Cross Asset Generalization Between S&P and NASDAQ

Nicholas Chen

STAT 430

Spring 2019

This study approaches price change prediction as a regression problem with cross-asset trained neural networks. Several network architectures are proposed and evaluated, then compared with an originally formulated benchmark for financial regression. While RNNs achieved the lowest error, all models performed extremely well in comparison with the benchmark, indicating that cross-asset generalization is a promising solution to limited data access.

I – Introduction

Much of this class's work has been confined to a traditional train/test paradigm of training on a subset of a single asset's historical data, then testing on a disjoint time frame of that same asset. However, such a workflow requires that a new model be trained on new data for every stock, option, and bond that an investor wishes to trade. Due to both hardware and monetary constraints (intraday data is extremely expensive for a student!), a single, generalizable classifier would be highly preferable.

This study aims to determine whether such a cross-asset predictor is feasible. That is, can a model trained on one asset retain predictive power when applied to a different asset? Ideally, similar underlying psychological market trends can be observed in any stock so an investor would need train only a single model. The traditional method of technical analysis - searching for visual recurring trends in stock prices - gives credence to the viability of cross asset generalization.

This study also attempts to bridge the gap between theory and practice in financial machine learning. In addition to the direction of price movement, a trader should be strongly interested in the magnitude as well. So far, we've generally examined the former in a classification framework as an academic challenge; I decided to instead evaluate the precision of regression in financial applications. Converting the output from discrete to continuous can provide valuable insights for confidence and bet sizing.

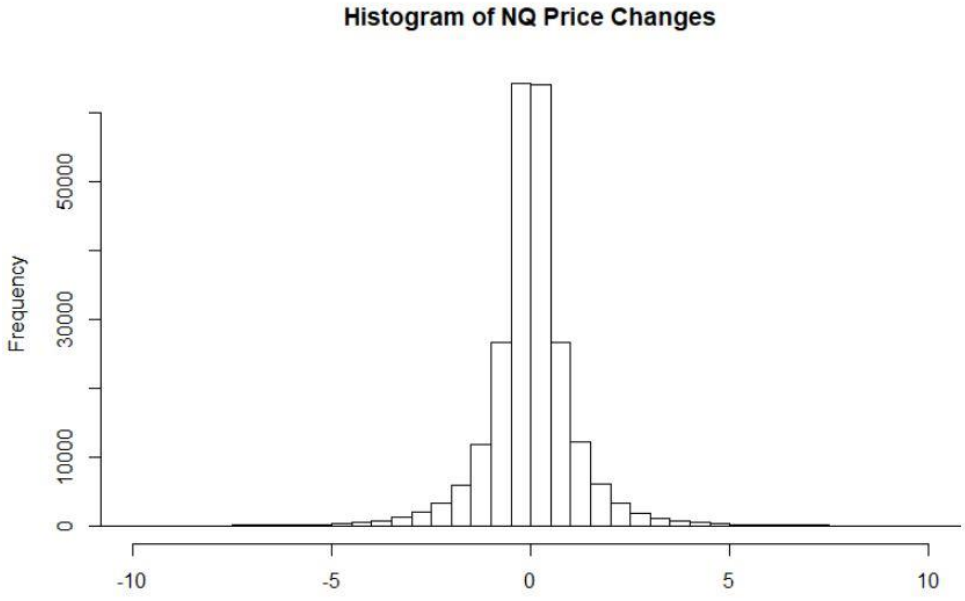
II – Dataset

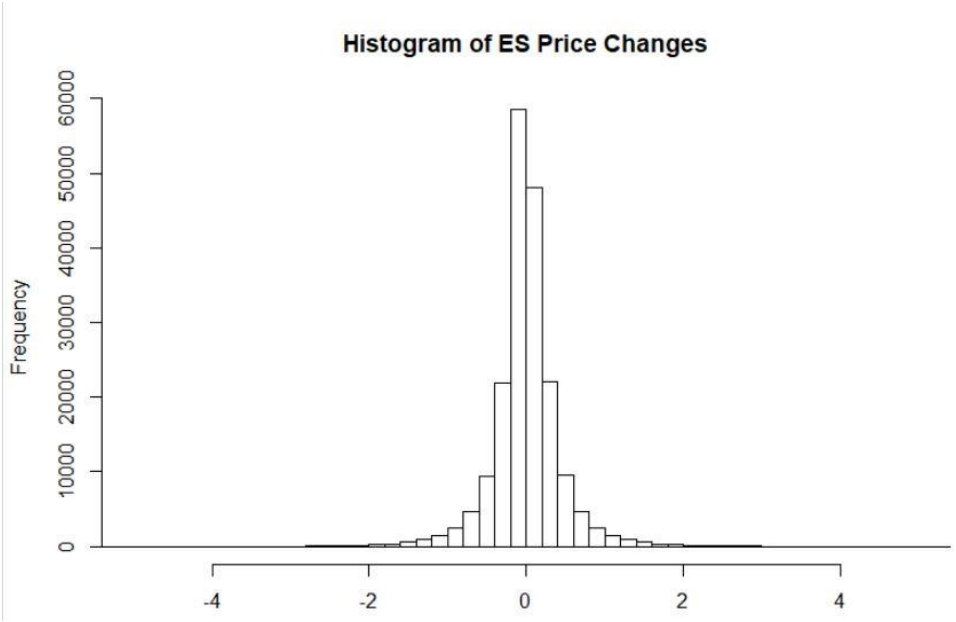
The dataset is one year of minute data on E-mini S&P 500 and E-mini NASDAQ 100 futures contracts provided by Professor Hua. From this data, I extract simple moving average and exponentially weighted moving averages of the VWAP for a variety of windows (d), which yields an (n,d,2) feature matrix. As per AFML (Lopez de Prado, 2018), I perform fractional differentiation on each column to achieve stationarity while retaining as much memory as possible, then normalize the data to mean = 0, standard deviation = 1. I don't normalize the Y values, however, because I want the regression's output to have real world significance as actual price change estimation.

III – Exploratory Analysis

Before training models, I needed to devise benchmarks for comparison. Though we have routinely evaluated classifiers by their performance relative to majority guessing, regression doesn't have such a clearly defined benchmark.

I immediately graphed the price changes as a histogram to visualize the label distribution.





Fortunately, both target value distributions appear to be approximately normal with mean = 0.

I assume that price changes follow a normal distribution. I will compare my model's mean absolute error to the expected MAE of a Gaussian function$\sqrt{\frac{2}{\pi}}\sigma$ -. If a model cannot beat this threshold, it effectively hasn't learned anything. Conversely, a MAE of less than that threshold implies that the model outperforms guessing. As calculated below, the ES MAE threshold is 1.6, while the NQ threshold is 4.3.

```
> sd(nq_Y, na.rm=T)*sqrt(2/pi)
[1] 4.294979
> sd(es_Y, na.rm=T)*sqrt(2/pi)
[1] 1.584785
```

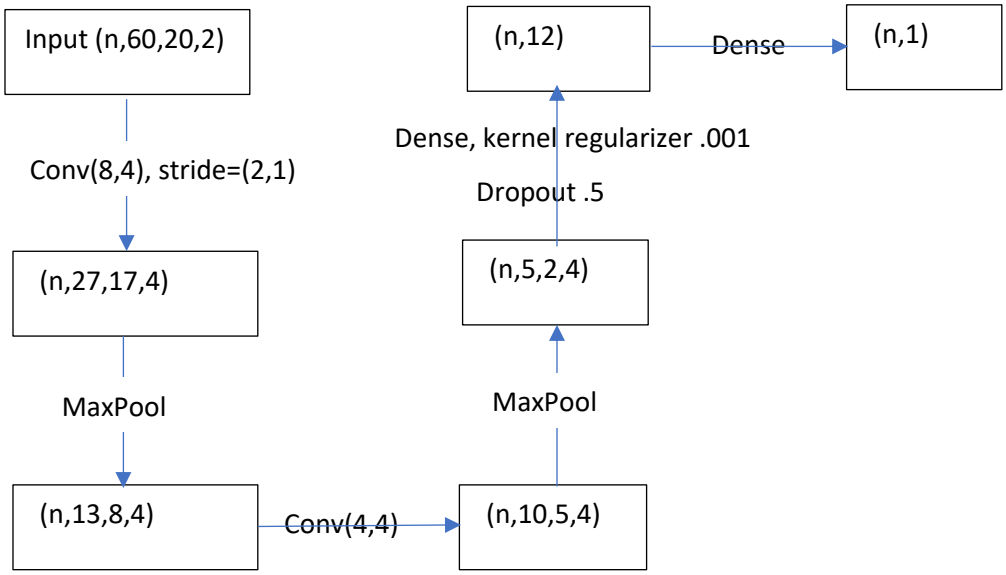
IV – Model Design and Training

Here, I will explain three neural net models that I ended up using – shallow CNN, deep CNN, and RNN.

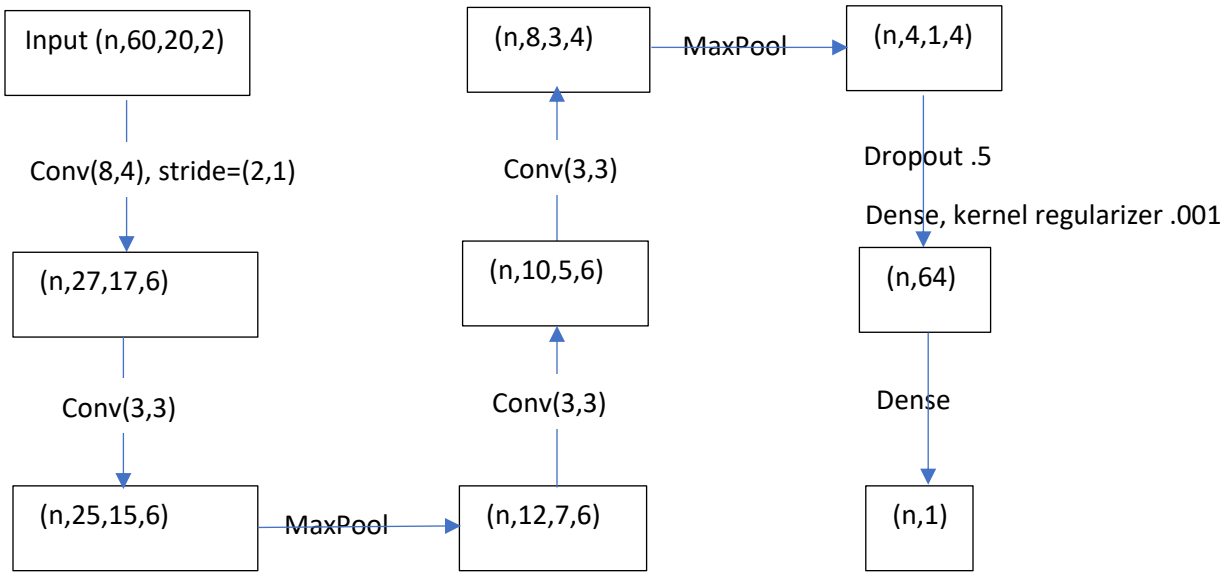
Shallow CNN/Deep CNN

When designing my NNs, my general framework is as follows, suggested by another course – I start by increasing model complexity until training error reaches roughly (the model can memorize the training date, then slowly remove layers, add dropout, and decrease the size of fully connected layers until validation loss is minimized. Initial design is very heuristic based, and I tried a variety of structures; in this case, the input data is (n,60,20,2), so I start with a rectangular filter and vertical stride of 2 to aggressively cut the vertical axis.

Shallow CNN – 1025 Parameters

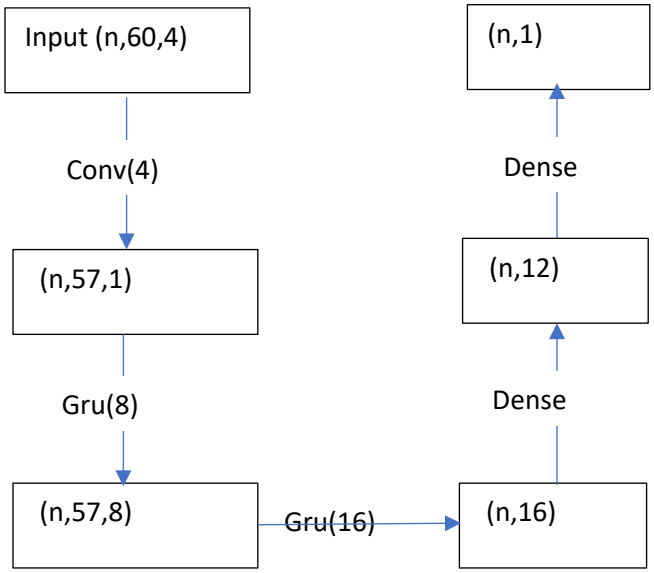


Deep CNN – 2423 parameters



RNN – 1818 parameters

I have no experience in designing RNNs so I kept it simple. Note that I first combine the X data channels via concatenation so that the input is of size (n,60,40).



For training, I split the dataset into 60% training, 20% validation, and 20% test. The validation set was used in early stopping and model checkpoint callbacks. I terminated training early if validation loss hadn't decreased for 5 epochs, and continually saved the model with the best validation loss.

Training hyperparameters: 20 epochs, batch size – 64, 250 steps per epoch

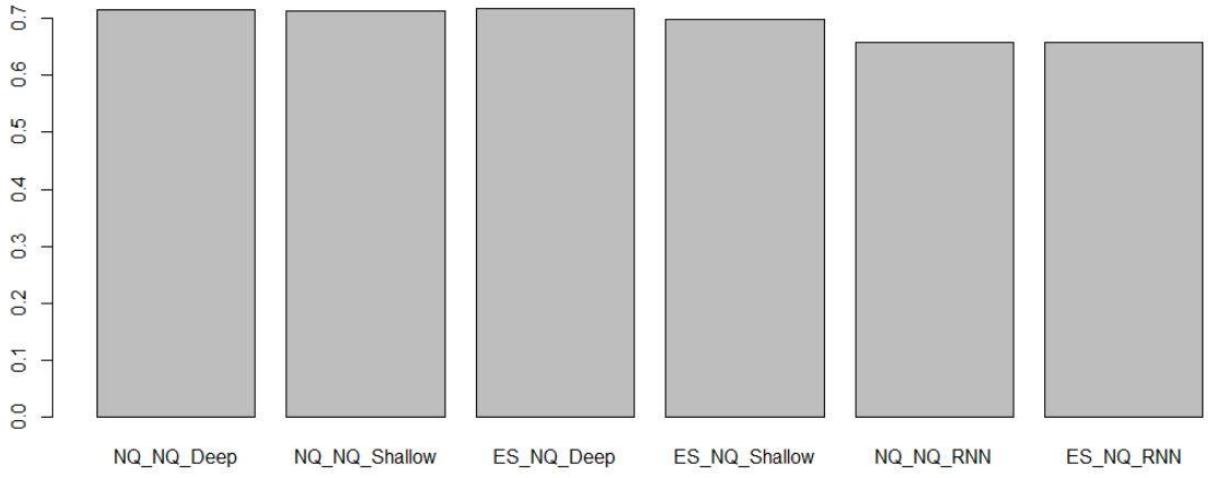
V – Regression Results

To reiterate my workflow – I took the minute data Professor Hua provided, then preprocessed the VWAP columns to generate 20 simple moving average features and 20 exponential moving average features. I then performed fractional differentiation on each column. This yielded the X datasets of dimension (n, 20, 2). The corresponding Y label for a given X time is the price change over the following minute. This process was repeated for both ES and NQ contracts.

Each CNN classifies Y[i] with X[i-59:I, ,], of size (n, 60, 20, 2), while the RNN concatenates the two channels to size (n, 40) and memory of 60 sequences. I have a total of 6 models – three model types trained on two different assets.

MAE for predicting NQ -

NQ_Deep	.716
NQ_Shallow	.713
ES_Deep	.717
ES_Shallow	.700
NQ_RNN	.659
ES_RNN	.659

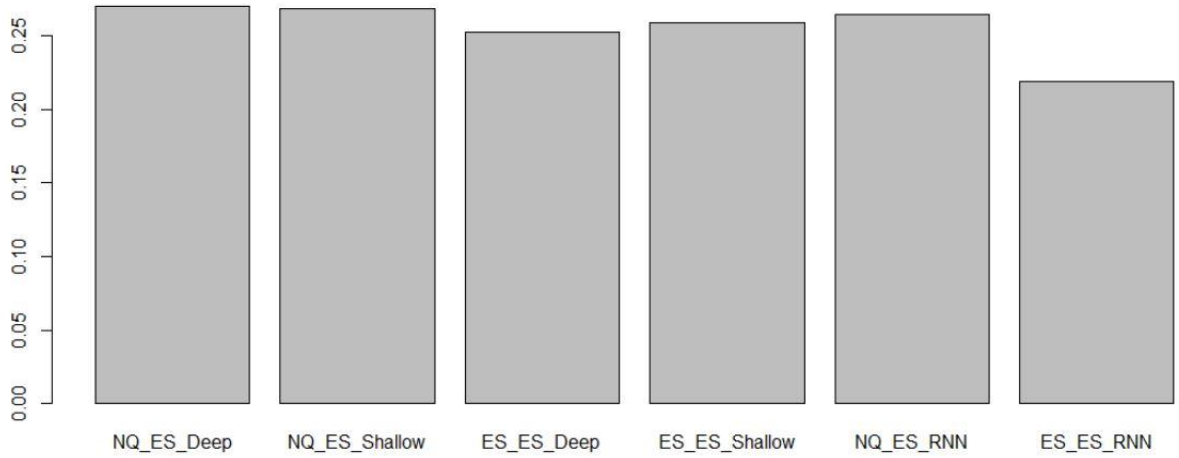


(note that the prefix indicates which asset the model was trained on, suffix is the tested asset)

Compared to the Gaussian RV expected MAE threshold of 4.3, all models performed extremely well. Even the worst performing model – NQ_Deep – yields an 86% reduction in MAE compared to random guess. The RNNs performed the best, and cross-asset trained models perform about as well as their counterparts (slightly better, even). Interestingly the cross-asset shallow network performed significantly better than the cross-asset deep network.

MAE for predicting ES -

NQ_Deep	.270
NQ_Shallow	.268
ES_Deep	.253
ES_Shallow	.259
NQ_RNN	.264
ES_RNN	.219



The ES prediction results tell a similar story. Again, all models were successful with even the worst model – NQ_Deep – decreasing MAE by 83% compared with random guessing. This time, the cross-trained RNN performs noticeably worse, but still in line with the CNNs, but the self-trained RNN is the best by far. Again, we see that the cross trained shallow model outperforms the cross trained deep model.

VI - Discussion

It makes sense that the RNN performed best in this study. The data format seems intuitively ideal for an RNN, which is specifically designed for temporally organized sequences. CNNs, on the other hand, need to discover the time series relationship on their own. Because I'm feeding the nets 60 minutes of previous information, the RNN is most naturally inclined to take advantage of this inherently structured data.

I am extremely pleased to find that cross-asset trained models performed just as well as their self-trained competitors. This indicates that the nets successfully learned generalizable price history trends. One personally significant implication is that I would only need to buy intraday data for a single asset, devote computational resources to training a single neural network to classify/run regression, then use that one model on any number of assets. This practical discovery is extremely exciting and makes algorithmic trading much more accessible for me.

Another critical observation was that the shallow models tended to outperform deep models in cross-asset performance. This is consistent with machine learning theory, which claims that model expressivity decreases approximation error but increases generalization error.

VII - Conclusion

This study indicates that cross asset generalization is a promising strategy for developing models with finite data access. Furthermore, regression analysis could be desirable for its applications to bet sizing and portfolio management. Moving forward, I would generally use simple models and the RNN class was most successful for cross asset generalization on this dataset. I intend to implement a basic backtesting simulator to explore the profitability of these strategies and better assess the effectiveness of regression as opposed to classification. I would also like to explore the optimal prediction time period – perhaps hourly trading would increase the signal to noise ratio by drawing information from a wider variety of market participants.

VIII - Acknowledgement

Professor Hua - I specifically modified your data generator code for this project. More generally, thanks for teaching this class! It introduced an incredibly broad array of machine learning technologies and your shared code was a great reference to better understand the material. Some of the topics were intimidating and challenging at first, but it retroactively feels immensely rewarding. I hope to use what I've learned from your course for years to come.

Appendix – R Code: