# Efficient Prediction of Gold Prices Using Hybrid Deep Learning

Turner Tobin[1] and Rasha Kashef[2](✉)

[1] Computer Science Department, University of Western Ontario, London, Canada
ttobin@uwo.ca

[2] Electrical, Computer and Biomedical Engineering, Ryerson University, Toronto, Canada
rkashef@ryerson.ca

**Abstract.** Gold prices, in general, act counter to the market and are thus predictable to a certain degree based upon the fluctuations of other market entities. Neural networks have been applied to significant effect to difficult prediction problems and have achieved success in making predictions beyond traditional regression-based statistical models. Generally, such networks are hyper-specialized, and thus have effectiveness in a small subset of problems. This paper endeavors to take two models: RNN and CNN and hybridize them, creating a new model founded on their underlying logical theories and architectures. The primary goal in this paper is to show the effectiveness of the hybrid model in achieving a better gold price prediction that produces higher quality results at the cost of slightly increased complexity. Challenges and limitations of the proposed hybrid model are also addressed.

**Keywords:** Gold price prediction · Time-series · RNN · CNN · Hybrid learning

## 1 Introduction

While most financial assets are valued for their intrinsic properties combined with their relation to similar assets, gold tends to operate more extrinsically with no factors inherent to itself [1]. When investors lose trust in the market or centralized currencies, gold is often seen by the individual investor to be a viable option to purchase as a hedge [2]. When predicting gold prices, the data collection focuses on readily available market conditions, as opposed to having to input a company's specific financial statements and monitor its actions [3]. The performance of a predictive algorithm in forecasting gold prices comes down to the strength of the algorithm itself [4]. In this paper, we cover leading models in deep learning as applied with great success to the problem of predicting gold prices. The paper endeavors to show how the qualities that lead to success in different architectures can be combined to create new networks. Meaningful results can be accomplished through creating new networks in such a fashion, beyond creating ensembles through taking the output of one model and feeding it to another. We explore the underlying logical theories and architecture in combing both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in a hybrid model to achieve successful prediction results. Experimental results on market data show improvement in predicting gold price two days in advance using the hybrid deep learning model. In Sect. 2, a brief discussion on CNN, RNN, and Autoregressive Integrated Moving Average (ARIMA) is given. Section 3 presents the hybrid model. Experimental results are presented and discussed in Sect. 4. Finally, derived conclusions and future directions are presented in Sect. 5.

## 2 Related Work and Background

Deep learning concerns itself with networks that achieve complexity through multi-layer interaction. Various models, activation functions and cost functions have been put forward and refined throughout the years [9]. It has been found that certain types of network architecture tend towards success in certain problem applications as CNN and RNN, especially for stock market prediction [17] and time-series analytics [5, 18]. In the following, a brief discussion on the convolutional neural networks, recurrent neural networks, and ARIMA predication model is presented.

### 2.1 Convolutional Neural Networks (CNN)

The CNN operate by reacting to input, passing that reaction forward to further neurons, and training a receptive field to interpret the response and begin to make predictions [5]. A CNN is implemented in a series of alternating layers; these layers are ordered such that convolutional layers alternating with pooling layers [6]. Convolutional layers apply an operation to the input but are not a trainable part of the network [6]. Pooling layers reduce the number of independent variables at the end of the process that is passed on to the receptive field [7]. Pooling layers can be either global or local; if the depth of the problem is shrunk between convolutional layers, it can be considered as a local pooling layer [8]. If the pooling happens at the end of the convolutions it is called global [8]. The more convolutional layers in the network the "deeper" it is considered. CNN are used to process images or visual data because of their ability to interpret spatially linked data [9]. The process begins with a 2-dimensional matrix of data that is tied together by proximity. Pooling allows for approximate answers for an area to be reached between steps to increase efficiency and to decrease the number of weights to be calculated down the line [7]. It is an effective way to take a data source with many variables, commonly pixels, and distill it into a smaller set of variables to use to predict with [5, 9].

### 2.2 Recurrent Neural Networks (RNN)

The RNN are networks where information is passed forward from node to node along a chain. Each node can process data from its memory. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network that can "forget" based on its predictions, using a "forget gate" [10]. A LSTM unit handles multiple operations while training. First, each unit decides how much of the factor to keep when bringing it into the unit (i.e. a weight) [11]. Then, it updates its future internal state. Finally, it determines the output to the next unit along the graph, and what to forget for later [11]. These "units" can be stacked, which creates an approximation of factors in time [12]. As CNNs are generally used in spatial data and images, RNNs are typically used in predicting the meaning behind historical data and speech [6]. Natural language processing can be applied through such RNN networks as nodes will remember things like names, genders, and plurality. Then nodes will make predictions as to what a sentence is saying using such factors because it encounters an ambiguous pronoun like a "they" or "she" [13]. It also retains the necessary information to make such decisions and forgets information as it no longer applies.

### 2.3 ARIMA (Autoregressive Integrated Moving Average)

ARIMA is a statistical model in which a series of coefficients are found and multiplied with some number of lagged values. This is combined with the linear combination of its margins of error over time to get a regression-based model with a combined margin of error [14]. This is covered as a standard normal statistical performance, as it is a widely accepted as a non-neural network method of forming predictions. Regression in some sense is useful to the structure of the model, as a gradient regression is applied to find the best possible weights and exponents for the series of nodes [15].

## 3 Hybrid CNN-RNN

The CNN have successfully modeled spatially related data, and RNN have effectively modeled temporally related data [9, 11, 13]. The ensemble approach endeavors to create a mixed model that has success modeling temporally related data, with a spatial aspect, or spatially related data with a temporal aspect. The hybrid model takes the concept of directed graphing and weighing from RNN, and pooling and convolution from CNN to form a more comprehensive model. This model is capable of solving either a problem meant for CNN or RNN or joint problems. Data is inputted into a matrix where every square in the matrix contains the root of a tree. Every tree starts with a node that contains a vector of one variable's values throughout time, a coefficient and an exponent, as well as whatever dummy nodes are required to maintain consistency in the shape of the tree as it evolves. This matrix, full of roots of trees, is then convoluted by passing a filter over the matrix. The filter takes an $n \times n$ square of the matrix and creates a new tree of these nodes, which then occupies a space in that matrix.

$$\begin{bmatrix} \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 \\ \mathrm{x}_4 & \mathrm{x}_5 & \mathrm{x}_6 \\ \mathrm{x}_7 & \mathrm{x}_8 & \mathrm{x}_9 \end{bmatrix} = \begin{bmatrix} \mathrm{x}_1 & \mathrm{x}_2 \\ \mathrm{x}_3 & \mathrm{x}_4 \end{bmatrix}, \begin{bmatrix} \mathrm{x}_2 & \mathrm{x}_3 \\ \mathrm{x}_5 & \mathrm{x}_6 \end{bmatrix}, \begin{bmatrix} \mathrm{x}_4 & \mathrm{x}_5 \\ \mathrm{x}_7 & \mathrm{x}_8 \end{bmatrix}, \begin{bmatrix} \mathrm{x}_5 & \mathrm{x}_6 \\ \mathrm{x}_8 & \mathrm{x}_9 \end{bmatrix} \tag{1}$$

The sub-matrix Xsub from filtering is defined as:

$$Xsub = \begin{bmatrix} \mathrm{x}_1 & \mathrm{x}_2 \\ \mathrm{x}_3 & \mathrm{x}_4 \end{bmatrix} \tag{2}$$

The sub-matrix *Xsub* is represented as a tree, as shown in Fig. 1. The tree represents the values in the convolution matrix, with the final multiplication operation returning just the left side of the equation. This process continues until the matrix is $1 \times 1$, and the resulting values calculated from that node are the final predictions. This can be interpreted as a shrinking matrix, a growing tree, or an equation builder. The hybrid model uses trees to represent predictions on behalf of one section of the matrix and uses those trees to generate a future tree that could describe another scenario. The advantage of the hybrid model is that there is no need to know the best approach, as the model can find not only the optimum way to weight the factors within the nodes but also the ideal way to arrange data within the matrix to form a tree that can yield a good solution.

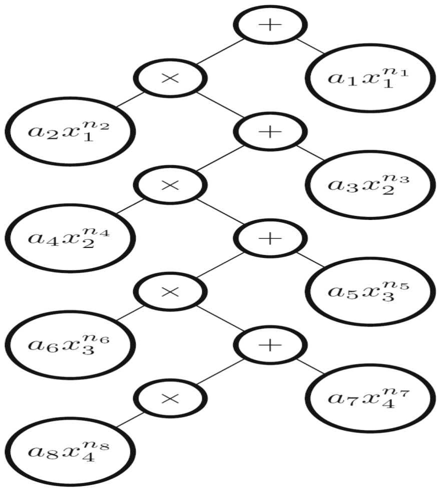


**Fig. 1.** Tree representation of the convolution matrix

## 3.1 The Hybrid Model

The input to our hybrid model is a 2D array as layers of sub-arrays, where the outside layer array is a container for sub-arrays, each contains a single variable's values through all trials and each index represents a one-time step. The output is another 2D array, where there is one sub-array which is the predicted values, with the last $x$ being not a part of the training set when predicting. The hybrid CNN-RNN model constitutes of five main steps, the convolution step, the filtering step, the pooling step, Gradient Regression, and finally the verification step.

### 3.1.1 The Convolution Step

Starting with a matrix filled with trees of one useful node, a square is passed to the matrix, such that this square along with the internal nodes within this square form the new subtree. Trees are formed on the basis of the interaction between new nodes and old nodes to simulate an equation. This resolves the problem of scaling complexity in equation building. For example, if given three variables, 7 different terms are required.

$$(x) = xyz + xy + xz + zy + x + y + z \tag{3}$$

Using a tree system is an attempt to come up with close enough approximations of prediction answers while maintaining the directed graph-like structure of an RNN. Answers are passed upward through to make further predictions. The basic theory is such that not every combination need be checked, an answer can be made sufficiently acceptable and actionable given one possible combination and calibrated to be the best it can be. Enough combinations can be attempted and optimized to find a solution that is acceptable without having the computational cost of attempting close to every permutation. The algorithm operates the worst case of O $(r \times k \times n)$ in the model used to gather results in this study, this equation would be of the form:

$$f(x) = x * (y * (z + z) + y) + x \tag{4}$$

This example is just one appropriate way to achieve a linear chain for initial testing as it is able to represent new information being brought in and yielding a polynomial answer instead of a linear combination. The program used to run this model could also be adapted to try different forms of combining nodes, at a cost of higher time complexity.

### 3.1.2 Filtering Step

Filter size was something of an arbitrary decision to test through this paper, we erred on the side of the highest precision manageable. This meant making as many sub predictions as possible and thus we went with subgroups of four elements. This was thought to have the most possible information to draw from. This would come into play in a larger degree if nodes were to "lock" meaning no longer recalculate as the sub predictions would have a higher degree of impact. Now the filter size only comes into play because of the actual geometric layout of the tree. This matters as two factors that have an effect on each other may not be able to interact as they may be 3 squares from each other and not 2. If given enough attempts to form different matrix layouts this would be minimized, but we may be able to see that larger filter size has a positive effect in tree building. The other issue with filtering is the overlap. To what degree does the next square to build a tree from overlap with the last square. In this model, the overlap is maximized, as time complexity was not overly concerning with 732 trials.

### 3.1.3 The Pooling Step

When each tree is formed, it becomes an element of the next matrix which takes more trees as input. In the CNN sense, pooling as the matrix gets reduced to a smaller operating size, data is retained throughout the process and the problem goes from being a very wide problem to very deep. Calibrating such a tree has the same complexity. The study began with the idea that in such a case the trees would at a certain point stop recalculating to save execution time, and they would become fixed positions. This would limit potential solutions to problems but would save enough time that solutions could be reached given very large problems. In the model analyzed for results, all nodes remained active throughout the problem solving as it was a small sample of data.

### 3.1.4 Gradient Regression of Weights and Exponents

In a typical CNN, as data is added, weights would be adjusted, given enough trials weights would find their way to an optimal solution that best classifies problems [5]. In our scenario, all data is transferred to the nodes. Upon calibration, coefficients and exponents of the network are altered and then tested against the knowledge of inputs and solutions. With low size of inputs, this would lead to difficulty in reaching a meaningful solution if the precision with which we increment is too high. However, if the data sources became large, doing so in such a process would become too computationally expensive. This would require either transitioning to a model in which the values were not recalculated and instead weights were organized on an ongoing basis or potentially through taking slices. This could lead to greater error as potentially some outliers within the graph may not get captured accurately in the slices.

### 3.1.5 Verification Step

Upon updating nodes, we must know whether or not the solution has become closer to an optimal solution. Calculating the mean squared error was tested as a valid option, being slightly more computationally expensive than calculating the mean error and more punishing on extremely incorrect cases. When calibrating on a mean squared error basis, the hybrid model tended to find solutions that were more moderate. Instead of modeling large swings and being correct on some large variations and incorrect on others, the hybrid mode predicted results that were equally incorrect for all trials but resulted in a lower level of error overall. Calculating the number of predictions that are correct, or within a desirable range would be effective as it is accurately expressing the types of solutions. However; this type of solution leads to less accuracy when calibrating as it outputs what is essentially a step function. When calibrating a node by incrementing a coefficient by one thousandth, it may make the solution better, but not cause an increase of one more correct trial instantly, and thus the program would interpret it as no change. Therefore, further experimentation needs to go into a verification method that outputs a metric by the number of successes, possibly could yield stronger results.

## 3.2 Challenges in the Hybrid Model

In the hybrid model for this problem instance there are three main challenges: (1) the occurrence of a prime numbered quantity of free variables, (2) data gathered in business days, and (3) the inability to use growth vectors due to the shortcomings of the data.

### 3.2.1 Prime Numbered Variables

In the case of odd-numbered variables, data needs to be filled into the matrix such that proper filtering can occur. This means that for every subsection of the matrix, that area of the matrix is populated. Originally, the solution was that when filling a matrix if the data had run out but there were still areas to populate, those areas should be filled in with variable number one. This variable acts as an arbitrary number that would be automatically factored out if irrelevant. Later in testing, it was seen that while filling the matrix, a simple solution would be to take the index to the mod of the number of variables, so it was not always the origin being used as a filler.

### 3.2.2 Data Challenges

Inherent limits to financial data are that there is only data for weekdays. The markets are only open Monday to Friday. The data used was additionally weak as it was collected once per day. This means per week there is only 5 data points per variable. For the model to be able to produce results with significance for the future, the model uses data two days out to make predictions. A prediction on Monday using Thursday's data would follow a different decision-making pattern than a prediction for Wednesday using Monday's data. Thus, the decision was made to keep all decisions within the week. This limits the effective data, as per week now there are only three predictions and thus there is 3/5 of an already small data pool. This means to get enough data to train a model, it must be trained over many years. Previous studies have shown financial models train better on a tighter timescale and stretching past a number of years leads to decreased accuracy in results [16]. Future work on using data across different time zones to reduce the gap between Friday and Monday is recommended.

### 3.2.3 Growth Vectors

Instead of holding absolute values in each node, it is recommended to use a growth vectors. The model can find the best combination of all growth vectors, then the estimated growth of the output variable from one trial to another would increase the ability to bring unlike data sources together. The growing of two factors, and the shrinking of another factor may lead the output to grow at an exponential rate, and the model then would not generate an absolute number. For financial data, multiplying the trading volume by the share price to get an answer in dollars of gold price. This does not make inherent sense and would not be in appropriate units. The model will still output a good prediction, but the prediction would make more sense as how much the gold price would grow, given the growth rate of the trading volume and share price of another asset. This would potentially solve problems in data sets such as this one where outputs vary from in the range of 80 to the range of 20. The model needs to only predict the drop and then predict it stagnates, as opposed to predicting its exact position. The issue is that to get growth numbers, numbers from $-1$ days are needed. However, we have daily data, and 2-day lead time, thus we only have predictions for Thursday and Friday, and we would have 2/5 of the sample size. The main purpose of switching this model specifically to a growth-based algorithm would be the ability to include things like sentiment analysis of news sources and Google Trends data into predictions. This will not give an output of trading volume only but increase in gold searches and a decrease in the gold price, thus this could have a meaning in terms of an increase in trading volume.

## 4 Experimental Results

In this section, we test the performance of the hybrid model to that of ARIMA, CNN and LSTM as benchmark [4]. The dataset used Yahoo Finance data, future datasets will be considered. The given time period used in the contrasting study was used. Data was gathered on the performance of the S&P 500 index and the Barrick Gold price, which included their daily highs and lows, trading volume, open, close and adjusted close. The data, other than the output variable of daily high for gold price, was shifted back 2 days so the model gave actionable information. The variables were not duplicated and shifted back since there is only data on weekdays, which makes shifting it a challenge. The output predictions are for the same entity (GOLD), and the data gathered is also at intervals of one day [4]. Figures 2, 3 and 4 show the prediction results of the ARIMA, CNN, and LSTM, respectively. We can observe that ARIMA has the worst performance and the LSTM achieves significant forecasting results as compared to the ARIMA and CNN. To increase comparability, we will show results of the hybrid model up to the same point in time as the other models. The analysis of the Mean Squared Errors (MSE) in Table 1 shows that the hybrid model outperforming the CNN, LSTM and ARIMA.

**Table 1.** Mean squared errors (MSE) comparison

| Model | Mean squared error |
|---|---|
| ARIMA | 1.90E+07 |
| CNN | 8.65E+00 |
| LSTM | 7.12E+00 |
| Hybrid CNN-RNN | 6.00E+00 |

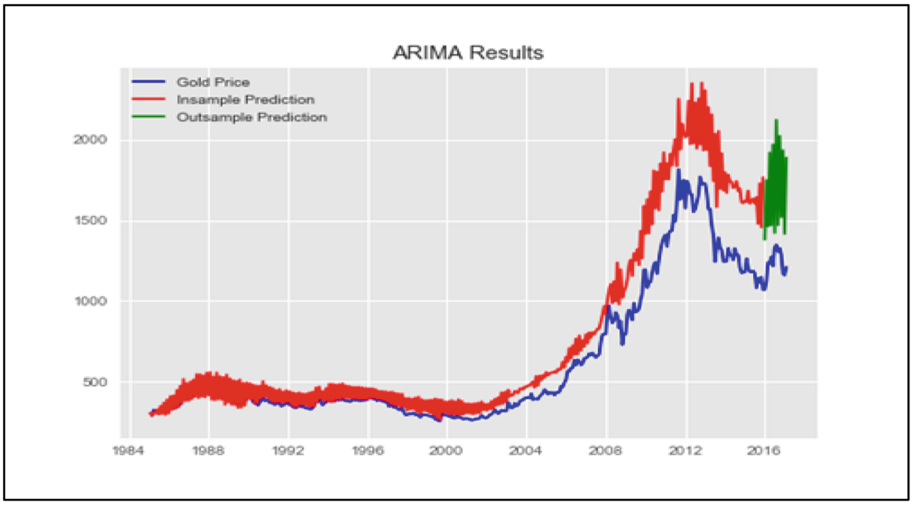


**Fig. 2.** ARIMA prediction results

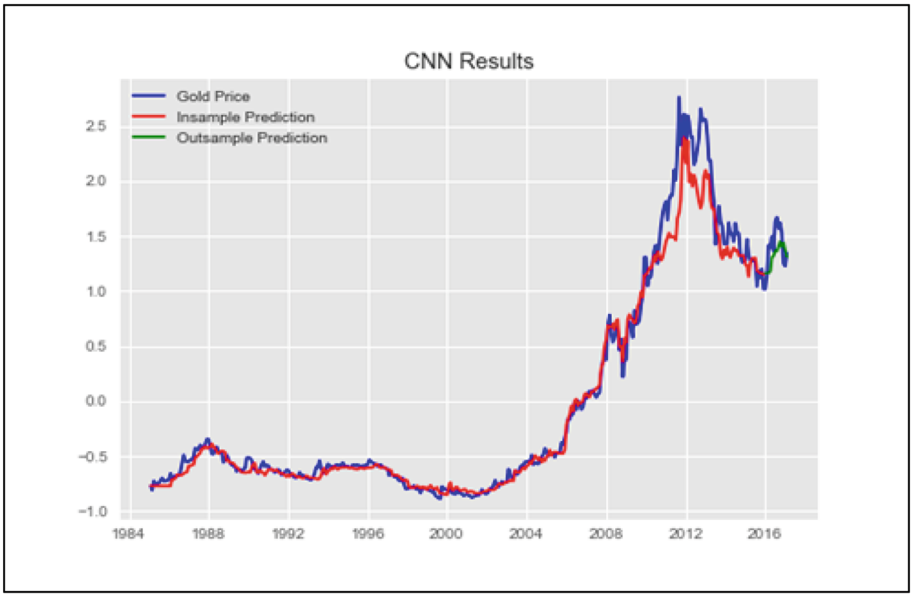


**Fig. 3.** CNN prediction results

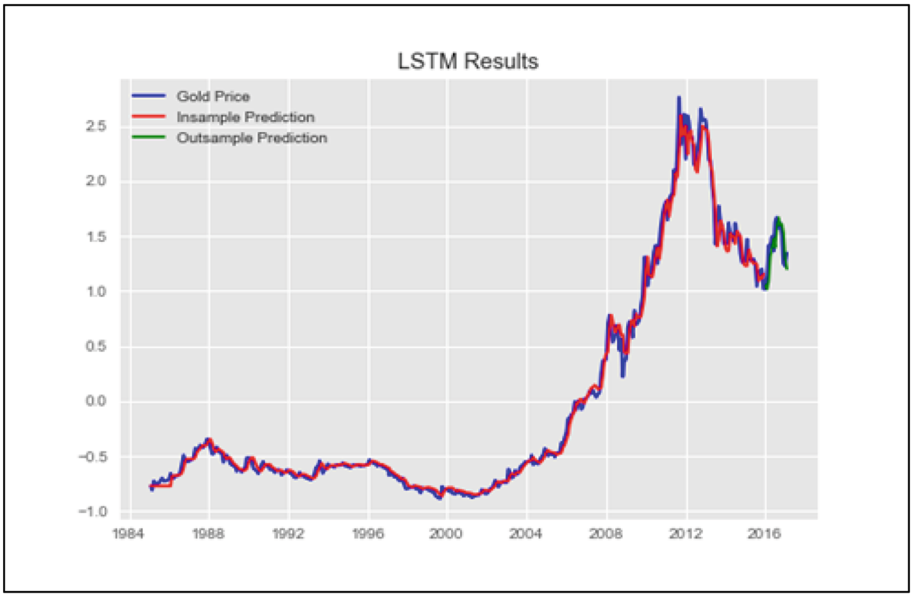
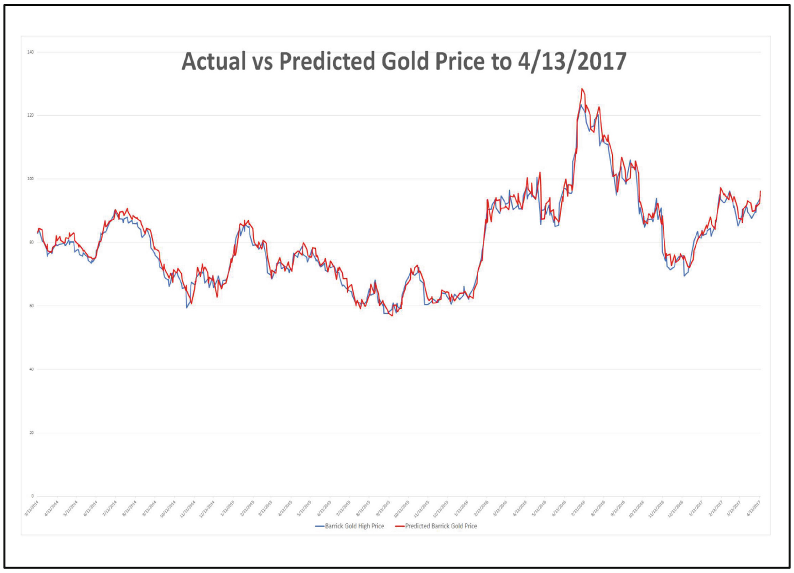
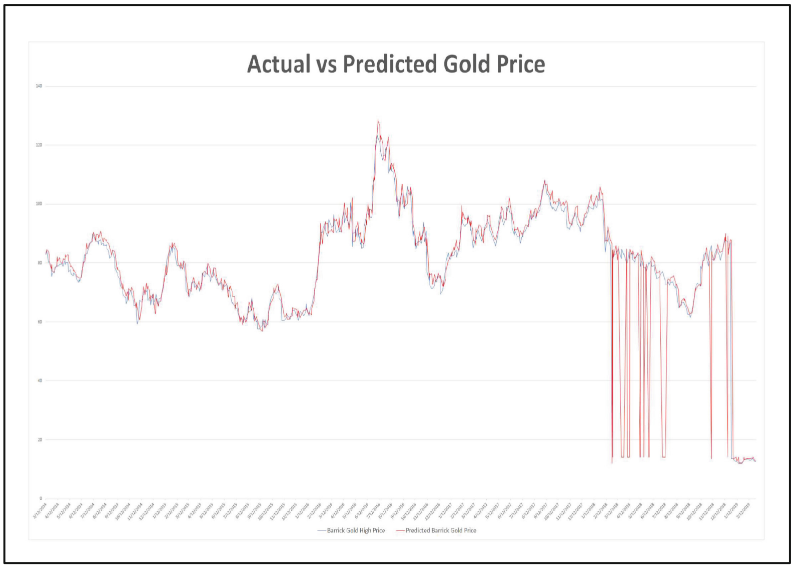


**Fig. 4.** LSTM prediction results

Figures 5.a and 5.b show the performance of the Hybrid Model, with one series representing predictions and the other represents the actual prices up until the end of the comparable data for data collected in 2017 and 2016, respectively.



(a) Predicted Vs. Actual prices up until the end of the comparable data (2017)



(b) Predicted Vs. Actual prices up until the end of the comparable data (2016)

**Fig. 5.** Hybrid CNN-RNN prediction results

Looking at the graphed results in Figs. 4 and 5, we can see that the LSTM and hybrid model perform to similar degrees of accuracy, and these mean squared errors represent deviations away from the standard pattern. Two large drops in the price cause a series of large mispredictions (Fig. 5.b). This was covered in the verification section, as switching to verifying solution quality by mean squared error ended up making the same predictions to a lesser degree: in essence, squashing the graph. Potential solutions to this would be to have a working buffer, so calibration forgets data at an arbitrary date cutoff, or the introduction of a formal forget gate into the tree structure which would require some logical reworking but may be necessary to fully emulate the LSTM structure within the hybrid model. The Hybrid model also has a speed up improvement up to 20% faster than both CNN and LSTM.

## 5 Conclusion and Future Directions

It seems a natural leap to combine algorithms (or models) with proficiency to achieve better prediction. This paper shows that higher quality connections can be made using limited data with a more comprehensive ensemble model. In this paper, we proposed a hybrid deep learning model that combines both CNN and RNN in an ensemble learning strategy. The model works significantly better with fewer data and data breadth. The Hybrid CNN-RNN model can outperform some of the existing statistical learning models. Future research directions include analyzing the effects of various filter sizes, quantities of filler data, amount to overlap filter, slicing of data, and depth cutoffs. An extended refinement to the verification phase using a better analyzing tree geometry is also of future investigation. Lastly, whether the calibration should occur while building the tree or at the end from the top down is a future research direction. The model needs further optimization by adding data from more market variables as well as adding sentiment analysis or Google Trends data. In addition, obtaining better quality data with more sources to bypass the one entry per day restraints and experimenting with growth vectors would be an extension to the work in this paper.

## References

1. Cai, J., Cheung, Y.-L., Wong, M.C.S.: What moves the gold market? J. Futures Mark. **21**(3), 257–278 (2001)
2. Baur, D.G., Lucey, B.M.: Is gold a hedge or a safe haven? An analysis of stocks, bonds and gold. Fin. Rev. **45**(2), 217–229 (2010)
3. Aggarwal, R., Soenen, L.A.: The Nature and efficency of the gold market. J. Portfolio Manag. **14**, 201–213 (1988)
4. Srivastava, S.: Deep Learning in Finance. Towards Data Science (2017)
5. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2014)
6. Abdel-Hamid, O., Deng, L., Yu, D.: Exploring convolutional neural network structures and optimization techniques for speech recognition. In: INTERSPEECH, Lyon (2013)
7. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR, San Diego (2015)
8. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv (2014)
9. Fischer, M.M.: Computational neural networks—tools for spatial data analysis. In: Spatial Analysis and Geo Computation, pp. 79–102 (2006). https://doi.org/10.1007/3-540-35730-0_6
10. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. In: 9th International Conference on Artificial Neural Networks: ICANN 1999, Edinburgh (1999)

11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Hansen, J.V., McDonald, J.B., Nelson, R.D.: Time series prediction with genetic-algorithm designed neural networks: an empirical comparison with modern statistical models. Comput. Intell. **15**, 171–184 (2002)
13. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S.: Recurrent neural network based language model. In: INTERSPEECH (2010)
14. Sowell, F.: Modeling long-run behaviour with the fractional ARIMA model. J. Monetary Econ. **29**, 277–302 (1992)
15. Adebiyi, A., Adewumi, A.O., Ayo, C.K.: Comparison of ARIMA and artificial neural networks models for stock price prediction. J. Appl. Math. **2014**, 614342:1–614342:7 (2014)
16. Sami, I., Nazir, K.: Predicting future gold rates using machine learning approach. Int. J. Adv. Comput Sci. Appl. **8**(12), 92–99 (2018)
17. Kim, T., Kim, H.Y.: Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PLoS ONE **14**(2), e0212320 (2019)
18. Tan, X., Kashef, R.: Predicting the closing price of cryptocurrencies: a comparative study. In: Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems (DATA 2019). Association for Computing Machinery, New York, NY, USA (2019). Article 37, 1–5. https://doi.org/10.1145/3368691.3368728