Algorithmic Financial Trading with Deep Convolutional Neural Networks Time Series to Image Conversion Approach Paper Notes

Approach of converting to images :
In order to convert financial time series into 2-D images, 15 different technical indicators each with different parameter selections are utilized. Each indicator instance generates data for a 15 day period. As a result, 15x15 sized 2-D images are constructed.

Locality Requirements (Important did not previously consider):
Also the rows are ordered in such a way that similar indicators are clustered together to accomplish the locality requirements along the y-axis. As a result, 15x15 pixels sized images are generated and fed into the deep convolutional neural network.

Conclusion of the Project :
The proposed model outperformed Buy & Hold, common technical indicator based models, most widely used neural network, namely MLP, and the state of the art deep learning time series forecasting model, namely LSTM, on short and long out-of-sample periods. Even though, this is probably one of the first attempts using such an unconventional technique, we believe, the proposed model is promising. Moreover, parameter optimization and model fine tuning might even boost the performance even further.

Fundamental Vs Technical Analysis
For analyzing and developing inference models from financial data, there are two widely-adopted approaches: technical analysis and fundamental analysis [1]. Fundamental analysis can be implemented through examining company specific financial data such as balance sheet, cash flow, return on assets. Meanwhile, technical analysis can be implemented through analyzing past financial time series data using mathematical and/or rule-based modelling.

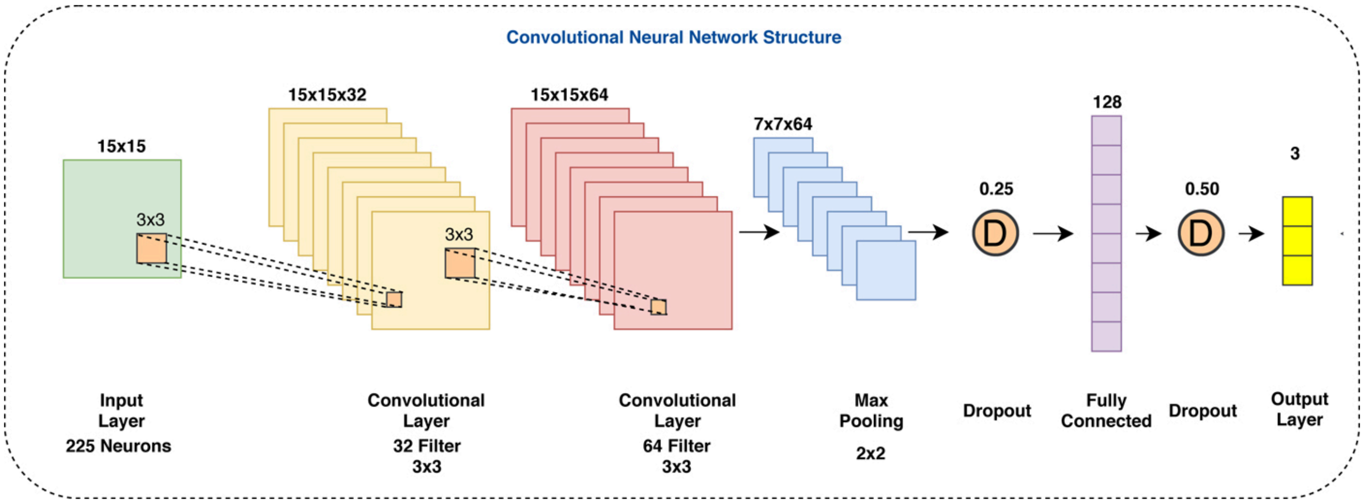
Side Note : Should I implement some sort of way to incorporate some fundamental data in the model like PE ratio etc.

More on Locality:
CNN is a feedforward ANN that takes its inputs as 2-D matrices. Unlike a fully connected neural network like MLP, the locality of data within the input vector (or matrix) is important. Hence, the neighboring data points within the matrix should be carefully chosen.
The order of the indicators is important, since different orderings will result in different image formations. To provide a consistent and meaningful image representation, we clustered indicator groups (oscillator or trend) and similar behaving indicators together or in close proximity.

Data Set

In our study, the daily stock prices of Dow 30 stocks and daily Exchange-Traded Fund (ETFs) prices are obtained from finance.yahoo.com for training and testing purposes. Stock / ETF prices between 1/1/2002 to 1/1/2017 are used for training and testing purposes. We adapted a sliding window with retraining approach where we chose a 5 year period for training and the following 1 year for testing, i.e. training period: 2002-2006, testing period: 2007. Then we moved both training and testing periods one year ahead, retrained the model and tested with the following year

Architecture:



Labelling Algorithm:

```
def labelling(close_price_list: List[float]) -> List[str]:
    window_size = 11  # days
    number_of_days_in_file = len(close_price_list)

    # Initialize all labels as "HOLD"
    result_labels = ["HOLD"] * number_of_days_in_file

    counter_row = 0

    while counter_row < number_of_days_in_file:
        counter_row += 1

        if counter_row > window_size:
            window_begin_index = counter_row - window_size
            window_end_index = window_begin_index + window_size - 1
            window_middle_index = (window_begin_index + window_end_index) //
2

            min_val = inf
            max_val = -inf
            min_index = -1
            max_index = -1

            for i in range(window_begin_index, window_end_index + 1):
                number = close_price_list[i]

                if number < min_val:
                    min_val = number
                    min_index = i

                if number > max_val:
                    max_val = number
                    max_index = i

            if max_index == window_middle_index:
                result = "SELL"
            elif min_index == window_middle_index:
                result = "BUY"
            else:
                result = "HOLD"

            result_labels[window_middle_index] = result

    return result_labels
```

Calculating Indicators :
Refer to the Appendix

My Conclusions:
The architecture with the number of layers etc seems good without being overly deep. The data strategy looks solid, however the way of assigning labels to the test set seems rudimentry. No information was provided about the optimizer and non linearity however i feel that a normal or leaky relu with adam would be the best bet.