E., Haratizadeh, S.: Cnnpred: Cnn-based stock market prediction using several data sources. arXiv preprint arXiv:1810.08923 (2018)
12. Marszałek, A., Burczyński, T.: Modeling and forecasting financial time series with ordered fuzzy candlesticks. Information sciences **273**, 144–155 (2014)
13. Nti, I.K., Adekoya, A.F., Weyori, B.A.: A systematic review of fundamental and technical analysis of stock market predictions. Artificial Intelligence Review pp. 1–51 (2019)
14. de Oliveira, F.A., Nobre, C.N., Zárate, L.E.: Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index–case study of petr4, petrobras, brazil. Expert Systems with Applications **40**(18), 7596–7606 (2013)
15. Poterba, J.M., Samwick, A.A., Shleifer, A., Shiller, R.J.: Stock ownership patterns, stock market fluctuations, and consumption. Brookings papers on economic activity **1995**(2), 295–372 (1995)
16. Qian, B., Rasheed, K.: Stock market prediction with multiple classifiers. Applied Intelligence **26**(1), 25–33 (2007)
17. Roberts, H.V.: Stock-market" patterns" and financial analysis: methodological suggestions. The Journal of Finance **14**(1), 1–10 (1959)
18. Schwartz, D., Alparslan, Y., Kim, E.: Regularization and sparsity for adversarial robustness and stable attribution. In: International Symposium on Visual Computing. pp. 3–14. Springer (2020)
19. Soni, S.: Applications of anns in stock market prediction: a survey. International Journal of Computer Science & Engineering Technology **2**(3), 71–83 (2011)
20. Teweles, R.J., Bradley, E.S.: The stock market, vol. 64. John Wiley & Sons (1998)
21. Wang, X., Phua, P.K.H., Lin, W.: Stock market prediction using neural networks: Does trading volume help in short-term prediction? In: Proceedings of the International Joint Conference on Neural Networks, 2003. vol. 4, pp. 2438–2442. IEEE (2003)
22. Zhou, X., Pan, Z., Hu, G., Tang, S., Zhao, C.: Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering **2018** (2018)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2

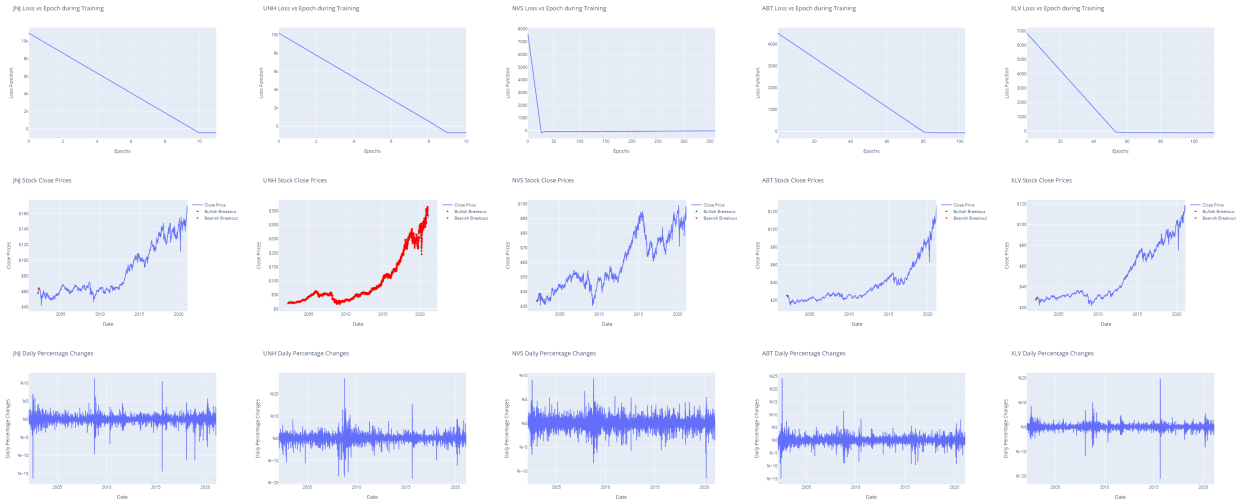
8 Appendix



(a) AAPL (b) MSFT (c) TSM (d) NVDA (e) XLK

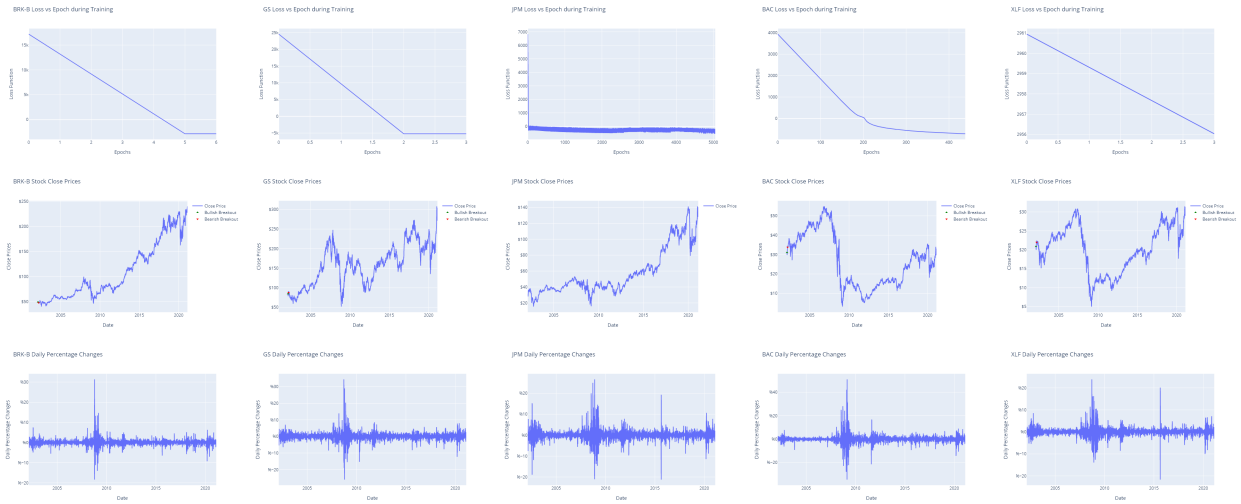
Fig. 2: Tech stock universe. First row represents loss versus epoch when training model for strategy #2. Second row represents close prices for the stock. Third row represents daily percentage changes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0515.v2



(a) JNJ (b) UNH (c) NVS (d) ABT (e) XLV

Fig. 3: Health Care stock universe. First row represents loss versus epoch when training model for strategy #2. Second row represents close prices for the stock. Third row represents daily percentage changes.



(a) BRK-B (b) GS (c) JPM (d) BAC (e) XLF

Fig. 4: Finance stock universe. First row represents loss versus epoch when training model for strategy #2. Second row represents close prices for the stock. Third row represents daily percentage changes.