Project Report : CS 7643

Sakae Watanabe
Gatech
swatanabe30@gatech.edu

Ashish Narasimham
Gatech
anarasimham3@gatech.edu

Michael Walker
Gatech
mwalker311@gatech.edu

Abstract

We apply state-of-the-art Deep Learning approaches to the problem space of short-term trading, using daily market open/close/volume data to create time-series batches of sequences within the dataset. We take 3 different approaches: Spatio-Temporal Transformer (STT), Liquid Neural Network (LNN), and Identity Recurrent Neural Network (IRNN); and attempt to extract signal from high-volatility public company stock movements in the healthcare space. Our results indicate that further investigation is required to create nuanced models that balance buying, selling, and holding inferences to achieve lower training loss. However we have been able to create a profitable trading strategy even with sub-optimal trading accuracy.

1. Introduction

1.1. Background

Simply, we attempted to generate profit on stock market historical data using deep learning methods. To reduce the scope, we focused on a basket of 5 stocks in the healthcare space that are publicly traded; we chose to focus on smaller companies, conjecturing that these are less influenced by algorithmic trading than the larger companies that have more visibility and are traded with higher volumes of shares exchanging hands. There are many large firms that have financial interests in optimizing profit algorithmically, so academic sharing of best practices isn't the standard for the industry. There are a few papers published that survey modern techniques, but largely we sourced ideas from other industries and reapplied the tools to a new context while considering what might work well for time-series datasets. Modern approaches generally have limitations across several dimensions. 1) They are better at classification than they are at regression. 2) They can predict near-term movements better than longer time horizons. 3) A myriad of data sources is required to accurately model stock movements - everything from political events and weather anomalies to X (formerly Twitter) posts and key individuals' thoughts can provide strong signal for stock movements. 4) They need significant amounts of data to accurately distill patterns into weights. If successful, beyond the immediate profits generated on the public trading markets this exploration would inform time-series deep learning approaches more broadly across industries. Real-time human vitals could be analyzed for anomalous adverse health events; automobile telemetry could be analyzed for part failure; and factory workers could be kept safe from industrial machinery malfunctions.

1.2. Data

We gathered data from Yahoo! Finance's daily stock data APIs to obtain historical pricing data for 5 different symbols (e.g. https://finance.yahoo.com/quote/ATNF/history/ [**?**]). A simple ETL pipeline was used to transform the data into usable input features. Four of the stock tickers were selected based on high volatility (more opportunity to outsmart the market if the stock goes further up and down), participation in the Healthcare space (interesting area that could be the next frontier for innovation), and small market cap (hopefully not as many competing algorithmic market participants). The final symbol SPX was selected to capture a broader signal to promote learning. Raw and processed data are stored in the GitHub repository.

1.3. Dataset Properties

The ticker data consists of columns date, open, high, low, close, adjusted close, and volume over the range 2020-10-18 to 2024-10-18. This daily ticker data can be used to train medium-term models on the order of days; intraday predictions would require data at 1- or 5-minute intervals.

1.4. Training Splits

The data was partitioned into training, validation, and testing sets as shown in Table 1.

We labeled each row with one of three labels: buy, sell, or hold. The labels were a simple calculation based on the difference with the previous day: buy if it went up, sell if it went down, and hold if the delta is under a threshold.

	Train	Valid	Test
Start Date	2020-10-18	2023-10-18	2024-04-18
End Date	2023-10-17	2024-04-17	2024-10-18
Time Span	3 years	6 mo	6 mo

Table 1. Data splits and time frames.

1.5. Target Examples

Training examples were created by partitioning observations into windows of fixed-length τ, where each window contained features $X_t = x_{t-\tau}, ..., x_t$. The target label for the window was computed based on next day returns. ¡Some formal equation related to how target labels are created - can do an equation with cases here.¿

1.6. Features

A variety of technical indicators are commonly used in the enrichment of financial time-series data [**?**]. The following features were added to the training data:

- Simple Moving Average (SMA): 10, 20, and 30 day
- Exponential Moving Average (EMA): 10, 20, and 30 day
- Relative Volume: 10, 20, and 30 day
- %B: Computes the relative position of the current price within the 95% confidence interval bands of SMA20
- Moving Average Convergence Divergence (MACD): An oscillation indicator using the 9, 12, and 26 day EMAs
- Relative Strength Index (RSI):
- Momentum / Rate of Change: 10 day momentum. $Moment_t = price_t / price_{t-10} - 1$

Human traders that engage with markets on this timescale often use technical indicators such as these to make buy/sell decisions on their portfolio. Future areas of exploration would include combining these short-term signals with longer-term signals like metrics from the company's balance sheet, sentiment analysis from earnings press releases, and social signals from the broader market participant pool.

2. Approach

We took existing approaches in other industries and reapplied them to this new-to-us domain, taking a no-frills approach to data engineering, model training, and trading simulation in order to gain a deep (no pun intended) understanding and first-hand experience of the challenges in the space. We decided to structure the task as classification of next day movement - up, down or flat - rather than regression. Simply - we do not care by how much a stock will go up, we only care that it goes up so that we can buy it and subsequently sell when we predict it will reverse direction.

We applied the Liquid Neural Network (LNN), Spatio-Temporal Transformer (STT), and Identity RNN (IRNN) to the 5-stock dataset; we go into the model descriptions below.

1. The Identity RNN (IRNN) is the simplest of our three models, leveraging the basic Recurrent Neural Network architecture. The name is such because this variant simplifies the initialization to an identity matrix (ones on the diagonal, zeros elsewhere) for the weights and 0's for the biases to encourage the RNN to behave more like an LSTM. This works because when initializing to the identity matrix, the architecture tends toward an LSTM with the forget gate set to not decay the cell state at all. This is not equivalent to an LSTM but can stand in for one with comparative performance and a lot less parameters (4,000 learned parameters as tested), making for faster training cycles and lower inference latency.

2. The Spatiotemporal Transformer (STT) leverages time2vec temporal encodings, a transformer-encoder stack, an LSTM decoder, and a fully connected classification module - 3.28M parameters as tested. The positional encoding utilized in the original transformer architecture only considers relative position within a context window. Temporal encodings extend upon this concept, allowing encodings to have relative importance across different context windows.

   STT combines temporal encoding with expanded multivariate time series data within the sequence. Financial time-series data consist of observations $x_t \in \mathbb{R}^d$, where x_t represents the feature vector at time t. Rather than project x_t to an arbitrary dimension, STT concatenates the temporal encoding $t2v \in \mathbb{R}^k$ with x_t^i, the i-th feature at time t to produce intermediate embeddings. The expansion of the features combined with the concatenation of $t2v$ allows for more complex interactions across sequences and variables.

   The intermediate embedding $E_t \in \mathbb{R}^{d \times (k+1)}$ is then passed through a dense embedding layer that projects to $\mathbb{R}^{\gtrdot \times \lessdot}$ that the transformer-encoder accepts. Pre-trained t2v Embedding The t2v module of STT was pre-trained using a simple task of having a date at time t predict the date at $t + 1$ for the date range [1900-01-01, 2100-01-01]. This pre-training step provided additional stability during the primary model training and evaluation process. A more complex pre-training

task may provide increased benefit, but was outside the scope of this experiment. Note: will help pass over once we get this in tex.

3. Liquid Time-Constant Network LTCN is a take on the Recurrent Neural Networks that is based on and inspired by biological neurons, specifically C. Elegens. It uses a process to solve a series of Ordinary Differential Equations (ODEs) to solve problems in a way that allows the neural network to act in a spiking fashion, as well as being transparent in how it is performing its actions. LTCNs tend to require fewer resources than their deeper cousins - parameter count was 7,107 as tested - and work well with time series data, though they suffer from some of the same issues other RNNs suffer from, specifically vanishing gradients, especially with longer sequence data. Some solutions to these issues have been made, specifically Closed-form Continuous-time Neural nets, which seeks to reduce the number of Differential Equations down to 1 in an effort to better learn spatiotemporal tasks, and to better learn from sequential data, we can always work with Liquid Structural State-Space Models (Liquid S4).

2.1. Evaluation Metrics

Evaluation of classification performance focuses on True Positive (TP), True Negative (TN), and their false counterparts False Positive (FP) and False Negative (FN). The following classification based metrics were used to evaluate model performance: Accuracy: The percentage of correctly predicted labels $Acc = \frac{TP+TN}{TP+TN+FP+FN}$

3. Experiments and Results

Here are the results we were able to achieve with best hyperparameter configuration per model class, with the additional constraint that we are using the same set of hyperparameters for all ticker symbols to promote generalization across market data. This was set up as a 3-class classification task - sell, hold, or buy.

¡Table: rows are Models, columns are tickers as target_symbol, values are profit from simulation¿

Although we were initially disappointed by the 50% test accuracy across all three models, they generally ended with gains across the 6-month test window when put into simulation. This indicates that although they were noisy in their inferences, the training data enabled them to identify the key inflection points of the chart as they were happening. Some models learned to be very selective when buying and liberal when selling, in effect identifying entry points with very high confidence of a positive movement and then selling after losing some of that confidence. ¡should we more formally lay this out in a table? Class distribution per model on test data¿

3.1. Shared Hyperparameters

We found that each stock will be predicted optimally by a specific set of hyperparameters; it may not be the case that a single set of hyperparameters should work well across the board, given the nuances of each symbol's movements. However that single configuration can give a great place to start from when incorporating a new symbol into one's trading portfolio. One other note - the transformer leverages about 1000x the parameter count as the other two models do. This begs the question, is it worth it? The results paint a mixed picture, which is not a clear signal towards the Transformer-based model. Under the construct of using a single hyperparameter set, we don't have an answer. However future research could investigate individually tuning per symbol, perhaps allowing a best algorithm to emerge there. The experiments below are ablation studies testing two hypotheses we had; we focused on the $BIVI symbol and stock ticker for these experiments.

3.2. Optimizers and Loss Functions

The first ablation study we were interested in was testing different loss functions due to the distribution of the source data. Since the label distribution is imbalanced ¡insert label distribution per ticker summary¿, class-balanced focal loss could help to appropriately weight the dataset. We contrasted this with vanilla Cross Entropy Loss.

Both the RNN and Transformer perform better with the standard cross-entropy loss, but the LNN performs best with class-balanced focal loss. For the tested symbol, the Transformer had the best overall strategy test outcome.

3.3. Adding Additional Data with Pretraining

Another idea we validated was that signal from a broader selection of market data will benefit learning for the target dataset and ticker. We tested including other tickers from our dataset in a pre-training phase. We included all tickers except the target symbol in the pre-training phase, running the ablation study by varying the percentage of epochs spent training on the all-but-target-symbol dataset versus fine-tuning on the target symbol. For fine-tuning, we introduced another parameter to step down the learning rate and held that constant per model across this study, using the best value found per model. Results are presented below.

4. Conclusion

The

Looking forward, we'd look at testing these models further to validate their generalizability across industries, instruments, etc. - commodities, futures, and cryptocurrency are all interesting areas with their own opportunities and challenges. In terms of engineering work, additional intelligence can be added both when creating the labels for the

dataset and when running the simulation to improve profitability of the strategy.

5. Work Division

Student Name	Contributed Aspects	Details
Sakae Watanabe	Implementation, Experimentation	Data & feature engineering; training base code; STT model build/tune; trading simulation
Ashish Narasimham	Implementation, Paper Writing	Dataset sourcing; training code generalization; pre-training feature; RNN model build/tune; paper writing
Michael Walker	Implementation, Experimentation	Refactor code and configuration for reusability; experiment setup; LNN model build/tune

Table 2. Contributions of team members.