# Summary of QuantConnect

QuantConnect is an open source, community driven algorithmic trading platform.

QuantConnect supports algorithmic trading by providing an enormous library of data of roughly 40TB in size for backtesting. This library includes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Asset Class | Source | Start Date | Symbols | Resolution | Type |
| US Equity | QuantQuote | Jan 1998 | ≈30,000 | Tick, Sec, Min, Hour, Daily | Trades |
| Forex | FXCM | Apr 2007 | 13 | Tick, Sec, Min, Hour, Daily | Quotes |
| Forex | OANDA | Apr 2004 | 71 | Tick, Sec, Min, Hour, Daily | Quotes |
| CFD | OANDA | Apr 2004 | 50 | Tick, Sec, Min, Hour, Daily | Quotes |
| Options | AlgoSeek | Jan 2010 | ≈4000 | Minute Bars Only | Trades & Quotes |
| Futures | AlgoSeek | Jan 2009 | ≈1100 | Tick, Second, Minute | Trades & Quotes |
| Crypto | Kaiko | Jan 2015 | 12 | Tick, Second, Minute | Trades |

**What is algorithmic trading and how does QuantConnect support it?**

Algorithmic trading is a process where trades are made by the computer instead of a human. The computer is programmed to take certain actions in response to certain varying market data.

With QuantConnect, users can design and test their strategies on free data before deploying it live. I think this is one of the greatest ways that QuantConnect supports algorithmic trading. By providing data for the testing of algorithms, users can have a risk free way to check if their algorithms work as well as they expect, encouraging users to design and experiment with algorithms and models at no cost.

Furthermore, backtesting on QuantConnect is executed in the cloud, where users are not required to have powerful computer hardware to run their algorithms over such huge amounts of data.

# Summary of Machine Learning and Trading Paper

The paper I have selected is titled Application of Deep Learning to Algorithmic Trading [1].

In the paper, a Long Short-Term Memory (LSTM) network, a time series version of Deep Neural Networks, was implemented to forecast the stock price of Intel Corporation (NASDAQ: INTC). For the model, three input features were use:

1. the historical trading data of INTC (OHLC variables)
2. commonly used technical indicators derived from OHLC variables
3. the index of the ﬁnancial market and the semiconductor sector

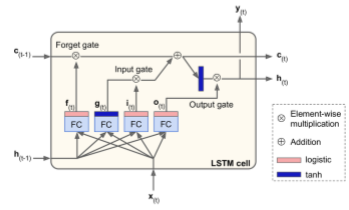
All of these features reﬂect daily values of these variables, and the network predicts the next day’s adjusted closing price of INTC based on information available up to the current day.



The original dataset is split into three equal periods, and each period is further divided into the train and test data sets as illustrated in Figure 1. This minimizes bias in both training and testing the model.

## Long Short-Term Memory (LSTM) network

The LSTM is a variant of the Recurrent Neural Network (RNN) and is good for use with sequential data. Like RNN, LSTM has a recurrent structure where each cell not only outputs prediction ˆ yt but also transfers activation ht to the next cell. The striking feature of LSTM is its ability to store, forget, and read information from the long-term state of the underlying dynamics, and these tasks are achieved through three types of gates. In the forget gate, a cell receives long-term state ct−1, retains some pieces of the information by amount ft, and then adds new memories that the input gate selected. The input gate determines what parts of the transformed input gt need to be added to the long-term state ct. This process updates long-term state ct, which is directly transmitted to the next cell. Finally, output gate transforms the updated long-term state ct through a tanh(·) function, ﬁlters it by ot, and produces the output ˆ yt, which is also sent to the next cell as the short-term state ht. The figure below illustrates this process.



Since this is a RNN, LSTM is trained via Backpropagation Through Time. The key idea is that for each cell, we ﬁrst unroll the ﬁxed number of previous cells and then apply forward feed and backpropagation to the unrolled cells. The number of unrolled cells is another hyperparameter that needs to be selected in addition to the number of neurons and layers.

As of now, I do not perfectly understand the math behind why and how this model works and shall thus leave it out of the paper summary.

## Trading Strategy

The trading strategy adopted by the paper is as such: the investor buys one share if the predicted price is higher than the current adjusted closing price. Otherwise, the investor sells one share of the stock.

## Conclusion

In conclusion, the paper demonstrated how the LSTM model, with their chosen set of hyperparameters, is able to predict the price of the stock at a higher accuracy than a regression model, especially when the stock price is lacking a trend.

# Algorithmic Trading Models

**For making money with Forex, what machine learning strategy can you apply? Is there more than one? If so, what are the pros/cons of each, select up to 3 models to write about.**

From the paper discussed above, we see that there are other machine learning strategies that can be applied to Forex, or trading in general.

Firstly, an Artiﬁcial Neural Network (ANN) with 1 hidden layer can be applied. However even if the ANN can forecast market trend, the paper concluded that a multiple linear regression and Support Vector Machine (SVM) outperforms the ANN. Extensions of the ANN such as dynamic artiﬁcial neural network and the hybrid neural networks which use generalized autoregressive conditional heteroscedasticity can be used as well, but these models suffer from low accuracies.

Next, the paper highlights deep learning models with multiple layers as a promising model suitable for predicting financial time series data. Citing other papers, it claims that a framework that integrates Wavelet transformation, Stacked Autoencoders, and LSTM outperforms the canonical RNN and LSTM in terms of predictive accuracy.

**Select one model for use on forex and describe how it works to select when to buy/sell and how it uses data?**

The preliminary model I used for forex trading is a Moving Average Convergence Divergence (MACD) model. This is a trend following momentum indicator that shows the relationship between two moving averages in prices. The MACD used in my model uses a fast period of 12 days, a slow period of 26 days and signal period of 9 days. The MACD is calculated by subtracting the 26 day exponential moving average (EMA) from the 12 day EMA. A 9 day EMA is then plotted on top of the MACD as a signal line to trigger trading.

We shall set a tolerance of 0.002 to avoid entering a position too early and experiencing a “faked out”.

Theoretically, if the MACD falls below the signal line, it indicates a bearish signal, which indicates that it is time to sell. On the other hand, if the MACD rises above the signal line, it indicates a bullish signal which suggests that the price is likely to experience an upward momentum and it is hence a good time to buy. The results from the backtest are as shown below.  
We made a net profit of $-4k with this model.



However, I realised that if we inversed this logic, by selling when MACD rises above the signal line and buying when MACD falls below the signal line, we are able to get a much better performing model which makes a net profit of $8k in the same time period of Jan 2016 to Sep 2018. The trading strategy is as following: if at an OnData event, we go long by selling. Otherwise, we go short by buying.



**What are three technical differences in model performance between a fiat model and cryptocurrency model?**

1. The fiat model experienced a much lower net profit of just $7.7k as compared to the cryptocurrency model at $95.8k.
2. The fiat model has a return of 7.44% compared to cryptocurrency model at -47.25%
3. The fiat model has a compounding annual return of 2.64% compared to cryptocurrency model at -20.7%

I believe this is due to bitcoin’s deflationary attribute backed by the currency’s 21 million capped supply, as opposed to forex currency whose supply is controlled by the central banks.

# Works Cited

|  |  |
| --- | --- |
| [1] | G. Chen, Y. Chen and T. Fushimi, "Application of Deep Learning to Algorithmic Trading," Stanford University, 2017. http://cs229.stanford.edu/proj2017/final-reports/5241098.pdf |