Predicting Stock Index Returns using Recurrent and Convolutional Neural Networks

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

Abstract

Convolutional neural networks (CNNs) are typically associated with image visioning rather than time-series. Recurrent neural networks (RNNs), in particular Long Term Short Memory (LSTMs), have been the most studied deep-learning model for financial time series. This paper aims to capture correlated features from the component stocks of the NASDAQ 100 through a CNN in an effort to predict the future NASDAQ 100 price. The baseline model is a Dual-Stage Attention-Based RNN (DA-RNN) which is tested against an LSTM, CNN, and an Encoder Decoder CNN-LSTM.

1. Introduction

Predicting intraday prices has traditionally relied upon supervised learning models such as SVMs, Random Forests, or Linear Regressions. These models often require an expert to engineer features that capture "alpha", or predictive edge. Deep-learning algorithms that can model time-series, like RNNs and LSTMs, have only recently been studied on intraday time periods. These studies primarily focus on a singular financial time series and often lack information from correlated financial instruments.

The main motivation for applying convolutional neural networks to a financial time series is to determine if the CNN is able to generate features that captures information from correlated products.

1.1. Input & Output

The data consists of 1-minute opening prices for 81 companies that are constituents of the NASDAQ 100 stock index and the corresponding NASDAQ 100 opening prices. The data is formatted into a 3d image with a single layer (C=1) of opening prices with a height equivalent to the number of time periods in each batch (H=100) and the width represents the number of stocks used (W=81).

The above data was fed to the following models: 1) Dual-Stage Attention-Based RNN (DA-RNN), LSTM, CNN, and an Encoder-Decoder CNN-LSTM to output a predicted future 1-minute opening price for the NASDAQ 100 stock index.

2. Related Work

Time series deep learning research has been primarily focussed on Recurrent Neural Networks (RNNs), in particular Long-Term Short Memory (LSTMs) cells, to predict a singular financial time series [1]. Traditionally, supervised learning models have been used to forecast the movements in index returns relying on expert-based features, technical indicators, and macroeconomic indicators [2].

Artificial neural networks showed some improvement over supervised learning models for daily index prediction [3]. Persio and Honchar laid the framework for comparing different neural network models for stock prediction in their paper [4]. The authors tested a Multi-layer Perceptron (MLP), a Convolutional Neural Networks (CNN), and a Long Short-Term Memory (LSTM) on the S&P 500 index daily closing prices and the results showed that CNNs could outperform LSTMs for financial time series.

Grob et al. showed that an ensembled model of a RNN and CNN could successfully predict intraday returns using correlated electricity contracts [5]. Rather than predicting the next time period price, the authors predicted the direction of the price using 3 level classification. The major benefit of their STaR model was that they were able to capture information from correlated electricity contracts through the CNN in their STaR model.

The Dual-Stage Attention-Based RNN (DA-RNN) model by Qin et al is considered one of the benchmarks for financial time series prediction as it utilizes index constituents to predict the future prices of the NASDAQ 100 [6]. Their concept uses the Encoder-Decoder framework and incorporates an Attention-Based to prevent the degradation of performance as the time window is lengthened.

Image Captioning is a new area of deep-learning that combines CNNs with RNNs to caption images. The primary focus of CNN and RNN ensembles have been for automatically generating captions for images [7, 8]. As of yet, there are very few papers that have applied image captioning to financial time series.

3. Methods

The models used in the paper are based on those covered by Grob et al [5], Qin et al [6,9], and Yunjey [10]. The DA-RNN model was tested by Qin et al on the NASDAQ 100 index and will be used as the baseline. Grob et al [5] described using a combined RNN and ST-CNN to create a new ensembled STaR model for electricity contracts. The LSTM is based on the RNN used in [5] but omits the dense layers. The CNN is a scaled down version of the CNN due to compute limitations. Because the NASDAQ dataset contains a

[1]

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

great deal more financial contracts than Grob's electricity contracts, an encoder - decoder framework similar to Image Captioning was used [10] for the CNN-LSTM.

3.1. Model Architecture

Dual-Stage Attention-Based RNN (DA-RNN). The DA-RNN model uses a two-staged model ensemble for time series predictions based on the paper by Qin et al [6]. The first stage the encoder is used to identify features. The second stage decoder selects the optimal features and applies them to generate a prediction. Figure 1 illustrates the DA-RNN model.

Encoder:

- Attention layer (Linear - TanH - Linear)
- Softmax
- LSTM

Decoder:

- Attention layer (Linear - TanH - Linear)
- Softmax
- LSTM
- Fully-connected layer

The model used the hyperparameters mentioned in [6,7]- batch size of 128, encoder hidden size of 64, decoder hidden size of 64, time window of 10.

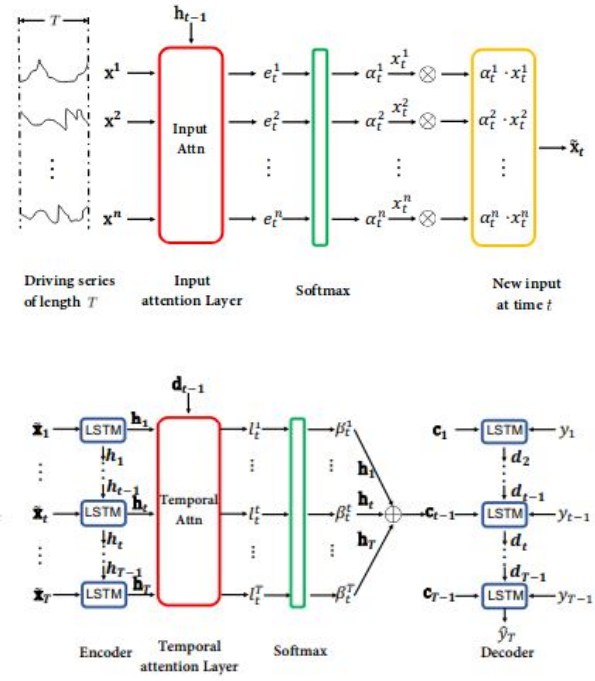

Figure 1. Dual-Stage Attention-Based RNN (DA-RNN) [6].

LSTM. The architecture for the RNN includes 2 layers of an LSTM using 100 hidden units each and a dropout. The original LSTM architecture from [5] can be seen in the top portion of the STaR model in Figure 2. The model used in this paper uses 1 dense layer as opposed to 2.

- LSTM (100 hidden units, 2 layers, TanH, dropout)
- 1x Dense (100 units) - ReLU - Dropout
- 1x Linear

The hyperparameters used were: hidden size of 100, time window of 100, feature size of 81 (number of companies), and batch size of 128.

CNN (CNN). The original ST-CNN architecture looked like the lower half of the STaR model seen in the bottom portion of Figure 2. Due to compute limitations, a linear layer was added to scale the width down to 10. The CNN architecture that was used contained the following:

- 2x CNN (channel=64, s=1, kernel=(3,5), padding=1) - ReLU - Dropout
- 2x Dense - ReLU - Dropout
- 1x Linear

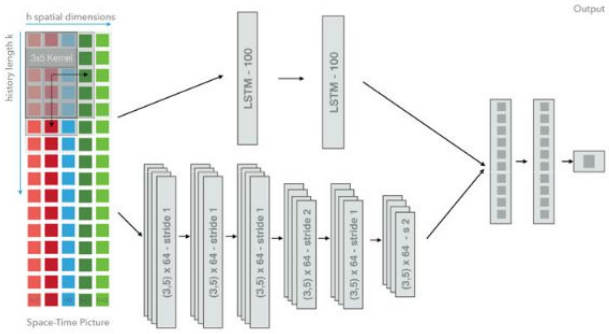

Figure 2. Space-Time Convolutional and Recurrent Neural Network (STaR) [5].

The hyperparameters used were: channel size of 64, time window of 100, feature size of 81 (number of companies), and batch size of 128.

Encoder Decoder CNN LSTM. The encoder decoder CNN-LSTM model is based on Image Captioning [8]. The encoder is a CNN that is used to extract a feature

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

vector for a given time sequence. That feature vector in then fed into the LSTM decoder where the model is conditioned to predict the future value of the NASDAQ 100 based on the previous NASDAQ 100 values and the values of the constituents. Figure 3 depicts the Image Captioning architecture that was the basis for the Encoder Decoder CNN LSTM.

Encoder:

- 2x CNN (channel=64, s=1, kernel=(3,5), padding=1) - ReLU - Dropout
- 1x Dense - ReLU - Dropout

Decoder:

- LSTM (100 hidden units, 2 layers, TanH, dropout)
- 1x Linear

The hyperparameters used were: channel size of 64, time window of 100, feature size of 81 (number of companies), channel size of 64, and batch size of 128.

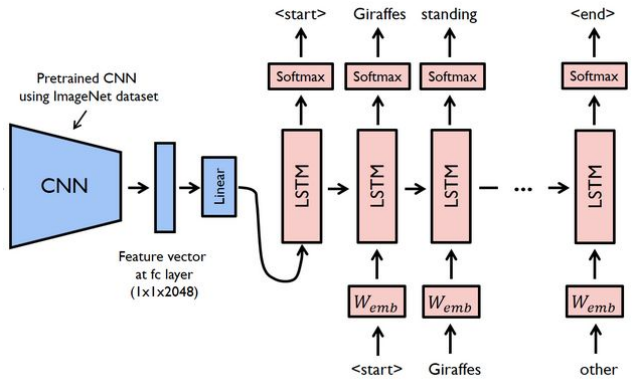

Figure 3. Image Captioning architecture [8].

3.2. Loss Function

The Mean Square Error loss function was used to evaluate model predictions against the true future value of the NDX index.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_{true} - Y_{pred})^2$$

3.3. Optimizer

An Adam optimizer is used to minimize a MSE loss for each model. The learning rate is decreased after 10,000 batches.

4. Dataset and Features

The above papers represent the most recent research methodologies on predicting financial time-series. This research expands on those papers by applying LSTM and CNN models to predict the price of the NASDAQ 100 index using features from its constituents. The following models will be evaluated:

- DA-RNN [6,9]
- LSTM [5]
- CNN [5]
- Dual-Stage CNN - LSTM [10]

The target index is the NASDAQ 100 – the top 100 domestic and international non-financial companies listed on the NASDAQ based on market cap.

The dataset is the original small dataset used in the DA-RNN paper [6] and includes 105 days worth of 1-minute OHLC bars of stock data from July 26, 2016 to December 22, 2016. Each day contains 390 data points except for November 25 (210 data points) and December 22 (180 data points).

On December 9, 2016, the annual index rebalance announcement was released. Four out of the 81 companies used in the dataset were dropped and the NASDAQ 100 price change is visible as a price jump towards the end of the test set.

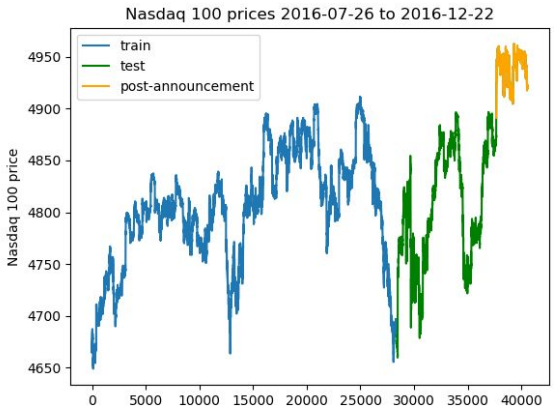

Figure 4. NASDAQ 100 prices.

Corporations with too much missing data were excluded leaving a total of 81 companies. For these remaining companies, linear interpolation was used to fill

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

in missing data. The time series data for each stock is originally from Yahoo Finance and includes:

- Open, high, low, close prices
- Trading volume

The training set consists of 28,391 time intervals and the test set consists of 12,169 time intervals. The target was normalized with the training mean subtracted from both the training and test target data.

For the CNN, the data input is shaped as (N, C, H, W) - with N representing the batch-size, the channels represent the stock data features (C=1, i.e. opening price), H is the time history window (H=100), and W is the number of companies (W=81). The training mean was subtracted for both the X and target train and test datasets.

For the LSTM, the data input is shaped as (N, T, W) - with N representing the batch-size, T is the time history window (T=100), and W is the number of companies (W=81). The training mean was subtracted for the target train and test dataset.

5. Experiments/Results/Discussion

5.1. Hyperparameters

The baseline parameters from the DA-RNN were used for the LSTM model. The best model was based off of smallest epoch that had the lowest error rate.

The DA-RNN parameters are: encoder hidden size =100, decoder hidden size = 100, T = 100, feature size =81, batch size = 128.

The LSTM parameters are: number of layers = 2, hidden size = 100, T = 100, feature size =81, batch size = 128

The CNN parameters are: input channel = 1, channel size = 64, T = 100, feature size = 81, batch size = 128.

The CNN-LSTM parameters are: input channel = 1, channel size = 64, T = 50, feature size = 81, batch size = 128. The CNN-LSTM uses a window size of T=50 rather than 100 due to memory constraints.

The CNN model did not converge despite changes to model architecture and different learning rates. Table 1 summarizes the results for the following model architectures:

- CNN 2-layer (3x5 kernel, S=1, P=1)
  1-layer (3x5 kernel, S=2, P=1)
- CNN 2-layer (3x5 kernel, S=1, P=1)
  1-layer (3x5 kernel, S=2, P=1)
  1-layer (3x5 kernel, S=1, P=1)
- CNN 2-layer (3x5 kernel, S=1, P=1)
  1-layer (3x5 kernel, S=2, P=1)
  1-layer (3x5 kernel, S=1, P=1)
  1-layer (3x5 kernel, S=2, P=1)

The optimal CNN model was the smallest 2-layer with a learning rate of 0.01.

The learning rates were also adjusted from 0.01, 0.05, to 0.1 to try and improve the loss convergence. The optimal learning rate is 0.01.

CNN Parameters	**Learning Rate**	**Train MSE**	**Test MSE**
CNN-2	0.01	2893.945	6839.386
CNN-2x1	0.01	2893.989	6840.339
CNN-2x1	0.05	2894.018	6842.493
CNN-2x1	0.10	2894.010	6847.120
CNN-2x2	0.01	2894.012	6844.917
CNN-2x2x1	0.01	2894.018	6842.333

Table 1. CNN hyperparameter search results.

5.2. Metrics

Because the output is a predicted future price, the Mean Squared Error (MSE) can also be used to gauge the model's accuracy. The model with the lowest overall test MSE is considered the best model for predicting the future NASDAQ 100 price.

Best Model	**Train MSE**	**Test MSE**
Da-RNN	19.326	266.691
LSTM	249.252	222.407
CNN	2893.985	6839.386
CNN-LSTM	126.890	21.452

Table 2. Train vs Test MSE.

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

The MSE was also calculated for the test period before the index rebalancing announcement and the period after. By comparing pre and post-rebalancing MSE, we can determine whether a model can successfully capture correlation information or if it is dependent on the NASDAQ 100 price history.

Test MSE	**Pre Announcement**	**Post Announcement**
DA-RNN	9.504	1085.411
LSTM	20.128	866.329
CNN	3043.172	18923.839
CNN-LSTM	12.238	50.922

Table 3. Pre and post rebalancing announcement test MSE.

5.3. Results

Baseline. The DA-RNN model's results matched the original paper. The training error converged after 50 epochs and the test predictions were matching fairly well until the index rebalance. The DA-RNN was not able to adjust to the new price range level and had a significantly worse post-rebalancing announcement MSE.

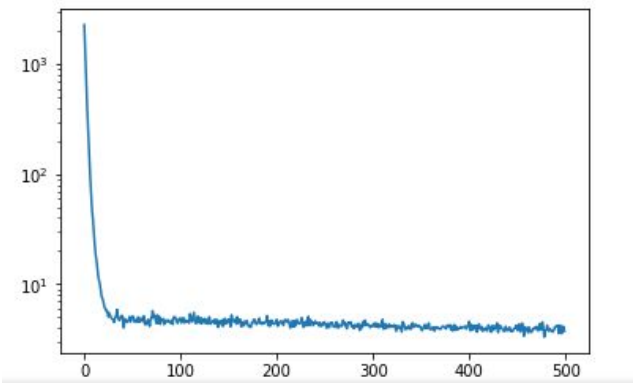

Figure 5. DA-RNN: Epoch training loss.

Figure 6. DA-RNN Test results - Epoch 50.

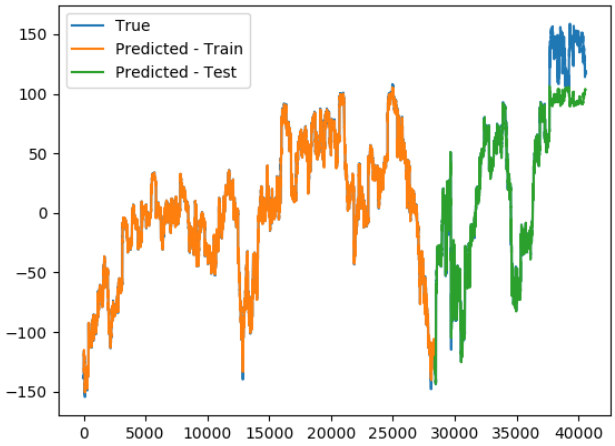

Figure 7. DA-RNN: Train vs test predictions.

LSTM. The LSTM had the second lowest overall test MSE. The optimal epoch was close to 25 and the epoch training loss graph showed over-fitting to be a major problem for this model. Visually, both the train and test predictions follow the same pattern as the NASDAQ 100 prices but are much noisier than the DA-RNN's predictions. The LSTM was able to capture some of the post-rebalancing price movements but not at an accurate level.

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

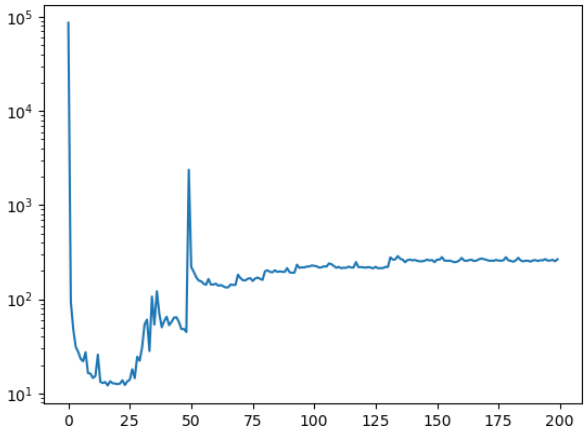

Figure 8. LSTM: Epoch training loss.

Figure 9. LSTM: Test results - Epoch 25.

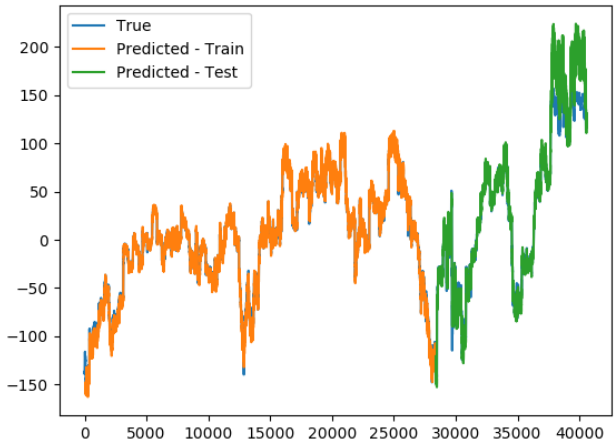

Figure 10. LSTM: Train vs test predictions.

CNN. The CNN as a stand-alone model for price prediction is a failure. While the training loss dropped after the first epoch, it pretty much did not show any improvement beyond the 2000 level. Figure 13 is a good indicator that the CNN optimized to reduce the MSE by basically predicting 0, or the training mean, which could be due to the price oscillation found in the training set.

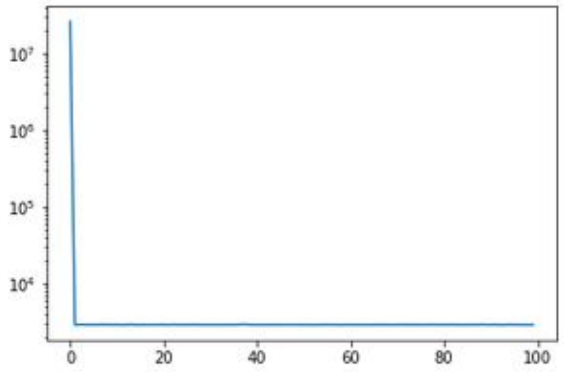

Figure 11. CNN: Epoch training loss.

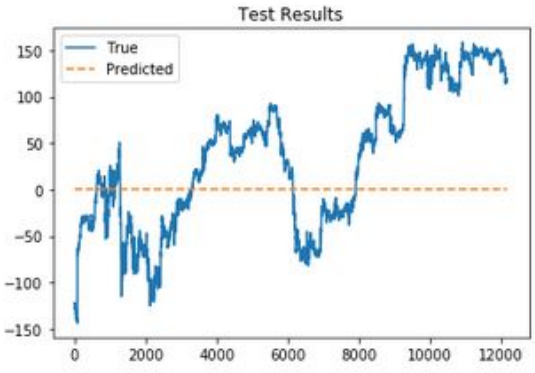

Figure 12. CNN: Test results - Epoch 100.

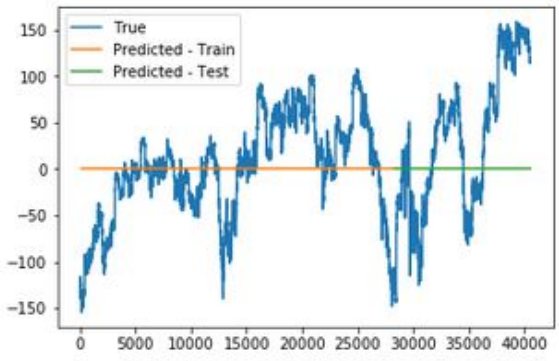

Figure 13. CNN: Train vs test predictions.

CNN-LSTM. The CNN-LSTM was the best model with the lowest overall test MSE and the lowest post-rebalancing announcement MSE. The MSE differential between the pre and post-rebalancing was surprising low which indicates that the CNN was able to

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

capture features that were not stock specific. The CNN-LSTM was also able to integrate those features into the LSTM without losing the price trend.

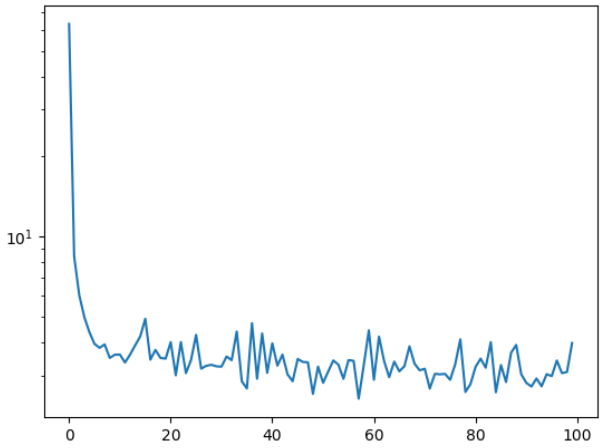

Figure 14. CNN-LSTM: Epoch training loss.

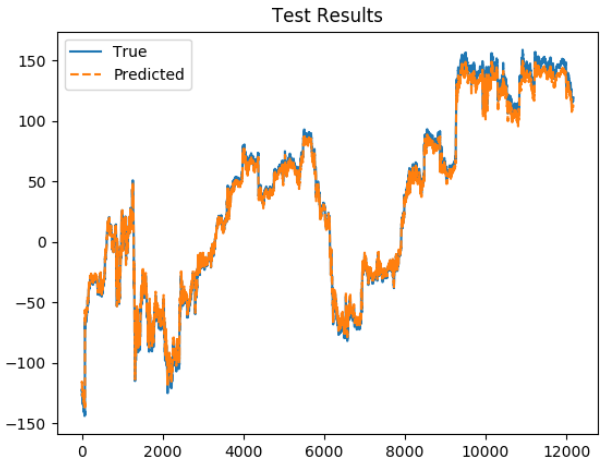

Figure 15. CNN: Test results - Epoch 100.

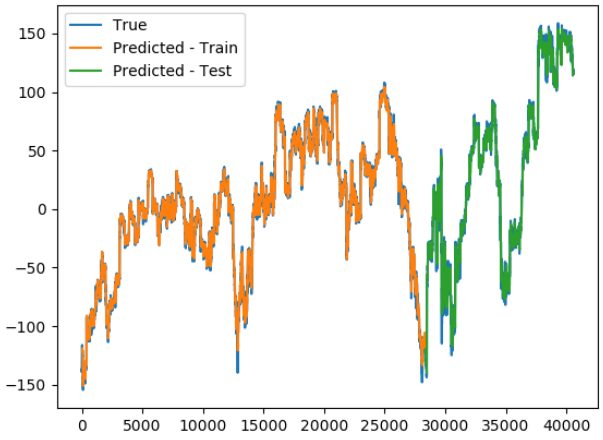

Figure 16. CNN-LSTM: Train vs test predictions.

6. Conclusion

The key takeaways are:

- Encoder Decoder frameworks look like they are able to capture multi-dimensional features not found in a singular time series. Both the DA-RNN and CNN-LSTM were able outperform the LSTM on pre-rebalancing announcement test data.
- The DA-RNN had the lowest pre-announcement MSE. The suggests that the encoder was able to successfully learn features that can help predict the future price. Those features are stock specific and the LSTM was not able to capture the price trend post-rebalancing announcement.
- The LSTM had the second lowest overall test MSE. This primarily came from its ability to track the index prices post-rebalancing announcement.
- The CNN model is not suited for predicting prices as it will converge to a value close to the mean in order to reduce the MSE.
- The CNN-LSTM was able to successfully

The best model for predicting the future price of the NASDAQ 100 is the Encoder Decoder CNN-LSTM. It had the lowest total MSE. The features learned in the encoder did not overwhelm the lstm prediction in the decoder. The post-rebalancing predictions were still able to capture the trend.

The DA-RNN had the lowest pre-rebalancing announcement test MSE. For a stationary set of data, the DA-RNN could potentially outperform the CNN-LSTM.

Stand alone CNN had a difficult time with predicting the future price. The optimization looked to minimize the MSE by predicting a value that was close to the mean value of the training set.

6.1. Future Improvements

With more time and computing power, I would work on improving the CNN model. The CNN model had a hard time improving the loss after the immediate epoch. A different loss function, target (log return rather than prices), or classification rather than regression might improve the model's predictive power. In the original paper [5], the authors used the STaR model for predicting the classification of price direction rather than the actual value.

Additional channels of features (i.e. closing price, high price, low price, and trade volume) could also be added

Tesa Ho, tesaho@stanford.edu
Stanford University, CS231n

to the model as well.

7. Acknowledgment

The code used in this project was based on the code by Chandler Zuo [9] and Yunjey Choi [10].

CS231n teaching assistants - Josh Beal, Michelle Guo, Pedro Gauzon, were also a tremendous help in debugging and helping to get the code running.

8. References

[1] Bao. W, Yue. J, Rao. Y. A Deep Learning Framework for Financial Time Series Using Stacked Auto-encoders and Long Term Short Memory. PLoS ONE 12(7): e0180944.

[2] Huang W., Nakamori Y., Wang SY., Forecasting Stock Market Movement Direction with Support Vector Machine. Computers & Operations Research, 2005.

[3] Moghaddam A., Moghaddam M., Esfandyarica M., Stock Market Index Prediction Using Artificial Neural Network, Journal of Economics, Finance and Administrative Science, 2016.

[4] Di Persio. L, Honchar. O. Artificial neural networks architectures for stock price prediction: Comparisons and applications, 2016.

[5] Grob W., Lange S., Bodecker J., Blum M., Predicting Time Series with Space-Time Convolutional and Recurrent Neural Networks. ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017.

[6] Qin, Y., Song, D., Cheng, H. Cheng, W., Jiang, G., Cottrell G. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. International Joint Conference on Artificial Intelligence (IJCAI), 2017.

[7] You Q., Jin H., Wang Z., Fang C.,Luo J. Image Captioning with Semantic Attention, 2016.

[8] Johnson J., Karpathy A., Fei-Fei L. DenseCap: Fully Convolutional Localization Networks for Dense Captioning, 2016.

[9] Zuo, Chandler. A PyTorch Example to use RNN for Financial Prediction

[10] Choi, Yunjey. PyTorch Tutorial 03-advanced. Image Captioning.

9. Code Resources

Data source:

NASDAQ 100 stock data

A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., Cottrell, G. *International Joint Conference on Artificial Intelligence (IJCAI)* , 2017

DA-RNN:

https://github.com/chandlerzuo/chandlerzuo.github.io/tree/master/codes/da_rnn

Image Captioning CNN-LSTM:

https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/03-advanced/image_captioning/model.py

Python packages:

Numpy
Pytorch
Pandas
Sklearn