CS230

Deep Learning in Predicting Ethereum Volatility

Xiaotian Lu, Lai Xu (Shirley), Riley Yang
timxlu@stanford.edu, thyme815@stanford.edu, yangmy@stanford.edu

1 Introduction

As the popularity of cryptocurrency is growing in recent years, much research is actively conducted in bitcoin price predictions. Crypto prices are very volatile and become a barrier for many traditional investors. Our project aims to build a deep learning model forecasting the volatility of Ethereum. In the project, we plan to investigate the influence of DeFi protocols, which prior studies do not involve. The results of our project can be a profitable application that helps investors make more informed investment decisions.

2 Background and Literature Review

Decentralized financial protocols built on blockchain may look similar to exchanges and banks in traditional finance at first glance. However there are a few important differences. First, transaction level data executed on blockchain are accessible to public while most public studies in finance can only make use of daily aggregated information. Second, DeFi exchanges and lending protocols use innovative mechanisms to eliminate middle-man intervention. Prior research in TradFi may not be applicable. Ethereum is the most popular smart-contract blockchain attracting leading DeFi protocols. ETH is its native token involved in most DeFi apps. Predicting ETH price volatility using on-chain DeFi data on Ethereum is a task that deep learning can be applied to.
Few studies of Ethereum price volatility have been conducted. Han-Min[3] et al investigated the blockchain information's impact on Ethereum prices. Their work still looks into daily prices from traditional exchanges using high level blockchain statistics like gas fees. They attempted SVM model but did not study deep learning models. Mohammed[4] et al studied LSTM model in predicting cryptocurrency prices. They used aggregated data from centralized exchanges and did not focus on volatility. Valeria[5] studied deep learning models in forecasting cryptocurrency volatility. They only used daily prices and attempted to predict daily price range, which is not suitable for use in option trading.

3 Dataset and Features

In contrast to ETH prices derived from transactions and order books in centralized exchanges like Binance, we turn our attention to a new place - the decentralized financial protocols. Here we introduce the primary data source for ETH prices, Uniswap, which is a leading AMM protocol so far on the Ethereum mainnet. In this milestone report, we use two sets of dataset. One is the ETH prices derived from uniswap decentralized protocols and the other one is public data from yahoo finance as a cross validation.
In the scope of this project, we first look at the pair of ETH/USDT. USDT is a token backed by USD assets managed by a private company. Most of the time, USDT's valuation against USD fiat currency does not deviate more than 0.1%. When a trading pool is created, X units of ETH and Y units of USDT are sent to Uniswap and a constant C=X*Y is also registered. Normal traders exchange their USDT for ETH, or vice versa, by interacting with Uniswap directly. For each transaction, the constant

CS230: Deep Learning, Autumn 2022, Stanford University, CA.

is maintained so that C=(X+ΔX)*(Y+ΔY). The effective price of ETH in USDT is P=ΔY/ΔX. P's value depends on the size of the transaction, a phenomenon called slippage. When the current price in Uniswap deviates enough from the majority view from other trading venues, arbitrageurs step in to rebalance the pool so that Uniswap's price reflects the valuation of ETH.
For every transaction against the ETH/USDT pool, a Swap event is logged and recorded in the block's transaction receipt. We run an archival ETH node and make use of a blockchain data indexing framework called Substreams in order to map every Swap event to a record in a database table. In each block, there can be zero or multiple Swap events.The uniswap dataset contains in total 4435973 records and contains all the uniswap transactions from 2020-05-26 to 2022-10-31.

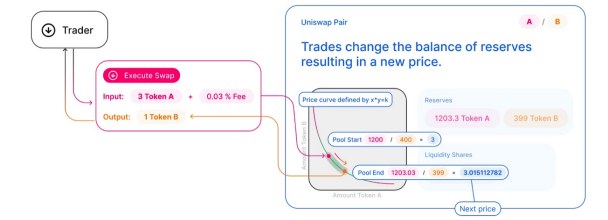

Figure 1: Demonstration of Uniswap

Data Field	Field Definition
block_number	blockchain sequence number
tx_timestamp	time when block is generated
log_index	index of swap event in a block
amount0_in	the amount of ETH sent to contract
amount1_in	the amount of USDT sent to contract
amount0_out	the amount of ETH received from contract
amount1_out	the amount of USDT received from contract

The effective price of the transaction is defined as:

$$P_{ETH} = max(abs(amount0\_In_t/amount\_0Out_t), abs(amount1\_In_t/amount0\_Out_t)) * 10^{12} \quad (1)$$

Note that the 1e12 factor exist in this equation since uniswap internal implementation uses a different unit of ETH.(Adams7 et.al)

4 Methods

4.1 Multilayer Perceptron

A multilayer perceptron (MLP) is a fully connected feedforward artificial neural network that is used extensively for multiple applications[10]. It consists of at least three layers of neurons: an input layer, an output layer, and at least one hidden layer. It utilizes nonlinear activation functions to distinguish itself from a linear perceptron and use backpropagation for training.

4.2 LSTM and GRU

A RNN is a popular method for sequential data modeling as it can use both past and present information for computation. However, if the data sequence is long, RNN won't memorize well and may leave out some important information from previous inputs. In addition, RNN suffers from the issue of vanishing or exploding gradients. The vanishing gradient problem happens when the gradient shrinks as we do backpropagation, so it is harder to update the weights and uses more computing power to reach the final results. Long Short Term memory (LSTM) improves upon the traditional RNNs and deals with this problem. LSTM uses gates to control the information flow[9].

Grate Recurrent Unit (GRU) is like a simplified LSTM: it doesn't have an output gate and combines the input and forget gates in LSTM into one update gate. The reset gate in GRU will determine how much past information to forget. Both are commonly used memorization techniques in RNN. In

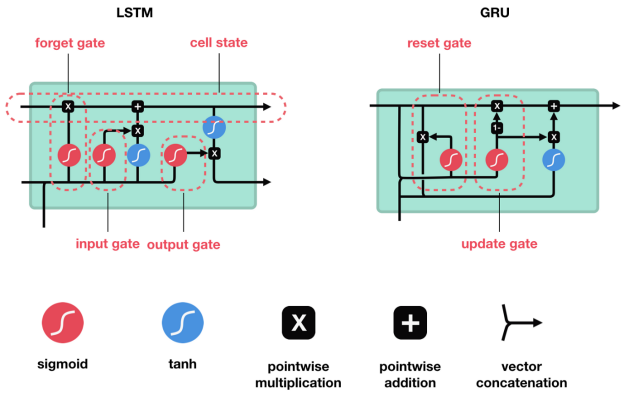

Figure 2: LSTM and GRU[8]

general, GRU has similar performance with LSTM but is less complex since it has fewer parameters to train and uses less memory. We will experiment with both structures and use validation MAPE to pick the better one.

5 Experiments/Results/Discussion

5.1 Baseline Model

From the price data we compute log return and estimate volatility. We currently have data from May 26 2020 to October 31 2022 at 10-min intervals for volatility. We split the most recent two months as our test set, the next most recent two months as our validation set, and the rest as our training set. We first experiment with a Vanilla LSTM for univariate time series forecasting, using past volatility as inputs and MSE as evaluation metrics. However, this experiment takes 4 hours to finish and gives an extremely small training MSE of 0.00014 and validation MSE of 0.00008. Considering the scale of our inputs and the limit in computational power we have access to, we change our evaluation metrics to Mean Absolute Percentage Error (MAPE) and restarts with a Multilayer Perceptron (MLP) model as our baseline. Our baseline model consists of 2 dense layers with 64 units and 1 unit respectively, with ReLU as our activation function, MAPE as our loss function, and Adam as our optimizer.

Table 1: Baseline Models

Model	Days Look Back	Num Epoch	Train MAPE	Dev MAPE	Loss Function	Input
MLP(64+1)	2	25	42.8545	36.5098	MAPE	Past Volatility
Exponential Moving Average	7	N/A	4828507805.72	0.0905	N/A	Past Volatility

As per TA's suggesstion, we also included another simple baseline which is the exponential moving average of past vol for comparison that we also attached below the MLP table. Note that the 'train MAPE' is very large in the simple expontential moving average model (EWMA approach), since MAPE can easily be increased/inflated by outliers. For example, the EWMA initially will give 0 since the initial vol history is short and will gradually factor into enough information after accumulating enough observations. Therefore, for us to compare our enhanced LSTM model result, we are focusing in comparing the correlation statistics compared to this EWMA approach. Now in the train dates, EWMA prediction has a correlation of **0.758** to train set and a correlation of only **0.102** to dev set period.

5.2 Hyperparameter Tuning of Baseline Model, Metrics Update, and Further Preprocessing

We test the performance of MLP with different look back steps and different numbers of training epochs. The train MAPE and validation MAPE both drop as we increase the number of days look back from 2 days to 7 days. However, as we elongate the training epochs, the errors become large. We suspect that this is because 2 dense layers cannot capture the complex relationship in volatility, so

Table 2: More Tuning of Baseline Models

Model	Days Look Back	Num Epoch	Train MAPE	Dev MAPE	Loss Function	Input
MLP(64+1)	2	25	42.8545	36.5098	MAPE	Past Volatility
MLP(64+1)	7	25	35.0078	29.6773	MAPE	Past Volatility
MLP(64+1)	7	50	370.4697	312.1855	MAPE	Past Volatility
MLP(128+64+1)	14	50	204.8080	244.8658	MAPE	Past Volatility
Vanilla LSTM(64+1)	2	10	164.9276	140.0614	MAPE	Past Volatility
Vanilla LSTM(64+1)	2	25	43.5881	36.7647	MAPE	Past Volatility
MLP(64+1)	14	100	0.1520	0.1330	MSLE	24h Rolling Volatility

we upgrade to 3 dense layers with 128 units, 64 units, and 1 unit respectively. The errors drop but the performance is still bad. To further improve our model, we upgrade to a LSTM structure.

We rerun the experiments with Vanilla LSTM with 64 units followed by one dense layer. We first try the model with 10 epochs which gives us a large train and validation MAPE. To improve our fit, we rerun the training process with 25 epochs, which gives us similar performance as MLP.

We notice that almost all our validation MAPEs are smaller than train MAPEs. We plot out our predictions and find that our model is severely impacted by some large jumps of volatilities. In addition, our predictions have a tiny range that is not so useful in the real world. To solve this issue, we take several actions. First, we further smooth out the data to 24-hour rolling volatility. Second, we update our loss function to Mean Squared Logarithmic Error(MSLE) because MAPE tends to stay large even when the training epoch number is big. We still keep MAPE as our evaluation metric. As a simple test of our remedy actions, we rerun our MLP model with 24-hour rolling volatility and MSLE and it reduces our training MAPE to 0.1520 and validation MAPE to 0.1330.

5.3 LSTM

we explored LSTM with different combinations of data. Note that we borrowed a relatively good set of hyperparameters that we have explored in the baseline scenario : 2 days look back and 64 nodes for LSTM. Note that in this version of model, we are running 100 epoch using 64 batches. The reason is that usually, having batch less than 1 can result in faster convergence. First, we are adding the 10 mins interval price moments to the model input. The moments we used are: First: First price of every 10 mins interval; Last: Last price of every 10 mins interval; mean: average of every 10 mins interval; max: max price of the 10 mins interval and min: minimum price of every 10 mins interval.

Since our baseline exploration shown that for LSTM, the 2 day look back is the best look back period, we decided to experiment with 2 day look back and start with 64 nodes. We experiment with two types of data combinations such as shown in table 3. This further decreases our training MAPE to 0.1332 and dev MAPE to 0.0995. Even though the test MAPE is similar to the naive EWMA base (dev MAPE 0.0905), the correlation is a strong measurement of the performance of the model. Prediction from Vanilla LSTM(64+1) using 24 Rolling Volatility and Price moments has a correlation of **0.988** to train set and a correlation of impressive **0.972** to dev set period. We also tested test and get a MAPE of 0.0896 and a correlation of **0.974** since it's our final proposed model. The time series plot can be seen in figure 3.

Table 3: Different Data Combination and LSTM model result

Model	Days Look Back	Num Epoch	Train MAPE	Dev MAPE	Loss Function	Input
Vanilla LSTM(64+1)	2	100	0.1332	0.0995	MSLE	Past Vol and Price moments

6 Project Novelty

The innovation for this project is applying new and challenging data sources such as the Uniswap data in volatility forecasting. Volatility forecasting has been first traditionally done by statistical time series models such as GARCH, the application of RNN and LSTM on the volatility forecasting is relatively new for finance field. From Uniswap's liquidity perspective, volatility is a major risk to the business model. Understanding sources of such risk in a quantitative manner has not been done in the past.The challenge is to under stand the impact on-chain data and its impact on volatility, and looking into different sources of data and Defi protocals. Another challenge is since there is

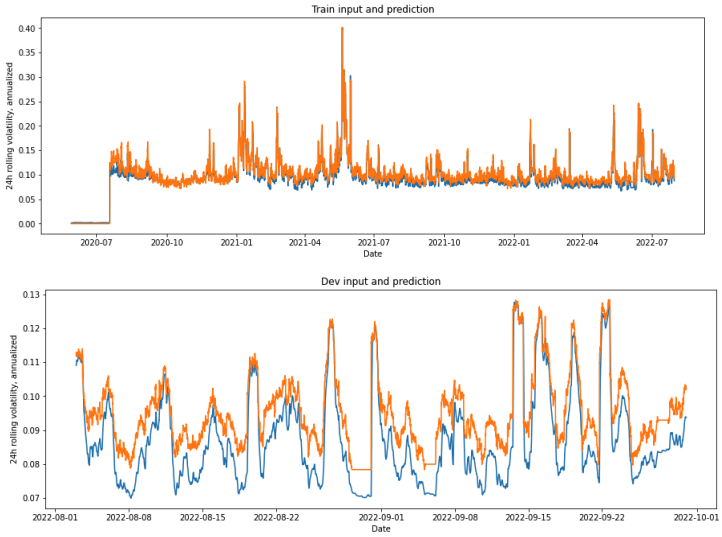

Figure 3: (a) Train set input vs prediction (b) Dev set input vs prediction

no pre-trained model, we looked into the process of training the model from scratch, which was extremely challenging and time consuming.

7 Conclusion/Future Work

We recommend a LSTM model in volatility forecasting using price information and historical volatilities. In our baseline exploration, we have found out that MAPE is a better training loss function, and we consistently measure the MAPE across different models. However, MAPE has its drawbacks due to two reasons:First, when we are adding more data to the model, even after min max scaling, the model was difficult to stop training and converging. We shifted to MSLE to avoid this issue. Also, due to the complexity and training time required for LSTM, we have found the best data combination to be past volatility and price moments to predict future volatilities. In the future, we are looking to expand more data sources in the framework, potentially linking other volatility data sources (ie, SP500 volatility).

8 Contributions

Xiaotian Lu: Knowledge transfer/research for the uniswap and onchain data. Created data extraction and data generation engine for blockchain data. Extracted other alternative protocol data to expand the model. Data preprocessing (Aave protocol). Model experiment with additional features and normalization methods. Set up the AWS. Final report writing.
Lai Xu (Shirley): Setting up github repo, data analysis,data processing, validation and cross checking with Yahoo finance public data. Generate model inputs and outputs (ie, prices, vols) for testing purposes.Final LSTM experimentation and the optimal model with LSTM. Final report writing.
Riley Yang: Build the baseline model with processed data.Write model methodologies. Experimented with different baseline model including MLP and LSTM. Hyperparameter tuning. Update the results for Milestone report. Final report writing.
Link to Our Github: https://github.com/thymeshirley/crypto_vol_prediction

References

[1] Hu, Yan, et al. “A hybrid deep learning approach by integrating LSTM-ANN networks with GARCH model for copper price volatility prediction.” Physica A: Statistical Mechanics and its Applications, vol. 557, 2020, https://www.sciencedirect.com/science/article/abs/pii/S0378437120304696.

[2] Chung, Junyoung & Gulcehre, Caglar & Cho, KyungHyun & Bengio, Y.. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

[3] Predicting Ethereum prices with machine learning based on Blockchain information, Han-Min et al, 2021

[4] Deep Learning Algorithm to Predict Cryptocurrency Fluctuation Prices: Increasing Investment Awareness, Mohammed et al, 2022

[5] Deep learning in predicting cryptocurrency volatility, Valeria et al, 2022

[6] Phi, M. (2020, June 28). Illustrated guide to LSTM's and GRU's: A step by step explanation. Medium. Retrieved November 13, 2022, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[7] Singhal, G. (2020, September 9). LSTM versus GRU Units in RNN. Retrieved November 13, 2022, from https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn

[8] Wikimedia Foundation. (2022, September 8). Multilayer Perceptron. Wikipedia. Retrieved December 1, 2022, from https://en.wikipedia.org/wiki/Multilayer_perceptron