AVA - Advanced Volatility Arbitrage

Alex Fine

June 2019

I would like to thank Guy Wuollet, Andy Bromberg, and Cory Levy for their help and guidance.

For those interested in skimming this paper, I recommend sections 1 (abstract), 3.1-3.2 (problem & solution), 6.2-6.3 (performance), and 8 (next steps).

1 Abstract

AVA is an AI powered prediction technology which can be used for volatility arbitrage in order to stabilize high risk assets and generate alpha. The technology is currently applied to cryptocurrencies, however it can be applied more broadly to any volatile asset class. The output of AVA is a single number which indicates with high accuracy where the market is expected to move over the next n minutes. Beneath this output lies a highly modular & dynamic technical framework which allows for real-time predictions with a data latency of less than 15 seconds.

Given the modularity of the AVA system it can be used for a variety of tasks beyond the initially intended volatility arbitrage use-case. The system can be used to research purposes on the theoretical limits of predictive intelligence, serve as the core of a Reinforced Learning based trading strategy, and assist fundamental traders in entering markets at opportune times. The versatility of the AVA technology is discussed further in section 8.

2 Background

2.1 Story

The idea for AVA first started to form in February of 2018. At the time I was a freshmen at Stanford studying mathematics and computer science, with a focus on financial mathematics and artificial intelligence. My background was in tech and I had just sold my first company (a machine learning & computer vision company). I wanted to invest and figured I could figure out how to do so intelligently. I was initially turned away from cryptocurrencies due to the seemingly baseless nature of the asset class. However, given the big crypto boom, I figured I would at least study those markets in order to understand them. Upon studying them I realized that massive market inefficiencies existed. The markets would make 100 bip moves 50-100 times a day. Additionally, it seemed to me that there were nuanced patterns surrounding a portion of those moves. I figured if I could design an automated way to capitalize on a small portion of those moves with high reliability it could make a decent amount of money.

Additionally, I was anxious to get back into real technical coding. In the two years prior I was a more outward facing CEO & felt disconnected from the technology my team was developing. I intrinsically love to play with math and saw this project as a unique opportunity to do something I love.

Fast forward to the summer of the 2018, and I found myself living in Mallorca, Spain, with a considerable amount of free time. I decided to take advantage of that time. However, I was acutely aware of the fact that I had just finished my freshmen year of college, and had zero finance experience, non-the-less experience in building sophisticated infrastructures for AI based quantitative

trading. Thus, I decided to design and build the infrastructure from scratch using a highly modular methodology. I set up the proper data cleansing and retrieval pipelines so all training & trading could occur seamlessly. From there I designed all systems to be easily replaceable. For example, the way the system is built it is very easy to try a different ML model and test if it performs better. It is also incredibly easy to implement a new trading strategy on top of the AI output prediction. Similarly it is not difficult to append a new data source to the RNN inputs. Even the time scale the RNN predicts on is easily adjustable.

I designed the system with the thought that "today I don't know how to build a good trading strategy, or even an efficient RNN. However, I will learn. And when I do, I want to be able to easily implement those learnings."

2.2 Philosophical Underpinnings

Traditional asset classes, like stocks, bonds, mutual funds, ETF's, fiat currencies, etc. are valued primarily through two mechanisms. The market cap is a function of current real value & perceived future value. The weight between these two values holds somewhere within a beta distribution of Beta(8,2). The actual true value of a company can be measured through formal processes like DCF's or more simply as a function on their multiples. The future value of the company is derived through a process of forecasting how the company will perform in the future under new and unforeseen market conditions. How the company's product will grow or shrink. Again, there are very traditional mechanisms used to predict a company's future profits. At the end of the day, the market cap is going to be primarily based upon the fundamental value of the company. It will fluctuate somewhat around the real value, but the efficient market hypothesis states that the market will value the asset accurately.

Cryptocurrencies operate entirely differently.

Unlike traditional asset classes, cryptocurrencies are valued differently. The market cap of a cryptocurrency can still be modeled as a function of current real value & perceived future value. However, the relative weighting compared to traditional asset classes is vastly skewed, many standard deviations from the norm. Cryptocurrencies hold very little real value. Very few people actually pay for goods with Bitcoin and Ethereum. Few vendors even accept them as valid payment methods. This effect causes the total value of cryptocurrencies like Bitcoin and Ethereum is highly speculative. Additionally, unlike a fiat currency like the Yen or USD, cryptocurrencies are not tied to a federal government. There can be no quantitative analysis of the probability that the governing body controlling Bitcoin collapses. There is no governing body. This difference further forces cryptocurrencies to be valued by public perception over true valuation metrics. Because of this fundamental difference we must value this asset class differently.

When valuing assets we like to think that we compare our model for the real value of the asset, compare it with the current value, and buy or short accordingly. But this mindset is trivial. When deconstructed, we really only care about the derivative of change from the current value. When investing it doesn't actually matter what the true value of an asset is as long as the investor can accurately model whether it is under or overvalued. Cryptocurrencies are valued based almost entirely upon public perception of value. Thus the factors driving price changes are going to be psychological in nature. Therefore, the algorithms to model cryptocurrencies should be quantitative algorithms to model human behavior.

This mindset towards investing is nearly orthogonal to much of what is taught in University Econ classes. So the natural question arises. What are the models that dictate cryptocurrency price movement?

3 Technical Intro

3.1 Problem

Today cryptocurrencies are a highly volatile asset class. If an investor is interested in entering these markets they effectively need to write off the cryto chunk of their portfolio and only view the enter & exit price on their investment. They cannot analyze the Sharpe on their investment because the σ is much too high. This reality is ludicrous. As of writing crytocurrencies have a combined market cap of $251 Billion, and the asset class holds a promising future. They have orthogonal returns to the S&P 500 and thus could significantly reduce aggregate portfolio risk if accessible as a usable asset. The fact that the majority of institutional investors can't touch crypto implies that massive market inefficiencies exist.

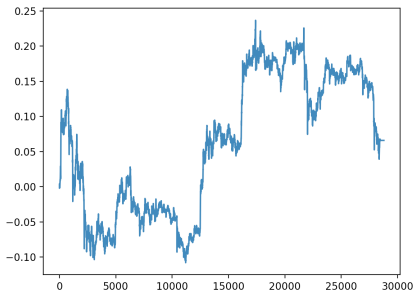
Furthermore, the traditional methodologies for asset stabilization do not work on cryptocurrencies due to the nuanced nature of the asset class. The most notable flawed yet popular idea to address is the idea of a crypto ETF, or an S&P 500 for crypto. This idea is flawed for two reasons. The first is that the beta among currencies is far to high to stabilize the asset class by bundling. The second reason this approach is poor is because if measured by market cap, the ETF would inherently contain "shitcoins." These are coins savvy investors should not hold, yet still dominate the charts.

Any viable solution must possess the capabilities to stabilize a singular asset, as opposed to simply leveraging the ramifications of the efficient market hypothesis (EMH) as applied to cryptocurrencies.

3.2 Solution

To stabilize these assets I built AVA. AVA's AI is designed to continuously monitor market signals & provide insights as to the direction of the market n minutes into the future. When the AI is very confident of an upcoming market move or a region of high volatility, it can move money into a stable base currency, long, or short the currency accordingly. The AI is trained on data from numerous currencies and thus applies translational learnings between them. This means that AVA can intelligently predict how a Bitcoin move three days ago will effect how Litecoin is going to move in the next 5 minutes.

In short, we turned the asset on the left, into the asset on the right.



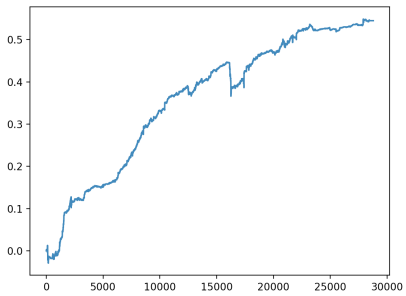


Figure 1: Litecoin chart. x axis = minutes from t_0, where t_0 is sometime on May 16th. y axis = total change as a fraction of original price

Figure 2: An equivalent investment with AVA. Axis the same.

AVA has the capabilities to stabilize highly volatile assets.

3.3 Investible Universe

AVA has the capability to trade any coin with enough liquidity. Today, this includes BTC, ETH, LTC, BCH, XRP, ETC & ZEC if traded from coinbase pro. The system has the capabilities to make predictions across, 1 minute, 5 minutes, 15 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, 6 hours, 12 hours, and 24 hours. Thus, when I refer to n in this paper, what I'm really saying is $\forall n \in \{1, 5, 15, 30, 60, 120, 240, 360, 720, 1440\}$. Much of this paper focuses on models & algorithms related to trading LTC with predictions for 5 minutes in the future. I chose this combination because there is relatively high inefficiencies at this time scale, and LTC is a rather unstable coin. Thus, this combination can highlight the benefits of the AVA system quite well. That being said, LTC & 5 minutes are in no way unique to AVA. It can perform similarly well on ETH @ 2 hours.

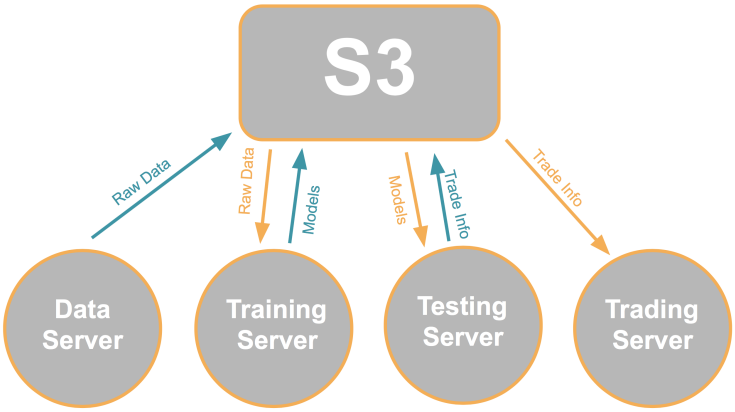
3.4 Market Analysis

3.5 Architecture

There are two key components of the AVA architecture. The first is the high level relationship between computational & informational servers. The second is the internal architecture of each computational server.

3.5.1 Relational Server Architecture

AVA is discretized into five servers, four computation servers & one information server. Each server has a highly specific purpose, and the computation servers have their own inputs and outputs (similar to a function). They also do not directly interact with each other - again behaving as clean independent functions as opposed to one glob of code. This design allows for modularity between components, and reliability in case one component breaks. The architecture can be visualized as follows:



The server flow starts with the data server. The data server pulls in live market data every three hours and preprocesses it as necessary. Then, it pushes all of that data to the S3. The training server starts by pulling down the necessary data from S3. It then takes this data and trains predictive models on the data. The training server then pushes just the best models to the server. The testing server downloads those models and analyzes them. After producing optimal investment criteria per a specific model, it uploads that information to S3. Finally, the trading server downloads the optimal investment information for the best model. It then trades on the market.

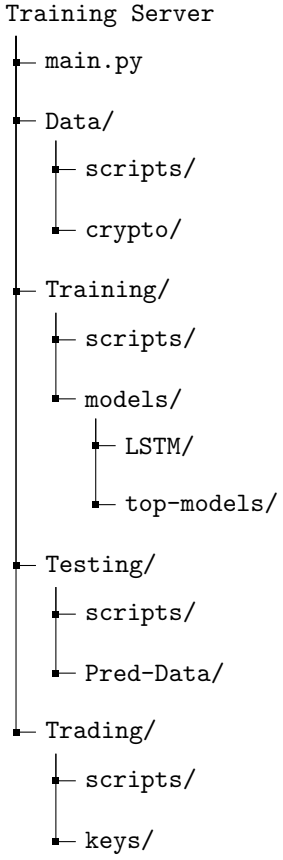
This paper's structure is inspired by the server architecture. From here forward, the paper linearly discusses four discrete sections on the data, training, testing, & trading.

3.5.2 Internal Server Architecture

Each server shares a relatively homogeneous internal architecture. Additionally, all files & functions are thoroughly commented. The combination of these design decisions makes debugging easy and on-boarding of new engineers into the system relatively efficient.

A main file controls all key functions inside of each server. This choice allows the engineer to run different operations which may occur different parts of the server successively without the need to be moving around the server continuously. Each major folder includes a scripts subsection for all code.

Below is an abbreviated version for demonstration purposes of the training server architecture, which can serve as a representative model for the other servers.



4 Data

4.1 Overview

The AVA system relies on per minute price data from BTC, ETH, LTC, & BCH. It preprocesses this raw data before use in the RNN.

4.1.1 Raw Data

The data server reads in raw data from the "cryptocompy" python API wrapper. Only the most recent day of per minute data is available from an API call, and the internal data pipeline must stay up-to-date. Therefore, the data server calls this API & stores the results every three hours. A separate error server monitors that this server is properly online. If the server goes down I get a text.

Getting the most recent price data is as simple as:

```
price.get_historical_data(currency, 'USD', 'minute', aggregate=1, limit=(500))
```

4.1.2 Pre-processing

After the raw data has been recorded, the AVA system produces price change and volatility derivative metrics to use as inputs into the RNN. The function to generate an input, X_{ij} solely depends on the price array. It can be calculated as follows:

$$X_{ij}(\boldsymbol{p}) = \Big[\Big(\begin{bmatrix} p_i \\ p_{i-j} \\ p_{i-2j} \\ ... \\ p_0 \end{bmatrix} \cdot \big[(\tfrac{11}{13})^0 \quad (\tfrac{11}{13})^1 \quad (\tfrac{11}{13})^2 \quad ... \quad (\tfrac{11}{13})^i \big] \cdot \frac{2}{13} \Big)^{-1} \cdot p_i - \overline{\mu}(X_{ij}) \Big] \cdot \frac{1}{\overline{\sigma}(X_{ij})}$$

To put simply, an individual exponentially moving average (ema) derivative data-point for time i is generated by taking the price at time i and dividing it by the ema, and then normalizing by the expected value and standard deviation of that random variable. The final two steps exist in order to have all distributions centered at zero with std. of 1. This calculation must be performed repeatedly to generate each individual i, j combination of data. As of writing, there are 33.6 million of these combinations.

Volatility metrics are calculated as a derivative on the ema metrics. It requires four inputs - the per minute high, low, price, and Xi value. The array of volatility data is calculated as follows:

$$\boldsymbol{V(X_j, h, l, p)} = \sqrt{\frac{|\boldsymbol{X_j - l}| \cdot |\boldsymbol{X_j - h}|}{\boldsymbol{p^2}}}$$

4.1.3 Data Quantity

The most recent sizes for the data matrices are as follows:
Raw: 692397 x 3 = 2,077,191 net pieces of raw data
RNN Input Data: 692397 x 88 = 60,930,936 net pieces of input data

4.2 Additional Notes

Multiple different data sources and attempts were made before deciding on this specific implementation. In the code base there is also the full infrastructure necessary for scrapping crypto relevant news from 37 major news sources. The code to sentiment these news articles in order to use the articles as RNN inputs is also implemented. However, due to live trading complications on news data this element of the code base is not utilized currently.

5 Training

5.1 Overview

AVA uses an RNN-LSTM to make predictions as to where the market is going to move n minutes in the future. RNN-LSTM's are notoriously good at modeling time series data, such as asset price data, due to their ability to discount old data while factoring in previous market moves. AVA tests on the most recent fifteen days of data, and trains on everything else.

5.2 Model

5.2.1 RNN-LSTM's Unique Effectiveness on Crypto Data

If a series of random variables (such as 5 minute crypto returns) does not limit to a normal distribution, that implies that the series of random variables is not independent and identically distributed, but instead the variables are somehow dependent upon each other. This dependence is especially true for crypto given much higher rates of volatility clustering than in traditional stock markets. See the data analysis section for more on this effect.

An RNN-LSTM model nulls the Markov assumption which many NN make. That is it does not assume $P(X_i = x|X_{i-1}, X_{i-2}, ..., X_0) = P(X_i = x|X_{i-1})$. This has the effect that RNN-LSTM's work very well on data where $P(X_{100} = x)$ might depend on X_{89}, X_{74}, X_2. It works very well on highly dependent series of random variables. Which, as we showed in the data section, perfectly describes the behavior of cryptocurrency returns as a RV.

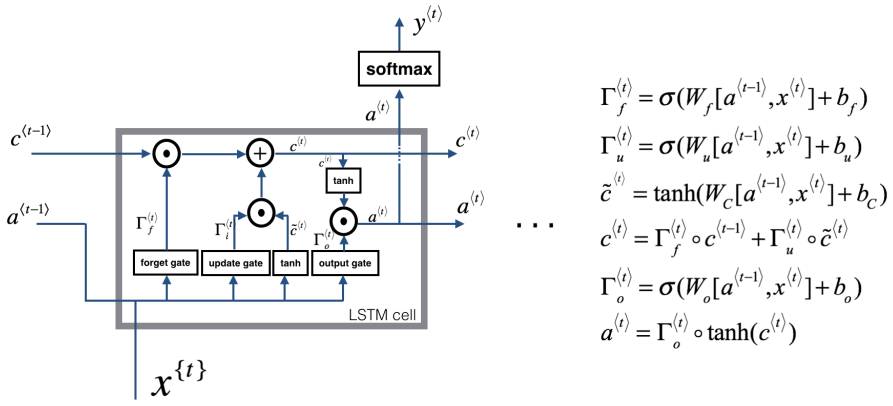
5.2.2 Underlying Math

AVA uses a traditional RNN-LSTM model. Many of novelties of AVA are not in the uniqueness of the model design itself, but rather in the data used, the tuning measures employed, and the comprehensive analysis afterwards.

What is unique about RNN's in comparison to other NN's is their ability to factor previous data into the prediction. It achieves this through the use of an additional weights matrices - illustrated below:

$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$
$$\hat{y}_t = Softmax(W^S h_t)$$

The LSTM layer discounts old data in order to weight newer market data higher. An illustration of this process is as follows:



$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$
$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$
$$\tilde{c}^{\langle t \rangle} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$
$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$
$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$
$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

5.3 Optimization Methodology

To optimize the hyperparameters of the RNN we employed the technologies of SigOpt. Doing so illuminated the sensitivity of the model to hyperparameter tuning. Without enough time to train a certain model, or with an incorrect learning rate, the model couldn't beat random guessing. However, with properly tuned parameters it could predict quite well. There were three naturally occuring phases to optimizing with SigOpt.

5.3.1 Phase 1

During first training round we did not give the models enough time to tune. This resulted in generally convoluted and often confusing results. However, the parameter importance data corroborated our expectations thus indicating the system was functioning correctly.

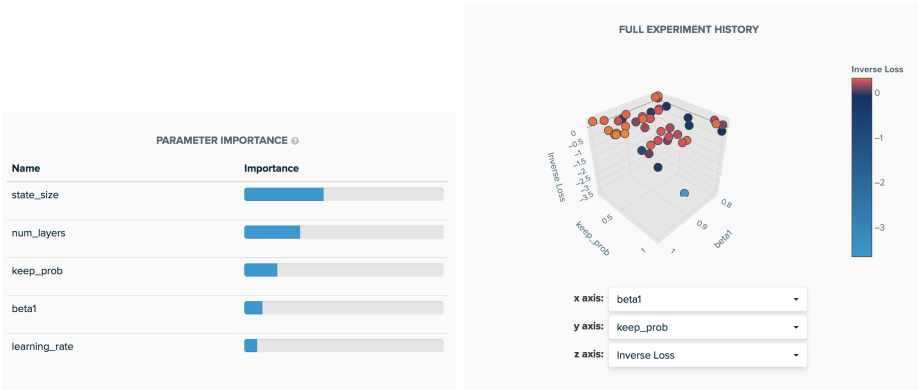


Figure 3: The relative importance of each hyperparameter on the left, the the distribution of results on the right

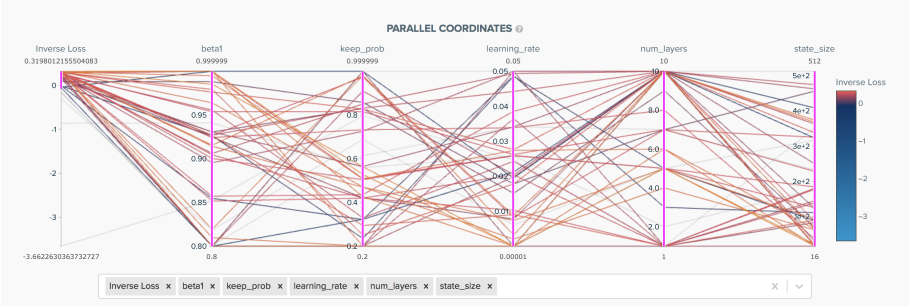
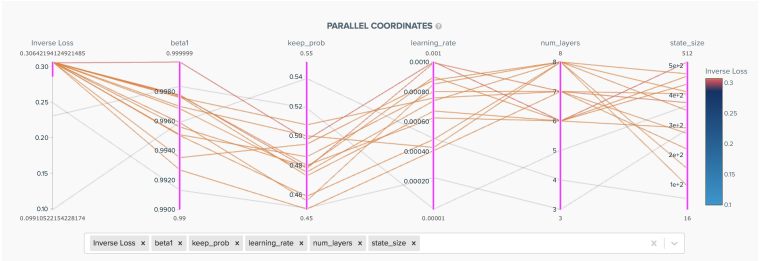


Figure 4: Results remained unclear

5.3.2 Phase 2

After giving the models enough time to train we were able to discern clear patterns in how to best optimize some of our hyperparameters. However, we still didn't know how to best optimize the number of layers or state size.



5.3.3 Phase 3

On our final batch of training, where we gave each model considerably more time to tune, we were able to identify very clear patterns in how to best optimize the models.

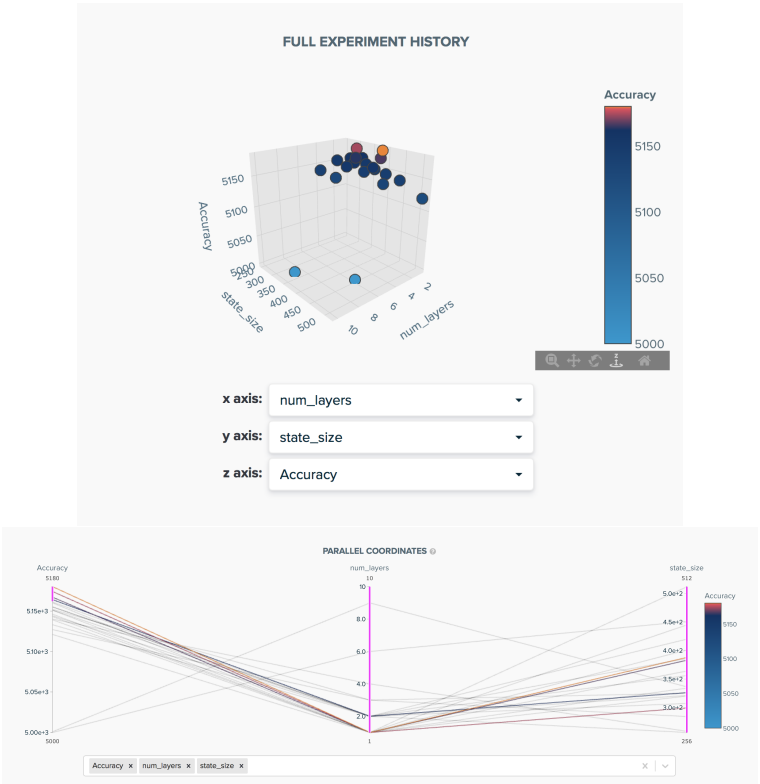


Figure 5: Superior Models in Orange

5.4 Results

This paper outlines results from two combinations of currency and investible periods, BTC 120 and LTC 5.

5.4.1 BTC 120 Minute

State Size	Num Layers	Test Accuracy	Test Alpha Accuracy
125	5	56.79%	49.78%
286	3	52.27%	50.51%
366	5	52.91%	50.83%
52	4	54.15%	51.45%
83	1	57.87%	50.32%

5.4.2 LTC 5 Minute

Note that the model represented on the fourth row had the same hyperparameters as the model on the fifth row, the only difference is that the fifth row model had significantly more time to train.

State Size	Num Layers	Test Accuracy	Test Alpha Accuracy
256	2	53.32%	51.51%
299	1	53.79%	51.74%
321	2	53.41%	51.55%
388	1	53.90%	51.80%
388	1	62.77%	54.59%

6 Testing

6.1 Approach

During the testing phase the goal is to understand how to best use the trained model to make predictions & optimize the Sharpe on trading. To do so I built multiple functions to optimize the buy/sell prediction point, and others to identify the highest accuracy prediction points. However, these numbers alone paint a clouded picture of what's really going on. To understand the testing phase it is first critical to understand the data we're analyzing, and how to use that data to inform intelligent predictions.

The output data which is analyzed in testing is an array of length 28800x3. The first column is how the market behaved over a 5 minute interval, and the second two columns are how we predicted the market would behave. Our predictions are in the range (0,1), which represents AVA's confidence that the market is going to go up or down. A sample subset of data is below:

Market % change	Negative Prediction	Positive Prediction (1 - negative)
0.0967305088024708	0.44528747	0.55471253
0.22228665313617857	0.40866876	0.5913313
0.13534416086620318	0.52520645	0.4747936
0.2896591677126554	0.3851425	0.6148575
0.1641718976339949	0.44165558	0.55834436

To effectively trade we need to know which subsets of our predictions are most valuable, and by how much. To discover this subset we're going to analyze recall over different confidence intervals.

6.2 Market Subsets

In this section we analyze subsets of our total prediction set. Subsets are determined by the prediction confidence.

6.2.1 Full Sample Space Ω

Over the complete sample space:
Positive Recall: 62.12%
Negative Recall: 63.32%

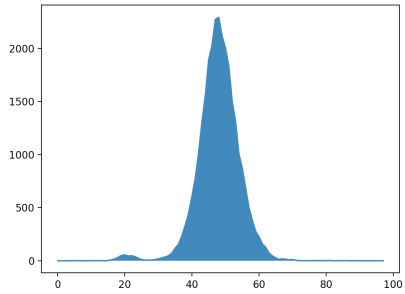


Figure 6: PDF of All Predictions (predictions scaled up by a factor of 100)

6.2.2 $\{X_i | X_i \in \Omega \wedge (X_i < 0.45 \vee X_i > 0.55)\}$

Only analyzing predictions below 0.45 or above 0.55:
Positive Recall: 74.81%
Negative Recall: 70.01%

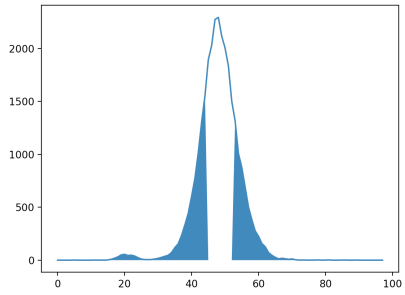


Figure 7: Subset of predictions at 0.55 or above in confidence

6.2.3 $\{X_i | X_i \in \Omega \wedge (X_i < 0.40 \vee X_i > 0.60)\}$

Only analyzing predictions below 0.40 or above 0.60:
Positive Recall: 80.81%

Negative Recall: 78.23%

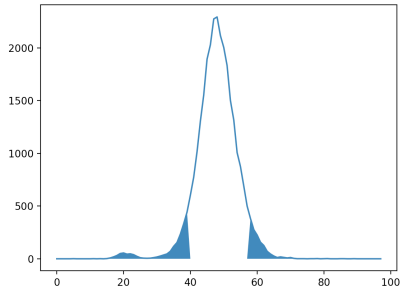


Figure 8: Subset of predictions at 0.60 or above in confidence

6.2.4 $\{X_i | X_i \in \Omega \wedge (X_i < 0.35 \vee X_i > 0.65)\}$

Only analyzing predictions below 0.35 or above 0.65:
Positive Recall: 78.14%
Negative Recall: 89.66%

Notice the positive recall dropped a little. I theorize this drop can be attributed to a smaller sample size and thus more variance in performance.

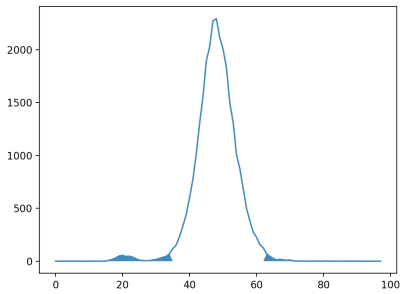


Figure 9: Subset of predictions at 0.65 or above in confidence

6.2.5 Conclusion

We're going to analyze the model performance on two sets, Ω and all predictions of greater than 0.6 confidence. A choice of 0.6 allows for enough of a sample to get statistically significant returns, while maintaining discretion on which

trades get executed. Analyzing results at 0.5 allows us to compare the effects of limiting trade to a subset of predictions vs. always trading.

6.3 Fee Based Analysis

In this section we analyze how the model would have performed in the market during the extended test set (because data updating is live we could analyze more than just the test set of unforseen data). We specifically analyze the performance given different fee structures.

6.3.1 Current Fee Structure - 25 bips loss/trade

If AVA traded today without any intelligent market entry algorithms it would face the common exchange fee of 25 bips per trade.

Returns & Sharpe with a fee of 2.5¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	-488.42%	-58.99%
Sharpe:	-19.9	-24.28

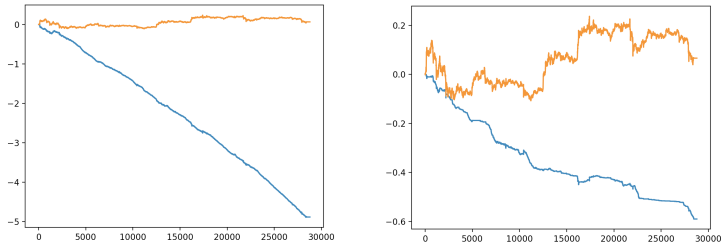


Figure 10: 0.5 Left, 0.6 Right

6.3.2 $10k Traded - 10 bips loss/trade

If AVA ran with $10k, again without any intelligent market entry algorithms, it would face an exchange fee of 10 bips per trade.

Returns & Sharpe with a fee of 1.0¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	-111.82%	-4.50%
Sharpe:	-5.05	-2.04

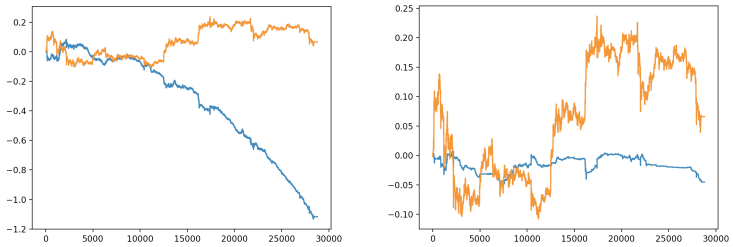


Figure 11: 0.5 Left, 0.6 Right

6.3.3 $1MM Traded - 5 bips loss/trade

If AVA ran with $1M, again without any intelligent market entry algorithms, it would face an exchange fee of 05 bips per trade.

Returns & Sharpe with a fee of 0.5¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	13.89%	13.65%
Sharpe:	0.639	6.27

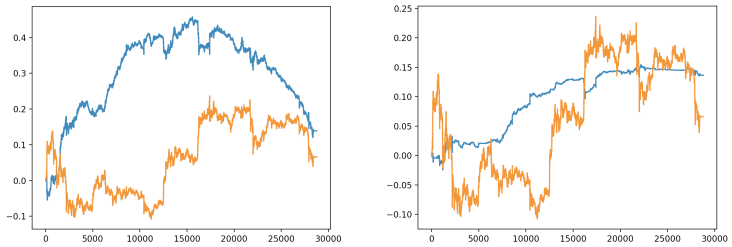


Figure 12: 0.5 Left, 0.6 Right

6.3.4 Highly Efficient Entry - 4 bips loss/trade

If AVA ran with intelligent market entry algorithms today it would face an exchange fee of < 4 bips per trade. 4 bips per trade is the lowest cost achievable by exchangify, who I've partnered with to route all my trades through.

Returns & Sharpe with a fee of 0.4¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	39.01%	17.27%
Sharpe:	1.79	7.95

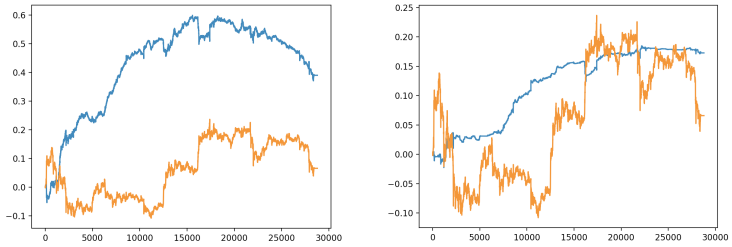


Figure 13: 0.5 Left, 0.6 Right

6.3.5 Theoretical - 2 bips loss/trade

If we could theoretically achieve a loss of 2 bips per trade performance skyrockets. Please keep in mind these are theoretical Sharpes.

Returns & Sharpe with a fee of 0.2¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	89.24%	24.54%
Sharpe:	4.12	11.31

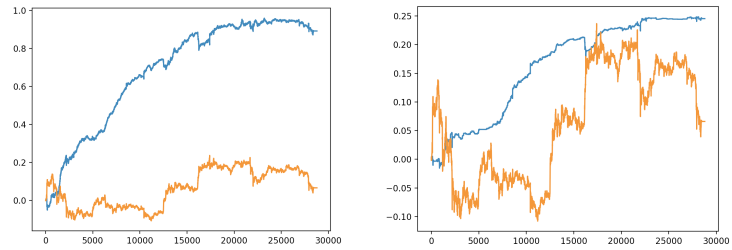


Figure 14: 0.5 Left, 0.6 Right

6.3.6 Theoretical Performance Limit - 0 bips loss/trade

If we could theoretically achieve a loss of 0 bips per trade we can fully separate the algorithm's prediction abilities from its market entry effectiveness. These are the theoretical limits of the Sharpe in a perfect investment state.

Returns & Sharpe with a fee of 0.0¢ per trade on every $10 invested:

	0.5 Thresh	**0.6 Thresh**
Returns:	139.48%	31.80%
Sharpe:	6.44	14.64

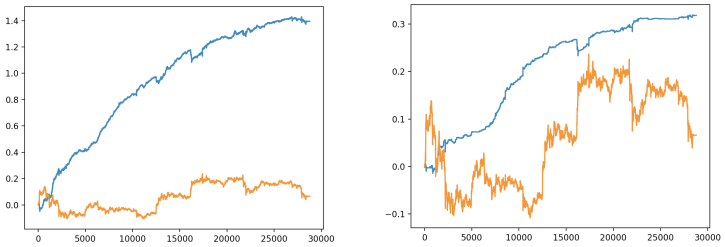


Figure 15: 0.5 Left, 0.6 Right

6.4 Conclusions

The insights gleaned from an analysis of this model & its performance vs. the market holds many conclusions. I discuss three of them here.

First, by intelligently investing on only a subset of potential trades we were able to more than double the Sharpe. This choice demonstrates the effectiveness of the AVA system at reducing volatility.

Second, the performance limit on this strategy is very high. When just analyzing the prediction technology there is lots of potential.

Third, even if you can predict the future that doesn't mean you can profit off those predictions. Section 6.3 illustrates the massive effect even slight changes in fees make to a quant strategy which trades frequently. Notice a one bip fee drop from 5 to 4 doubles the returns on the 0.5 strategy.

7 Trading

7.1 Implementation Approaches

The output of AVA so far is an insightful prediction between (0,1). Granted we know how to predict market quite accurately on a subset of these predictions, however, they still remain predictions. Binary buy & sell at a certain threshold is not necessarily the best way to convert these predictions into alpha. The natural next question is "how do you most effectively trade on these predictions?" In this section I overview the live trading algorithm and offer a few trading strategies I especially like.

7.2 Technology Behind Live Trading

It currently takes over two hours to load in all new data, pre-process it, feed it into the RNN, and output a result. If you're making predictions about 5 minutes into the future, the result you get is useless. That five minute window passed, 1 hour and 55 minutes ago. Thus, to construct a live trader, I had to find a way to cut a 2 hour load time into less than a minute.

I achieved this reduction in latency by simplifying the data processing pipeline and constructing a more effecient way to process and return information.

7.3 Implementation Approaches

There are theoretically an infinite number of trading strategies which can be superimposed on top of AVA's prediction set. I outline three here.

7.3.1 Binary Buy Sell

Binary Buy Sell is perhaps the simplest of all prediction based trading strategies. This strategy is the one used in section 6.3. If the prediction value is above or below a certain threshold, invest or short accordingly. If not, hold in a stable currency. The returns from this strategy are solid. It can only go up from here.

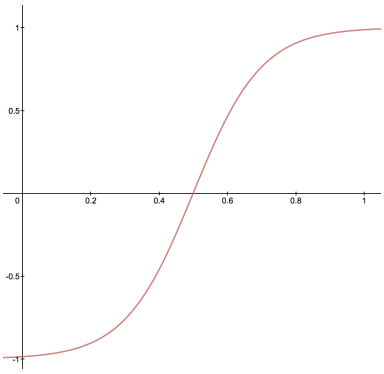
$$f(x) = \begin{cases} -100\% & x < 0.4 \\ 0 & 0.4 \leq x \leq 0.6 \\ 100\% & 0.6 < x \end{cases}$$

7.3.2 Dynamic Asset Scaling

Dynamic Asset Scaling is a slightly more sophisticated version of Binary Buy Sell. Instead of buying or selling as a discrete function of price it scales the total assets held in crypto to factor in the unusual distribution of predictions.

$$f(x) = \frac{2 \cdot e^{5(x-0.5)}}{e^{5(x-0.5)} + e^{-5(x-0.5)}} - 1$$

This function graphed appears as follows, where the x axis is the prediction output, and the y axis the percent of the portfolio held in crypto (negative is short):



With shorting the strategy performs relatively well, as a near intermediary between the 0.5 and 0.6 Binary Buy Sell strategy. The Sharpe's are 7.23 with no fees and 3.92 with 4 bips/trade fees. The first picture is the strategy without the ability the short, which yields a Sharpe of 0.605.

Figure 16: Left: No fees, Right: 4 bips/trade fee

7.3.3 Stop Order Optimization Strategy

Stop Order Optimization Strategy is a strategy where you automatically sell when the asset hits a pre-optimized price, and if it doesn't hit that price in m

time you auto sell at market.

To identify how to sell, we first create what I call "market objects." A single market object is the market at one minute in time, and the opportunity of buying at that price. An example market object is as follows:

```
12/01/18 03:45:00: {
    price: 5000,  # current price
    max: 1.0052,  # Relative maximum in the next n time
    min: 0.98442, # Relative minimum in the next n time
    close: .9953, # Close in n time relative to current price
    pred: 0.36    # Model prediction for future movement, between [0,1]
    }
```

We want to optimize the expected return for selling at a certain price, λ. This is equivalent to optimizing the percent of times we successfully sell at λ, multiplied by λ, minus, the expected price we sell at when we don't hit λ, multiplied by the percent of times we don't hit λ. Mathematically this is as follows:

Returns: $f(\lambda) = len(\chi)(\lambda) - len(\chi^c)(E[E[\text{"}close\text{"}]])$

First define variables to help understand the math.
Let A be the set of all market objects.

Let $\gamma \subset A s.t.$ for each gamma, the model buy conditions hold. Gamma is market objects where we invest.

Let $\chi \subset \gamma s.t.$ for a given sell price, λ, the order is successfully executed.

Let $\chi^c \subset \gamma s.t.$ for a given sell price, λ, the order is not successfully executed.

$len(\chi^c) + len(\chi) = len(\gamma)$

Goal: Find λ such that $g'(\lambda) = 0$ given $g(\lambda) = \frac{len(\chi)(\lambda)}{std(len(\chi))} - \frac{len(\chi^c)(E[E[\text{"}close\text{"}]])}{std(XY)}$

I haven't implemented this algorithm on the most recent data yet, thus no results to show. I think the results could be interesting.

8 Applications

Today AVA is designed to actively predict price changes on crypto markets. However, I personally cannot profit on these predictions due to the high fees associated with low liquidity and no stable market entry algorithm. AVA could however deliver tremendous value to a quant shop which already has these assets.

Any crypto shops which do exchange arbitrage, or any form of high frequency trading could make a lot of money off of AVA.

If you see AVA benefiting your organization, feel free to reach out at afine@stanford.edu. I'm excited to see where AVA goes next.