Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting with applications to cryptocurrency data

Anastasia Yaschenko
anastasia.yaschenko@skoltech.ru

Nikita Glukhov
nikita.glukhov@skoltech.ru

1 Introduction

Over recent years cryptocurrencies have seen a surge in popularity. Being a means of payment in several markets, such as NFT art, they are still largely viewed as an investment asset. However, they are infamous for huge price volatility and lack of justification of their price based on their fundamental characteristics. These factors combined make cryptocurrency price prediction an extremely difficult task. With the widening use of algorithmical trading, investors are put into even greater risk of losses due to inability of their models to correctly predict prices.

We aim to study if the price prediction could be improved by exploiting relationship between different cryptocurrencies. Hence we turn to the task of multivariate time-series forecasting, using correlations between time series for better forecasts. In particular, we focus on graph neural networks (GNN). In such formulation the task consists in predicting node values (which are the values of time series at particular times), given the previous values of time series.

We aim to use StemGNN [4], as it models both intra-series and inter-series correlations. Furthermore, it utilises spectral representation of time series, since it is claimed that after such transformation the patterns become more well-defined, thus improving predictions. Moreover, this model does not require manual conversion of time series into a graph. This is done through the so-called latent correlation layer, which learns latent task-specific correlations between the series.

2 Literature review

Time series prediction is an example of modelling sequential data. A conventional neural network architecture for this task is recurrent neural network. However, recently other types of neural networks started to be used for such purposes. [1] employ convolutional architecture and show that it outperforms common recurrent architectures such as LSTM on a range of tasks. [8] fuse these two architectures into one model that extracts both short-term dependencies and long-term trends. [10] employ similar idea and extract both global and local features to predict multivariate time series.

Graph neural networks (GNN) are starting to be used for time series forecasting. They allow to account for a variety of signals. For instance, [9], [14], [13], [2] leverage both temporal and spatial features, which can be useful in certain areas such as traffic forecasting. Model by [12] allows to integrate external knowledge, such as variable attributes, into the graph. A work which is close to our is by [5], who exploit relationship between stocks and sectors to recommend top-k stocks using GNN with attention module.

3 Model

The authors propose a two step approach. The first step converts matrix X, a multiple time series table of size $N \times T$ in graph datatype, in particular, it returns a pair (X, W), where W is the adjacency matrix. This is done using so-called "Latent Correlation Layer" which feeds X into a Gated Recurrent Unit to obtain hidden representation R and then applie self-attention block to R. The result is the adjacency matrix and operation takes $O(N^2d)$ time, where N is the number of time series. The second step works with the obtained pair and applies so-called StemGNN block. Briefly, this is a block that first applies Graph Fourier Transform (GFT) to capture inter-series relationships, secondly it applies the Spe-Seq Cell that aims to decompose each individual time-series after GFT into frequency basis and learn feature representations on them and finaly Spectral Graph Convolution is used. Authors compare their results on multiple datasets and obtain that their method beats other methods (including LSTM based methods) for all type of errors (RMSE, MAE).

4 Graph filtering

We have tested 2 methods to extract graph backbone – minimum spanning tree and disparity filter [11]. Although most of the edges in the obtained graph have low weight and only few have high weight, we decide to use filtering to find what are the strongest connections between cryptocurrencies. Minimum spanning tree aims to find subset of edges which has the minimum weight and connects all nodes without cycles. There are several algorithms to construct such a tree, such as Prim's algorithm [6], Kruskal's algorithm [7] and Boruvka's algorithm [3]. However, this algorithm may not capture complex dependencies in graph. An alternative is disparity filter, which filters edges based on p-values of normalized edge weights.

5 Data

We use Kaggle dataset with 400+ cryptocurrency pairs with resolution of 1 minute collected from Bitfinex exchange [1]. For each pair, open, high, low and close prices are provided per 1-minute interval, but we use close price only. The reason for using this dataset in particular is that it has more high-frequncy data, which is usually used in algorithmic trading, and large number of pairs. The drawback is that it dates back to 2013, so may not be representative of large price volatility periods and recent trends. However, in relation to our task the benefits of this dataset by far outweight its disadvantages, since our work focuses on testing the model for prediction and is not an empirical study of latest cryptocurrency trends.

The data on all pairs is temporally misaligned, so we choose a subset of cryptocurrencies all of which have enough data for a particular period. We obtain dataset consisting of 95110 observations for each of 12 cryptocurrencies: BTC, ZEC, OMG, UST, LEO, NEO, ETCIOT, EOS, XRP, LTC and ETH. We take 80% of the dataset for traing, 10% for validation and the 10% for testing.

6 Experiments

6.1 Trading strategy

We test models with commonly used pairs trading strategy. It is a statistical arbitrage strategy that aims at exploiting long-run relationships between asset prices X and U. First, spread $S_t = X_t - U_t$ is calculated, as well as its mean and standard deviation. Then, 3 thresholds are determined: α_L for buying assets, α_S for selling and α_{exit} for exiting position. If spread crosses α_L, trader buys X and sells U, if it crosses α_S, then sell X and buy U. Otherwise, if α_{exit} is crossed, the position is exited from. However, the thresholds are not trivial to determine. Improvement of the classical strategy lies in accounting for predicted asset prices.

We undertake the following approach based on pairs trading strategy. First, we need to determine the most profitable pairs. We simulate trading with sliding window of size 750 for baseline ARIMA (1,0,1) model and of 20 for StemGNN. However, to trade we need to determine thresholds α_L and α_S for each stock individually. We approximate them as follows: take validation dataset and calculate spreads on it. After that, determine percentage change in spread. Finally, calculate 0.2 and 0.8 quantiles and 0.1 and 0.9 deciles of this distribution of spread changes. Then simulate trading the pair on test dataset with given set of thresholds and calculate portfolio returns for all periods. It is

[1]`https://www.kaggle.com/datasets/tencars/392-crypto-currency-pairs-at-minute-resolution`

calculated with respect to capital invested, which equals 1000000 for whole portfolio. Then select set of thresholds that gives highest mean return. Commission of 0.1% is taken into account. For each period, the trading algorithm with fixed thresholds is as follows:

- Window of size 750 is taken for baseline model ARIMA and of size of 20 for StemGNN. Spread $X_t - U_t$ is calculated and is used to fit the model. This model is used to make a 1 step prediction of spread, which is used to calculate expected spread change.
- If this change is less than the lowest threshold α_S then we sell all of X and invest all in U If this change is larger than highest threshold α_L, sell all of U and invest in X.
- The portfolio is revalued on each day prior to any trades.

6.2 Metrics

We train with mean squared error loss and use the following metrics to validate model‘s performance (y - target price value, $\hat{y}$ - predicted price, n - size of validation dataset):

Mean absolute error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Mean absolute percentage error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i}|$$

Root mean squared error (RMSE)

$$MAE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

6.3 Training details

We trained model using Google Colab GPU. We used Adam optimizer with learning rate 1e-4 and default parameters and experimented with batch size, prediction horizon and moving window size. The training and validation plots can be found in Appendix. We tried using batch size of 64 and 32, and found that smaller value leads to smaller error. Also, we used different window sizes – 8, 12, 20, 50 and 100 – and found that smaller window sizes lead to more stable training, otherwise validation metrics become volatile. Interestingly, predicting for 3 values ahead instead of 1 positively contributes to the performance. Lastly, we found that using regularization (weight decay = 1e-6 in Adam optimizer) negatively impacts validation performance.

6.4 Graph filtering

We obtain the following graph from adjacency matrix W of the trained model

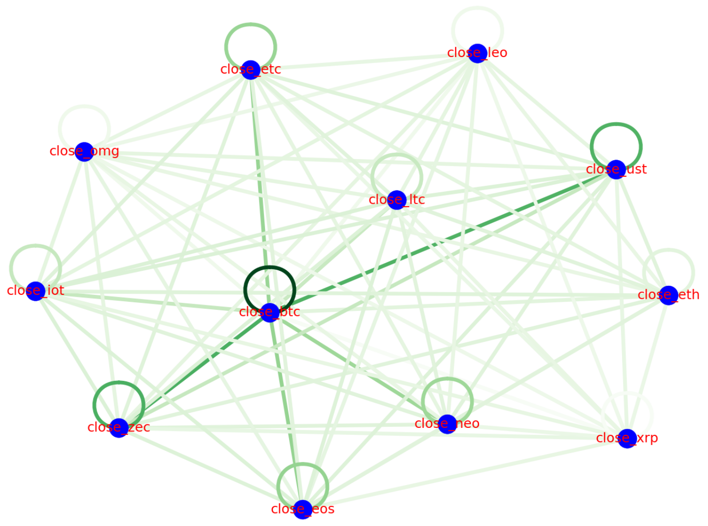

Figure 1: Graph obtained from adjacency matrix W. The brighter the color, the larger is edge weight

As can be seen, such cryptocurrencies as ZEC and UST exhibit significant correlation with BTC, while ETC, EOS and NEO have smaller correlation. Interestingly, all cryptocurrencies correlated with BTC do not show high correlation between each other. All other pairs of assets exhibit low correlation. Overall, the graph supports the expectation that currencies should be centered around BTC.

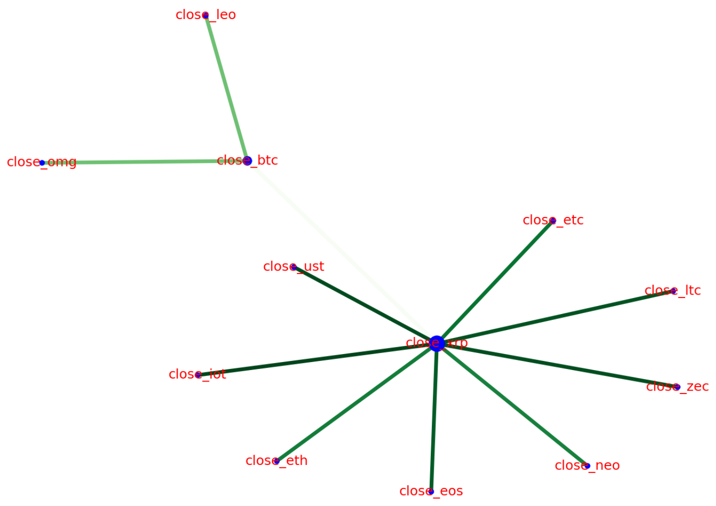

Figure 2: Graph obtained via minimum spanning tree. The brighter the color, the larger is edge weight

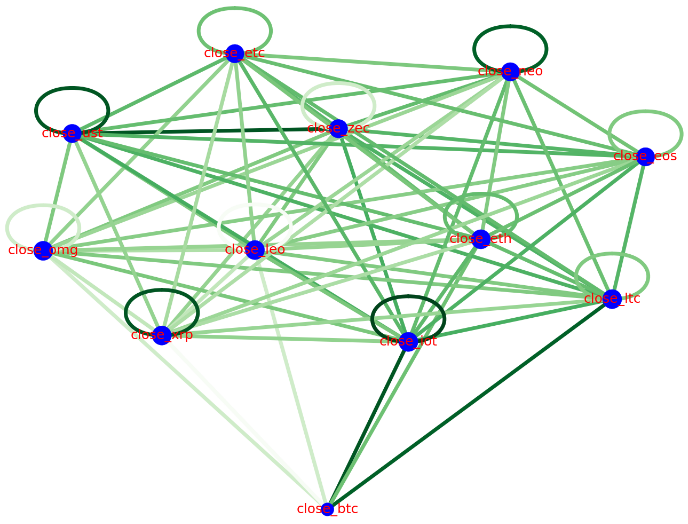

Figure 3: Graph obtained via disparity filter. The brighter the color, the larger is edge weight

It can be noticed that minimum spanning tree indeed simplifies graph structure. It divides the graph into 2 subgraphs: one contains BTC, LEO and OMG, all not significantly correlated, and other cryptocurrecies are highly correlated and included in the second group. Disparity filter recognizes more complex structure of the network, however, it shows that only pairs UST-ZEC, BTC-IOT and BTC-LTC are significantly correlated. Overall, disparity filter algorithm also support the idea that there are only few significantly related cryptocurrency pairs, as opposed to results of minimum spanning tree.

6.5 Trading strategy results

Based on the graph constructed from adjacency matrix W, we identified the following cryptocurrencies with the highest correlation from which 3 pairs were constructed based on validation dataset: BTC, ZEC and UST. As baseline model we used ARIMA (1,0,1) model, which for a time series y_t and forecast error e_t is identified as

$$y_t = a_0 + a_1 y_{t-1} + b_1 e_1$$

Interestingly, StemGNN seems to underperform in comparison with simple ARIMA model, since it does not identify any profitable pairs while ARIMA finds BTC-ZEC and BTC-UST profitable

Pair	StemGNN return (%)	ARIMA return (%)
BTC-ZEC	-1.20	1.00
BTC-UST	-0.10	1.00
ZEC-UST	-0.10	-1.18

However, there are practical limitations in using ARIMA since it is slow in inference and needs larger window size, otherwise its coefficients cannot be robustly computed.

7 Classical approaches of Social Network Analysis to stock market data: sub-project inside our main project

In the light of unsatisfying results with GNN, we decided to conduct some additional experiments for the sake of interest. We follow a more traditional social network analysis path for analysing stock data, collected on a daily basis (day resolution) from 2010 to 2020 (including the coronavirus crisis). Those shares that had NaN values were omitted. We use this dataset as it was collected for some of our research that we did during the BSc.

The goal of the project is two-sided: first, it was our safe option in case of unsatisfying results with StemGNN and second is just of pure interest, as daily stock data can be more subject to fundamental shocks which can be traced using correlations. These correlations are the basis for our analysis.

Our key insights:

- Analyzing stock data using topological metrics of the network bring statistically significant features (for predicting next day prices)

- Analyzing the network of the stocks allows to dynamically observe changes in the clusters of stocks with respect to economical conditions

7.1 Methodology

Strongly generalizing, let's say that the stock has three states over a period of more than 1 day - it is clearly growing, walking sideways or clearly falling. The stock can clearly fall or rise in two main reasons: a) own reason (something related specifically with the company, or specifically with the market for this share, speculators) and/or b) the reason is global (crisis, Friday, moods). If growth occurs for the latter reason, we will have to notice changes in correlations with other stocks. In particular, we filter our prices using some classical statistical time series methods and then use filtered prices to obtain correlations matrices which are considered as graphs and filtering using the Minimum Spanning Tree can be applied. We can use SNA methods to analyze the interpretation of the stock network and to find latent signals (metrics based on the network topology). In particular, we use topological parameters such degree centrality etc. These are considered as features that then are used with an XGBoost regressor to predict next day prices. XGBoost's feature importance indicates that the features based on graph topology are significant in predicting next day prices.

7.2 Results

In figures 4 and 5 we demonstrate the dynamics of stock clusters during the "normal" trading days and turbulent days of coronavirus market crash. We can see that in figure 4, the topology of the network is more "diverse". In particular, stocks tend to cluster in different groups. While in Figure 5 we can see that stocks are very centred and group together in one big cluster. We can see that on average the weights of the edges are smaller (the color is less dark-greenish). In other words, the "distances" are smaller, meaning correlations are stronger. The stocks became more correlated during the market crash which is well-known result. This is good since graph-based analytics allows to trace these effects giving an opportunity for risk management.

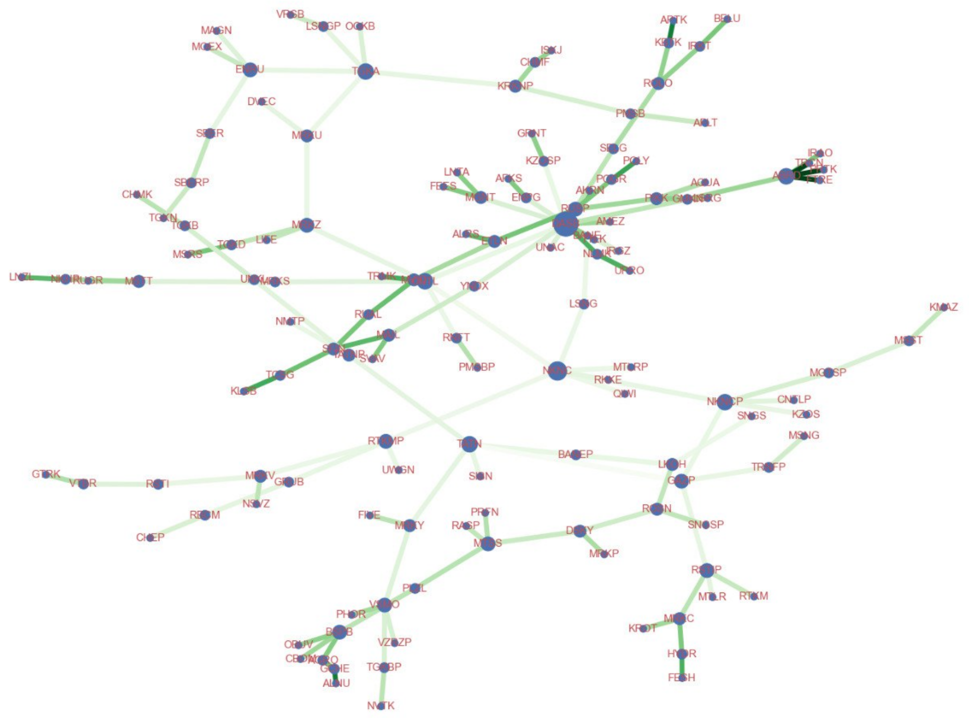

Figure 4: Filtered graph of Russian stocks 5 months before the coronavirus crisis

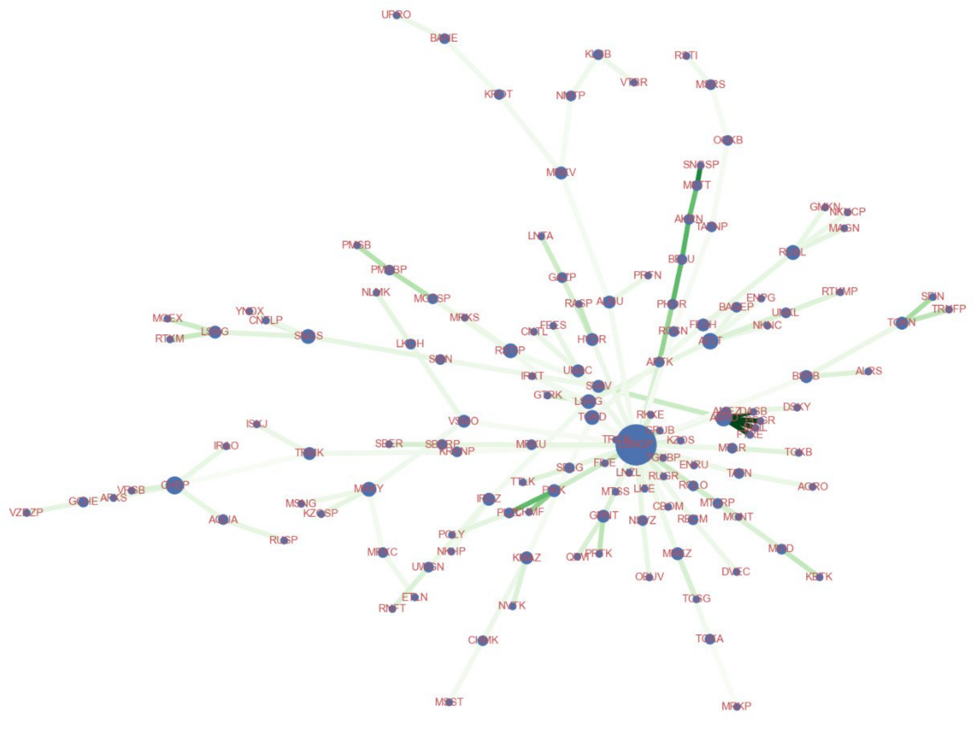

Figure 5: Filtered graph of Russian stocks during the peak of the coronavirus crisis of March-April 2020

8 Appendix

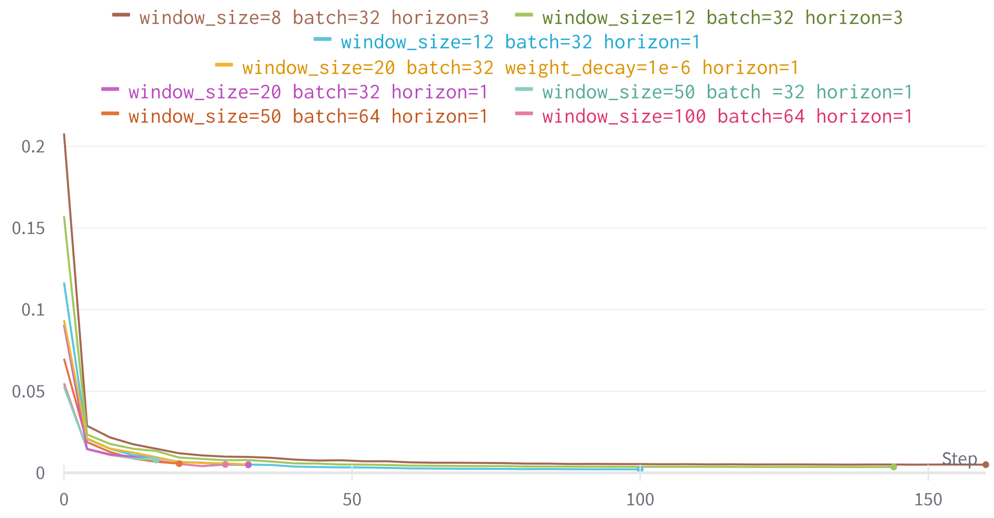

Figure 6: Training loss

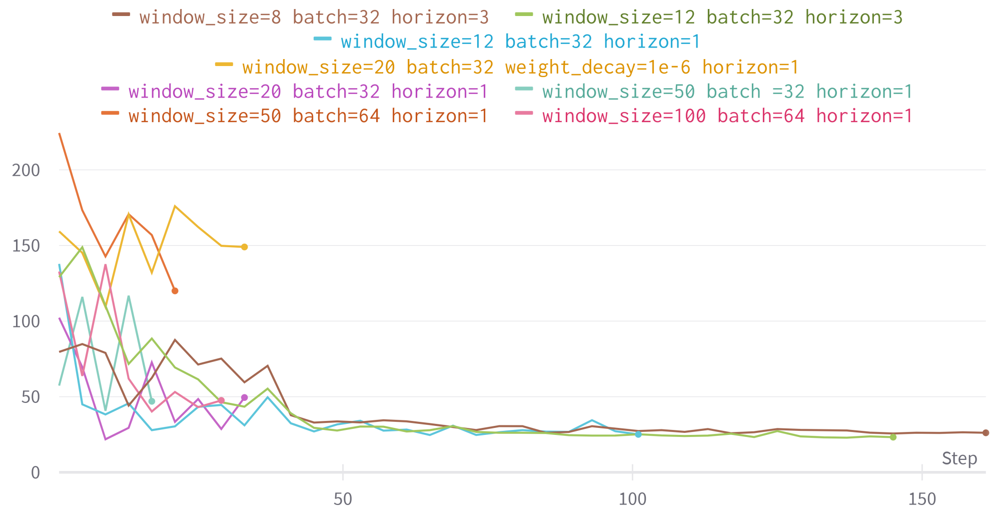

Figure 7: MAE

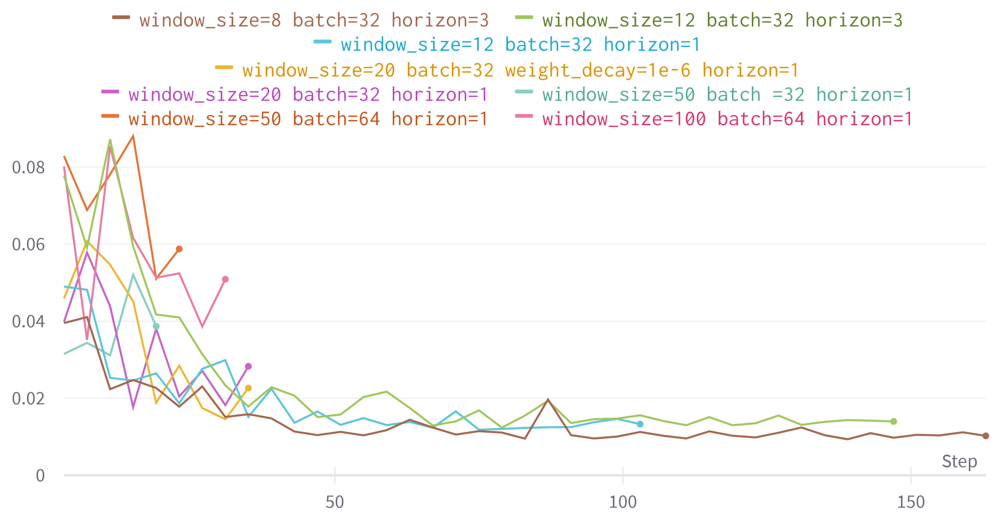

Figure 8: MAPE

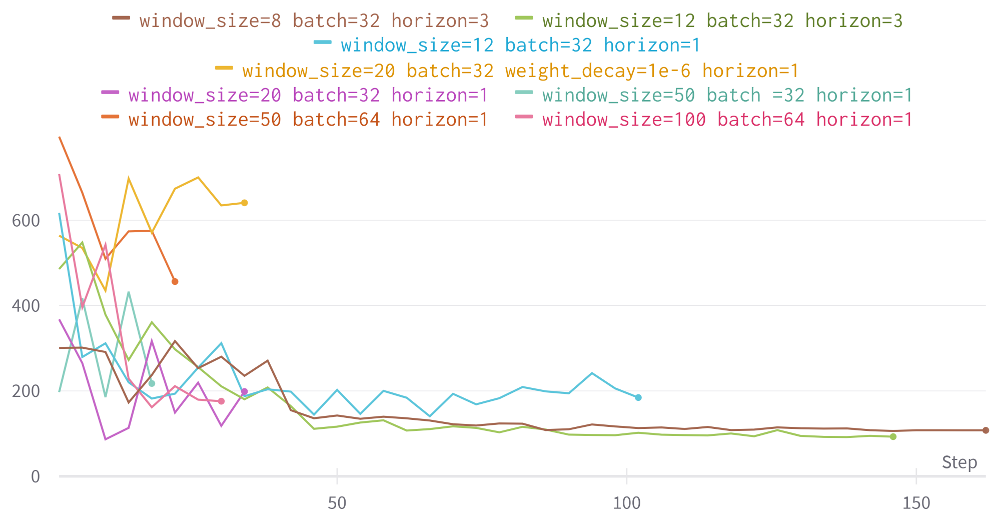

Figure 9: RMSE

References

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[2] Stefan Bloemheuvel, Jurgen van den Hoogen, Dario Jozinović, Alberto Michelini, and Martin Atzmueller. Multivariate time series regression with graph neural networks, 2022.

[3] Otakar Boruvka. O jistem problemu minimalnim. *Prace Moravske prirodovedecke spolecnosti*, III(3):37–58, 1926.

[4] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Spectral temporal graph neural network for multivariate time-series forecasting, 2021.

[5] Yi-Ling Hsu, Yu-Che Tsai, and Cheng-Te Li. Fingat: Financial graph attention networks for recommending top-k profitable stocks, 2021.

[6] V Jarnik. O jistem problemu minimalnim. *Prace Moravske prirodovedecke spolecnosti*, 6(4):57–63, 1930.

[7] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, 7(1):48–48, 1956.

[8] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks, 2017.

[9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2017.

[10] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting, 2019.

[11] M. Á Angeles Serrano, Marián Boguñá, and Alessandro Vespignani. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106(16):6483–6488, apr 2009.

[12] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks, 2020.

[13] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling, 2019.

[14] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2018.