Prediction of Ether Prices Using DeepAR and Probabilistic Forecasting

Andras Ferenczi[1][0000−0001−6785−9416] and Costin Bădică[1][0000−0001−8480−9867]

University of Craiova, Craiova, Romania
ferenczi.andras.h5f@student.ucv.ro,costin.badica@edu.ucv.ro

Abstract. Ethereum is a decentralized public blockchain powered by its native cryptocurrency, the Ether ($Ξ$), which is second to Bitcoin (BTC) in market capitalization. To ensure the integrity of the network, Ethereum requires a fee for every transaction. This fee is called gas (by analogy to the fuel used by cars) and can fluctuate based on supply and demand. This volatility stepped up a number of initiatives to predict future gas prices. The paper proposes a novel solution beyond current state-of-art using DeepAR. This is a model built on Recurrent Neural Networks with the ability leverage hundreds of related time series to produce more accurate predictions. Our implementation uses features extracted from the Ethereum MainNet as well as off-chain data to achieve accurate predictions.

Keywords: Ethereum · blockchain · gas price · proof of work · machine learning · prediction · probabilistic forecasting · DeepAR.

1 Introduction

Ethereum is one of the most popular blockchains and its cryptocurrency is ever increasing in value. Just as other blockchains, such as Bitcoin, it uses a consensus algorithm called Proof of Work (PoW), a.k.a. mining, to maintain the integrity of the blockchain and to prevent double spend. The blockchain provides financial incentives to miners to perform PoW in the form of newly-minted coins and transaction fees paid by users wanting to perform transactions. In Ethereum these fees are called "gas".

The term comes from the analogy that a car needs fuel to run and gas is the fuel that helps recording of transactions on the distributed ledger. Gas is the unit of measurement of computational power required for a miner to process a transaction and is measured in $WEI = 10^{-18}ETH$. The price of the execution of a transaction, a contract, or a deployment for a smart contract is $GasCost \cdot GasPrice$ [10]. Just as fuel prices in real world, gas price may vary being subject of a negotiation process. The sender of a transaction specifies the maximum amount they are willing to pay, just as the miner has the options of accepting, partially refunding, or rejecting the offer.

Ethereum is also a hotbed for innovative Decentralized Applications fueled by Smart Contracts [21], the foundation for tokens representing digital assets or even real-world objects [14], [18]. Token transfers, as much as cryptocurrency transactions have an impact on gas prices, and hence provide valuable information for our prediction model.

2   Background & Related Works

2.1   Background

The Ethereum Blockchain A block in the blockchain contains a header and several Merkle Patricia Trie structures [19], including one that has the transactions in it. Our model uses a limited number of fields as presented in Table 1.

Table 1. Block header fields

Item	Description	Source
Timestamp	When block was assembled.	Block Header
Gas used	Total gas used for block.	Block Header
Block number	Index of block.	Block Header
Hash	Transaction hash	Transaction
Gas	Gas paid by sender	Transaction
Gas price	Price in Wei	Transaction

The Estimation Model For this experiment we decided to use DeepAR [13]. Amazon SageMaker DeepAR is a tool that implements an unsupervised forecasting model based on autoregressive recurrent neural networks (RNN).

Unlike other forecasting methods, such as autoregressive integrated moving average (ARIMA) and exponential smoothing (ETS), DeepAR can learn a global model from multiple time series. The empirical experimental results produced by the Amazon team [13] show an improvement on standard metrics of up to 15% compared to state-of-the-art methods, such as Facebook's Prophet [16].

2.2   Related Works

Three methods to analyze and predict gas prices are highlighted in [9]. The first method assumes the analysis of pending transactions in large Mempools [6]. Mempool is a buffer area where pending transactions sent by Ethereum clients are stored before they are added to the Ethereum blockchain. This method proves to be resource-intensive and complex to implement as it requires access to multiple Mempools and also it assumes that the owners of these Mempools are honest.

A second method analyzes recently committed blocks using oracles. These are systems that connect the blockchain to the outside world. Specifically, gas price oracles provide guidance to users regarding the gas price to pay to ensure that miner will accept the fees and commit the submitted transactions into subsequent blocks [15]. Examples include Ethereum client, Geth [8], EthGasStation [3], Gas Station Express [4].

A forecasting model based on Gated Recurrent Unit (GRU) [7] and a Gas Recommendation engine that leverages the output of the forecasting model was proposed in [20]. The approach used a Neural Net model that also included an additional parameter

that reflects the urgency of the transaction (the higher the gas price, the faster the transaction is committed). The model reduced fees by more than 50% while increasing the waiting time by 1.3 blocks, when compared to the GETH oracle.

Rawya Mars et al [12] evaluated the LSTM, GRU and Prophet models [16] to anticipate gas prices. An empirical evaluation resulted in better outcomes from LSTM and GRU models than Prophet model and the GETH oracle.

A Gaussian process model to infer the minimum gas price is presented in ChihYun et al [10]. Gaussian process is a non-parametric Bayesian approach to estimate a posterior over functions based on prior over functions using test data. This model performs better than GasStation-Express and Geth only when gas prices fluctuate widely. For this reason, they propose a hybrid solution combining GasStation-Express with their model.

3   Experiments and Results

3.1   Data Collection and Pre-processing

According to Salinas et al [13], the covariates can be item and/or time dependent. For collecting the historical blockchain data, we used the Kaggle Ethereum Blockchain Complete live historical data (BigQuery) [2], as well as live minute-by-minute Ether (ETH or Ξ) and Polygon **MATIC** prices from cryptodatadownload.com [1], as seen in Table 2. Our Jupyter notebook [11] prepared the data for the training.

Table 2. Features used for training and inference (minute-by-minute intervals). The target time series are the Gas Prices and there are 6 additional dynamic features. During our experiments we found MATIC prices not to be helpful

Feature	Description	Source
Gas prices	Gas prices by transaction	BigQuery *transactions* [2]
Ξ Prices	Ether to USD price minute-by-minute	Binance exchange data [1]
MATIC Prices	MATIC to USD price minute-by-minute	Binance exchange data [1]
Transaction values	Value transfer by transaction	BigQuery *transactions* [2]
Committed transactions	Transactions in blocks	BigQuery *blocks* [2]
Token transfers	# ERC20 token transfers	BigQuery token_transfers [2]
Contract events	# ERC20 token events	BigQuery *logs* [2]
Gas used	# units of gas per transaction	BigQuery *traces* [2]

For the training and validation phase, we processed the mean of all the time series for every 20 minutes. After experimenting with various time series frequencies, we chose 20 minute intervals as the best for this type of data. Ethereum gas prices fluctuate widely and hence the data is very noisy. In spite of not smoothing the data by eliminating outliers, our model performed very well. In the end, we had data processed at 20 minute intervals for 291 days (January 1, 2021 to October 18, 2021). We used 80% of the data for training and 20% for validation.

3.2   Experimental Setup

We built a Python Jupyter notebook [11] and used Gluon Time Series (GluonTS) [5] for probabilistic time series modeling. The *DeepAREstimator* is an implementation of the model described by Salinas et al [13]. We have configured the estimator as follows:

– Prediction length of 40, thus providing $40 \cdot 20 = 800$ minutes = 13 hours and 20 minutes.
– Architecture of 4 layers with 40 cells per each layer.
– Dropout rate = 0.1.
– Context length (number of steps) of 80 (double of prediction length). Context length is the number of points provided to the model to make the prediction.
– Cell type GRU. Note that we experimented with LSTM cells as well, although we did not notice any significant improvement, but rather a slight slow-down of the training.
– The learning rate callback had the following settings: patience=10, base LR=10^{-3}, decay factor=0.5.
– Training was configured to run for 200 epochs.
– We selected the checkpoints from 2 models, based on the best metric values.

The experiments were performed on a desktop computer equipped with Intel Corporation Xeon E3-1200 v6/7th GenCore Processor with 32 GB RAM and 240GB SSD, NVIDIA GeForce GTX 1080 TI GPU, running *Ubuntu 18.04*.

3.3   Experimental Results

The tests were run for various date/time targets, and we noticed empirically an overall improvement in metrics of predictions as we added more features.

To find the best combination of features, we performed a greedy approach. First, we selected the feature with the highest impact by running the algorithm 7 times. Since using the MATIC prices gave worse results than using 0 dynamic features, we dropped this data from subsequent tests. Once we found the feature with best results, we ran the training and inference with the remaining 5 feature data and selected again the one giving best results. We continued the process for the rest of the features, thus performing a total of $7 + 5 + 4 + 3 + 2 + 1 = 22$ trials.

Given y_i as the observed value, $\hat{y}_i$ the predicted value and n the number of samples, we computed the following metrics using *sklearn* package:

1. Mean Absolute Error (MAE): $MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$
2. Quantile Loss (QL) for a given quantile q, defined as: $L(\hat{y}_i, y_i) = \max\{q(\hat{y}_i - y_i), (q - 1)(\hat{y}_i - y_i)\}$. This value is averaged across all predictions. We compared the values obtained for the following quantiles: $q \in \{0.1, 0.5, 0.9\}$.
3. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) defined as: $MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$ and $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} = \sqrt{MSE}$. To calculate the RMSE metric, we used data normalization by re-scaling the test and predicted data to have a mean of 0 and variance of 1.

Our best performance result with 5 dynamic features is presented in Figure 1. Table 3 shows the values obtained for each metrics, depending on number of features.
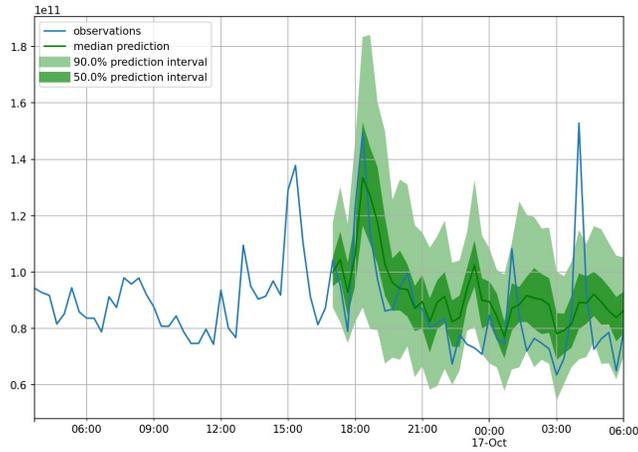
Fig. 1. Prediction with 5 dynamic feature inputs: Ξ prices, Transaction Values, Committed Transactions, Token Transfers, and Gas Used

3.4   Discussion

Time series forecasting is one of the most important tools used by businesses, and there are a number of frameworks data scientists can use for this purpose. As expected, there is no "silver bullet" for any problem. The empirical research conducted by Žunić et al [17] concludes that DeepAR (AWS) models "show superiority over classical methods only when they have a large number of signals over which to create a model, and in the case of articles with a short history." Since cryptocurrency prices in general and Ξ in particular have numerous covariates that can be used, one can perform accurate predictions with shorter history, a benefit that low-power (e.g. IOT) edge devices can take advantage of when deciding on the timing for submitting transactions to the blockchain.

Table 3. Comparison of metrics by number of dynamic features involved. Values have been converted to Ξ ($10^{18} WEI$).

Metric	0 feats.	3 feats.	5 feats.
MAE	19	15	12
MSE	623	395	286
Quantile Loss [0.1]	292	195	173
Coverage Quantile [0.1]	0	0.225	0.250
Quantile Loss [0.5]	769	636	496
Coverage Quantile [0.5]	0.1	0.82	0.775
Quantile Loss [0.9]	477	373	284
Coverage Quantile [0.9]	0.625	0.975	0.975
RMSE	25	20	17
Normalized RMSE	0.287	0.229	0.194

Our analyzed data ranges between January 1, 2021 and October 18, 2021. During this time, gas prices ranged between 10 and 4315 GWEI, or a 431.5% fluctuation. As of this writing, a regular Ξ transfer has a limit of 21,000 units of gas. At the given price range, one would have to pay anywhere between 0.00021 and 0.9 Ξ. At current exchange rate of 1 Ξ to 4,189 USD this comes to roughly between 1 and 3,770 USD for a simply sending Ξ to a different address. This clearly shows the importance of timing these transactions based on accurate predictions.

Although our experiment has room for improvements, it shows the power of probabilistic forecasting using DeepAR. Using this approach we were able to obtain accurate predictions in spite of a noisy dataset. DeepAR requires minimal feature engineering. We performed down sampling but did not remove outliers. We did, nevertheless, have to perform normalization, in spite of suggestions in the literature otherwise. Our model did not converge without normalizing the features first. By adding dynamic features, we achieved improvements in the prediction metrics (see Table 3). As the figures show, DeepAR's Monte Carlo sampling-based quantile estimates are accurate and can be very useful in practice.

4   Conclusions and Future Works

Empirical analysis of Ethereum gas price prediction with DeepAR proves that carefully chosen covariates can improve model performance. Gas prices are impacted by various factors, including seasonality, volume of transactions, transaction values, number of token transactions, Ξ price, amount of gas used per block. Our focus in the future will be to identify additional features that can improve the performance of our model, by researching factors that have an impact on the supply and demand for gas. Such examples may include off-chain data, such as twitter or other social media events that may influence the volume of transactions on the Ethereum blockchain and, hence indirectly, gas and/or Ξ prices.

As DeepAR performs better than Facebook's Prophet with a smaller amount of sales data [17], we will research potentials for deployment of our models on low-power connected devices.

References

1. Binance eth to usd minute-by-minute, `https://www.cryptodatadownload.com`
2. Ethereum blockchain. complete live historical ethereum blockchain data (bigquery), `https://www.kaggle.com/bigquery/ethereum-blockchain`
3. Ethereum gas station, `https://github.com/ethgasstation`
4. Gas station express oracle, `https://github.com/ethgasstation/gasstation-express-oracle`
5. Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D.C., Rangapuram, S.S., Salinas, D., Schulz, J., Stella, L., Türkmen, A.C., Wang, B.: Gluonts: Probabilistic time series models in python. ArXiv **abs/1906.05264** (2019)
6. Blocknative: Gas estimation for builders, by builders, `https://www.blocknative.com/gas-platform`

7. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1724–1734. ACL (2014). https://doi.org/10.3115/v1/d14-1179

8. Chuang, C.: Official go implementation of the ethereum protocol, `https://github.com/ethereum/go-ethereum/`

9. Chuang, C.: A practical and economical of gas price prediction, `https://medium.com/getamis/a-practical-and-economical-of-gas-price-prediction-d9abe955ac63`

10. Chuang, C., Lee, T.: A practical and economical bayesian approach to gas price prediction. In: Awan, I., Benbernou, S., Younas, M., Aleksy, M. (eds.) The International Conference on Deep Learning, Big Data and Blockchain (Deep-BDB 2021). Lecture Notes in Networks and Systems, vol. 309, pp. 160–174. Springer (2021). https://doi.org/10.1007/978-3-030-84337-3_13

11. Ferenczi, A.: Using probabilistic forecasting for gas price prediction in ethereum blockchain, `https://github.com/andrasfe/ethgas`

12. Mars, R., Abid, A., Cheikhrouhou, S., Kallel, S.: A machine learning approach for gas price prediction in ethereum blockchain. In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 156–165 (2021). https://doi.org/10.1109/COMPSAC51774.2021.00033

13. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020). https://doi.org/10.1016/j.ijforecast.2019.07.001

14. Simionescu, S.M., Bădică, A., Bădică, C., Ganzha, M., Paprzycki, M.: On the role of blockchain in evolving the online business landscape. In: 2021 Conference on Information Communications Technology and Society (ICTAS). pp. 85–90. IEEE (2021). https://doi.org/10.1109/ICTAS50802.2021.9395045

15. Smith, C.: Ethereum oracles. ethereum.org (2022), `https://ethereum.org/en/developers/docs/oracles/`

16. Taylor, S.J., Letham, B.: Forecasting at scale. PeerJ Preprints **5**, e3190v2 (2017). https://doi.org/10.7287/peerj.preprints.3190v2

17. Žunić, E., Korjenić, K., Delalić, S., Šubara, Z.: Comparison analysis of facebook's prophet, amazon's deepar+ and cnn-qr algorithms for successful real-world sales forecasting. International Journal of Computer Science and Information Technology **13**(2), 67—-84 (Apr 2021). https://doi.org/10.5121/ijcsit.2021.13205

18. Wackerow, P.: Erc-20 token standard. ethereum.org (2021), `https://ethereum.org/en/developers/docs/standards/tokens/erc-20/`

19. Ward, C.: Merkle patricia tree (2020), https://eth.wiki/fundamentals/patricia-tree

20. Werner, S.M., Pritz, P.J., Perez, D.: Step on the gas? a better approach for recommending the ethereum gas price. In: Pardalos, P., Kotsireas, I., Guo, Y., Knottenbelt, W. (eds.) Mathematical Research for Blockchain Economy, pp. 161–177. Springer Proceedings in Business and Economics, Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53356-4_10

21. Zheng, Z., Xie, S., Dai, H.N., Chen, W., Chen, X., Weng, J., Imran, M.: An overview on smart contracts: Challenges, advances and platforms. Future Generation Computer Systems **105**, 475–491 (2020). https://doi.org/10.1016/j.future.2019.12.019