Application of Deep Convolutional networks applied to Portfolio Optimization

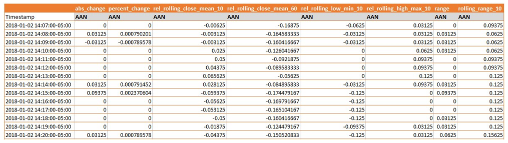
Author: Roy Justus
Email: roy@justushome.net
Created for Stanford CS230, Winter'20

Sponsored By:
polygon.io Financial Market Data APIs
Amazon EC2

Project Objective:
Create a portfolio allocation strategy that provide better than market returns by applying deep learning to near-past data from hundreds of equities.

Underlying Hypothesis:
Where correlations exist between prices of equities, the mechanisms of actions of some subset of these correlations are likely to take effect with some delay allowing some equities to serve as weak leading indicators of the behavior of others.

Input Features:
The prediction inputs are a 60 timestep by 346 equity by 8 feature matrix generated for each minute of each trading day based on the preceding 60 trading minutes. These features are selected to be nearly stationary. (Not subject to periodic or long term trends)
The features provide current and near-past summaries of the market state across all 346 equities.

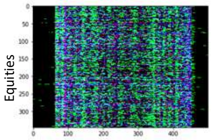


	abs_change	percent_change	rel_rolling_close_mean_10	rel_rolling_close_mean_60	rel_rolling_low_min_10	rel_rolling_high_max_10	range	rolling_range_10
Timestamp	AAN	AAN	AAN	AAN	AAN	AAN	AAN	AAN
2018-01-02 14:07:00-05:00	0	0	-0.00625	-0.16875	-0.0625	0.03125	0	0.09375
2018-01-02 14:08:00-05:00	0.03125	0.000790201	-0.003125	-0.164583333	-0.03125	0.03125	0.03125	0.0625
2018-01-02 14:09:00-05:00	-0.03125	-0.000789578	-0.003125	-0.160416667	-0.03125	0.03125	0.03125	0.0625
2018-01-02 14:10:00-05:00	0	0	0.025	-0.126041667	0	0.0625	0.03125	0.0625
2018-01-02 14:11:00-05:00	0	0	0.05	-0.0921875	0	0.09375	0	0.09375
2018-01-02 14:12:00-05:00	0	0	0.04375	-0.089583333	0	0.09375	0	0.09375
2018-01-02 14:13:00-05:00	0	0	0.065625	-0.05625	0	0.125	0	0.125
2018-01-02 14:14:00-05:00	0.03125	0.000791452	0.028125	-0.084895833	-0.03125	0.09375	0.03125	0.125
2018-01-02 14:15:00-05:00	0.09375	0.002370604	-0.059375	-0.174479167	-0.125	0	0.09375	0.125
2018-01-02 14:16:00-05:00	0	0	-0.05625	-0.169791667	-0.125	0	0	0.125
2018-01-02 14:17:00-05:00	0	0	-0.053125	-0.165104167	-0.125	0	0	0.125
2018-01-02 14:18:00-05:00	0	0	-0.05	-0.160416667	-0.125	0	0.03125	0.125
2018-01-02 14:19:00-05:00	0	0	-0.01875	-0.124479167	-0.09375	0.03125	0.03125	0.125
2018-01-02 14:20:00-05:00	0.03125	0.000789578	-0.04375	-0.150520833	-0.125	0.03125	0.0625	0.15625

Name	Time Period
Absolute Change	1 minute
Percent Change	1 minute
Relative Rolling Mean (10)	10 minutes
Relative Rolling Mean (60)	60 minutes
Relative Rolling Low (10)	10 minutes
Relative Rolling High (10)	10 minutes

Pre-processing:
Each feature of each input equity is standardized to mean of 0 and a variance of one.

Analogy to 2D images:
The architecture of my model treats this data as a 2D image with 8 channels. The Image to the left visualizes 3 of these channels to provide intuition as to how this is processed in a Convolutional Network.



Minutes of trading day

Legend:
R: Percent Change
G: Relative Rolling Mean
B: Range

Input Data Factsheet:

Time Period: 5 years
Resolution: per minute
Number of datapoints: 434433
Number of equities: 364 Midcap S&P400 stocks
Raw Data Features: Open, Close, High, Low, Volume
Total size: 26.9GB uncompressed
Source: All data provided by Polygon.io
Stocks Optimized: MANH, CLI, FULT, CMC, GGG, KMT, ZBRA, OI, TCBI, GDOT, JCOM

Model Architecture:

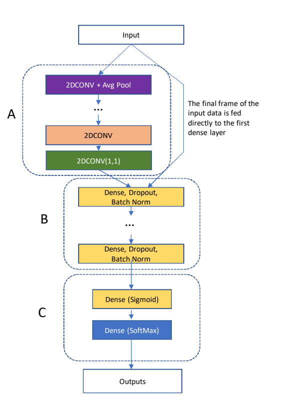
A: Convolutional Feature Extractor
The Convolutional layers extract information from the complex Input data, finding trends and correlations that contain useful information

Layer	Filters	Kernel size	Stride	Padding	Pooling
1	64	3, 1	1	Valid	AVG
2	64	2, 1	1	Valid	AVG
3	128	2, 1	1	Valid	-
4	256	2, 346	1	Valid	-
5	64	1, 1	1	-	-

B: Dense Market State Representation
The features are passed into a number of dense layers that map them to the expected profitability of each of the stocks in the portfolio

Layer	Nodes	Dropout	Activation	Batch Norm
1	1050	0.2	Relu	Yes
2	800	0.2	Relu	Yes
3	1100	0.2	Relu	Yes
4	1300	0.2	Relu	Yes
5	88	-	Relu	Yes
6	11	-	Sigmoid	No
Output	11	-	SoftMax	-

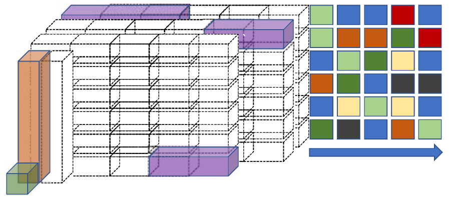
Illustration



The final frame of the input data is fed directly to the first dense layer

Convolutional Intuition:
This model works primarily because the first three convolutional layers are able to pick up complex features of an individual time series (using filters common to all equities) which are then convolved with filters which find the relationships between what happens to different equities at neighboring points in time, respecting the temporal relationship just at typical CNN models respect special structure.

The atypical, tall structure of layer 4 (2,346) ensures no vertical strides are taken and as such the filters learn only by scanning forward over the timesteps in the input data, each parameter of each filter being permanently associated with one of the input equities.



Limitations:
- The optimization model does not consider any execution costs and assumes that it can re-allocate the portfolio every minute without incurring any trading costs, market impact or other costs not reflected in the quoted close price of the stock.
- The Algorithm assumes that it can purchase fractional shares
- The Algorithm is not able to hold cash between periods and must in vest in one or more of the 11 portfolio shares each period.
- The Evaluation Algorithm assumes that it can sell it's current holdings and purchase new shares at the quoted close price for each period

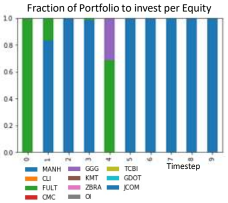
Loss Function:
In order to efficiently optimize for portfolio profitability the loss function is simply the sum within a particular timestep of the portfolio allocation to each stock multiplied elementwise with the corresponding change to that equities value.
A second "capital preserving" term is used with Beta = 1 which causes losses to be penalized twice as much as gains are rewarded.

$$L = -\sum_{n=0}^{s}\{(y_n * \hat{a}_n) - \beta min(0, y_n * \hat{a}_n)\}$$

n = index of an equity in the list of portfolio stocks(range: 0 to $s-1$)
$\mathbf{y}$ = column vector (length n) of stock price increases in next minute
y_n = Scalar true increase in stock price of equity n in next minute
$\hat{\mathbf{a}}$ = column vector (length n) of recommended fractional allocations.
$\hat{a}_n$ = Scalar recommended fractional allocation of portfolio funds to equity n
β = parameter for additional risk aversion.

Output Data:
The resulting output consists of a time series of fractional portfolio allocations. Often times these are direction to place all of the portfolio's value in a single equity and at other times it distributes the equities among 2-3. The graph below provides a sample of the allocations over time:

Fraction of Portfolio to invest per Equity

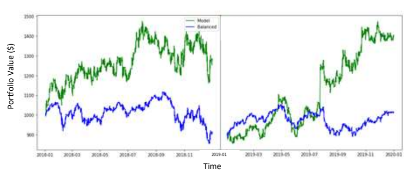


Results:
The resulting model is able to optimize a portfolio of 11 stocks to produce a return well in excess of that which would be earned by investing equally in each of the 11 stocks.

Experiment	Model	Training Data	Test Data	Percent Return
Training / Validation	Balanced Portfolio	-	2018 (97285 records)	27.9%
	Deep CNN	2015-2017 (239469 records)		-9.1%
Testing	Balanced Portfolio	-	2019 (97679 records)	73.6%
	Deep CNN	2015-2018 (336754 records)		16.9%

Below you can see the training and test performance plotted for one year of trading each:

Training / Validation **Testing**



As expected we see good results in the validation dataset, unsurprising as the model hyper parameters such as dropout having been tuned to allow the model to generalize well.
This test run, the first use of my 2019 dataset, also provided strong performance, missing a potentially valuable investment in ZBRA early in 2019 (ZBRA rose over 53% in 4 months and my model failed to invest even 0.001% there) but correctly capitalizing on a spike in MANH later in the year.

Discussion:
Significance of results:
Overall the results of this experiment have been very promising as they clearly show that there is some usable information about the future performance of stocks available by applying data about the current and past state of related equities.

Conclusion of model effectiveness:
The results indicate that the Deep Leaning architecture used in this project provides a compelling solution for efficiently extracting features across hundreds of equities with relatively little pre-processing of data.

Possible adjustment for major sources of error:
By adjusting the loss aversion of the model though tuning of the beta parameter of the loss function and by adding an option to hold a portion of the portfolio as cash it may be possible to partly mitigate the periods of loss seen in the current model.

Consideration for major sources of excess returns:
It is clear that the model cannot predict all cases of exceptional returns on a single stock, for example ZBRA significantly outperformed the portfolio's other members in early 2019, leading to our model underperforming the portfolio average because it was not invested there. In other cases such as the large value spike in late July 2019 it is able to capitalize on signals to shift allocation to the appropriate stock and generate excess returns.

Conclusion:
Although this is a promising start, the translation of this information into a practical trading strategy required continued effort, likely along the lines of developing a Deep Reinforcement learning algorithm to learn when it is profitable to act on this information (i.e. execution costs are less than expected profit)

Future Direction:
With additional time and compute resources I propose the following follow-on work:
- Add a 12th asset allocation for Cash which does not gain or lose value each period in order that the model may pull money out of the market if it detects signals of widespread decline.
- Exploration of the benefits of re-training the model on timescales shorter than one year, for example re-training monthly or daily to ensure recent data is included in the training set.
- Incorporation of this model into a Reinforcement Learning Agent with the objective of optimizing allocation even given trading costs, market impact and other exclusions Additional Hyperparameter search, particularly in the convolutional layers

References:
K. Khare, O. Darekar, P. Gupta and V. Z. Attar, "Short term stock price prediction using deep learning," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017, pp. 482-486.

L. Sayavong, Z. Wu and S. Chalita, "Research on Stock Price Prediction Method Based on Convolutional Neural Network," *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, Jishou, China, 2019, pp. 173-176.

A. Ariyo, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, Cambridge, 2014, pp. 106-112.

Y. Hu and S. Lin, "Deep Reinforcement Learning for Optimizing Finance Portfolio Management," *2019 Amity International Conference on Artificial Intelligence (AICAI)*, Dubai, United Arab Emirates, 2019, pp. 14-20.

T. Sanboon, K. Keatruangkamala and S. Jaiyen, "A Deep Learning Model for Predicting Buy and Sell Recommendations in Stock Exchange of Thailand using Long Short-Term Memory," *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, Singapore, 2019, pp. 757-760.

J. Eapen, D. Bein and A. Verma, "Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2019, pp. 0264-0270.

Thushan Ganegedara, "Market Predictions with LSTM in Python" January 1st, 2020. Retrieved from the internet at: *https://www.datacamp.com/community/tutorials/lstm-python-stock-market*

Yacoub Ahmed, "Predicting stock prices using deep learning" Oct 11, 2019. Retrieved from the internet at: *https://towardsdatascience.com/getting-rich-quick-with-machine-learning-and-stock-market-predictions-696802da94fe*

Presentation Link: https://youtu.be/XqRoQfdqO4c