Stock Price Prediction with StemGNN and Dynamic Time Warping

Kwok Chin Yuen
SCSE, NTU
Singapore, G2104201A
kwok0062@e.ntu.edu.sg

Suhwan Chung
SCSE, NTU
Singapore, G2004391L
suhwan001@e.ntu.edu.sg

Pasquale Balsebre
SCSE, NTU
Singapore, G2004597H
pasquale001@e.ntu.edu.sg

Chaiyasait Prachaseree
SCSE, NTU
Singapore, G2103203C
prac0003@e.ntu.edu.sg

How Chun Hung
SCSE, NTU
Singapore, G2100647D
howc0006@e.ntu.edu.sg

ABSTRACT

In time series forecasting, modeling cross-series interactions has become increasingly popular, due to its promising results. The idea of multi-variate time series representation learning has an advantage over its uni-variate counterpart for the opportunity to identify underlying patterns between series that show similar behaviors. The main challenge of the aforementioned technique is to concurrently model both intra- and inter-series interactions. Spectral Temporal Graph Neural Network (StemGNN) [4] is a recent proposal that employs a self-attention mechanism to learn correlations between series. In our work, we improve the original architecture from two perspectives: first, we incorporate Transformers instead of GRU in order to learn the intra-series representation. Second, we construct a robust clustering layer using Dynamic Time Warping to gain prior knowledge on inter-series correlations among stock series. After passing through the the time warping clustering layer, the cluster of joint-series input is fed into StemGNN blocks to evaluate improvements in forecasting accuracy. The dataset used to investigate the benefits of our approach, is built from the set of S&P 500 listed stocks. It is a real-world dataset with prices from a time window of 5 years, and offers great economic opportunities. Extensive experiments, along with an ablation study, prove the effectiveness of our model with respect to the original work. Moreover, our proposed solution proves to be more time-efficient, thanks to Transformers non-recurrent nature, as well as parameter-efficient. A video for the oral presentation of the project is available at: https://youtu.be/XOEGLxBRKfE

CCS CONCEPTS

• **Information systems** → *Entity resolution*; • **Geographic information systems** → *Information integration.*

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Neural networks, graph attention, Graph neural networks, stock forecasting

ACM Reference Format:
Kwok Chin Yuen, Suhwan Chung, Pasquale Balsebre, Chaiyasait Prachaseree, and How Chun Hung. 2021. Stock Price Prediction with StemGNN and Dynamic Time Warping. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Hedge funds and investment banks have been committed to the stock price forecasting task for decades. The applications of such task in finance are tremendous and offer great economic opportunities. An example of stock price prediction is shown in Figure 1: the future stock price of three major tech companies, namely Facebook (Ticker: FB), Microsoft (Ticker: MSFT) and Apple (Ticker: AAPL) is to be predicted. In the proposed work we approach the problem as a *multi-variate* time series forecasting task, given the strong correlation between some companies and their performance in the market. As shown in the figure, the future behavior of each time series is highly dependent on the past as well as on the trend of related stocks that may cause a rise or a fall, delayed in time, thus predictable.

In order to effectively make use of historical data to predict future trends, an algorithm needs to tackle two main challenges: (1) how to represent past behaviors that can be meaningful in the future (intra-series analysis) and (2) how to learn dependencies and interactions among different series to leverage meaningful data from different channels and produce accurate predictions (inter-series analysis).

In this study we propose a revised and improved version of StemGNN [4], able to effectively and efficiently learn dependencies among time series. The contributions of this work can be summarized as follows:

- We reproduce the StemGNN [4] architecture[1], and deploy it to solve the stock price prediction task. This new task is more challenging from the benchmarks on which the original paper is tested, as it presents very high-dimensional time series, due to the amount of stock interactions considered.

[1]To check its correctness we referred to the original implementation, available at: https://github.com/microsoft/StemGNN

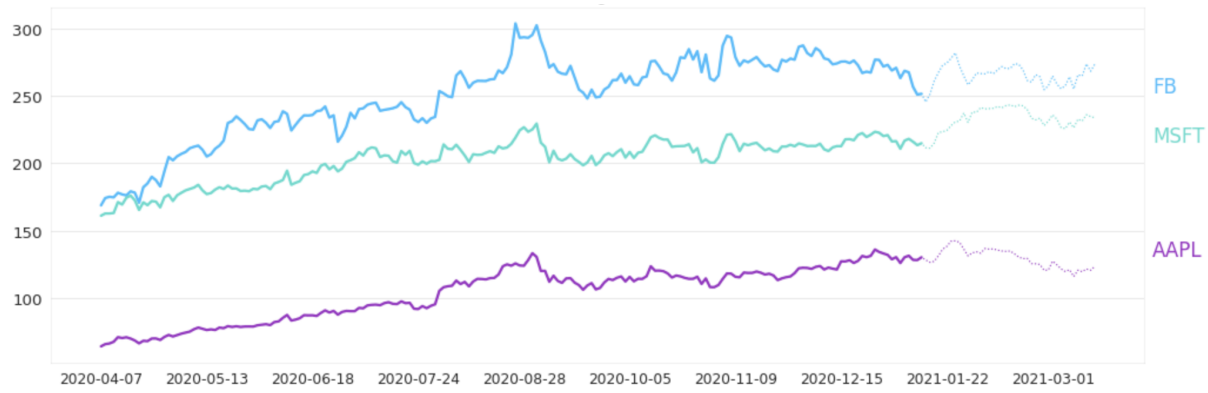

Figure 1: An example of the task of Stock Price Prediction

- We improve the original architecture from two different points of view. First we use Transformers [15] instead of the GRU, in order to learn the intra-series representation and build edges between stocks in the graph; second, we add a clustering layer using Dynamic Time Warping (DTW) to cluster related stocks and avoid interactions between unrelated ones.
- Extensive experiments on the real-world data we collected from Yahoo Finance[2] API in a 5-years period of the price of every stock in S&P500 at the end of the week, prove that our method outperforms previous original works, as well as the original StemGNN. An ablation study shows the effectiveness of our carefully-designed components and strategies.

2 RELATED WORK

In this section we discuss recent approaches to the broad field of Time Series Forecasting as well as solutions proposed to approach the specific problem of Stock Price Prediction.

2.1 Time Series Forecasting

Deep Learning (DL) techniques have become the de-facto standard for time series forecasting and spatial-temporal data analysis. The problems of skeleton-based action recognition [17], traffic flow forecasting [1] and anomaly detection and diagnosis in power plants operation [19] are some examples. A daunting challenge in the use of neural networks to model spatial-temporal data is found in missing observations, both in time and across features. This makes the usage of this kind of architectures tricky. Many proposals aim at learning representation in the spatial-temporal domain by exploring attention and time series correlation. [18] is a pioneering work in the use of Transformers [15] for multi-variate time series representation learning. StemGNN [4] tackles the problem from a different perspective, representing both intra- and inter-series correlations in the *Spectral Domain*. StemGNN achieves state-of-the-art results in a range of benchmarks, but comes with the shortcoming of a prohibitive time complexity, due to the application of eigenvalue decomposition in large graphs of high-dimensional time series.

[2]https://finance.yahoo.com

2.2 Stock Price Prediction

The task of Stock Price Prediction has attracted researchers from both financial and technical domains for decades. The high level of noise and the strong influence of external factors contribute to make it a notoriously difficult task. In [16] Wang et al. applied neural networks to predict stock price, but focused on a single feature, namely the market volume. An SVM model was applied by Ince et al. [9] for short-term forecasting in the same area. Different quantitative analysis have been studied by Liu et al. in [12] as well as a joint CNN-LSTM-based neural network model. While the CNN serves for the stock selection, LSTM preserves the time-evolving nature of the data. Fischer and Krauss [6] also used Long Short Term Memory (LSTM) networks to tackle the same challenge proposed in our work: they used the month-end lists for S&P for a long time window that spans from 1989 to 2015. LSTM layers demonstrate the ability to model long-time dependencies [7] but cannot model the inter series interactions alone. While tremendous advances have been made in the field of stock price prediction, many proposals have failed to adopt cutting-edge deep learning solutions and to effectively model interactions between stocks.

3 METHOD

3.1 StemGNN

In this paper, we discuss on the problem of multivariate time series forecasting. Different from uni-variate case, a series can be simultaneously influenced by multiple parallel events. Hence we aim to learn the inter- and intra-series correlation of multiple series given a fixed window period. We adapt the framework of StemGNN [4], whose novelty is to model multiple time series as graph nodes and their interactions as edges, incorporating graph convolutional operations in the spectral domain. We believe that it is beneficial to work in spectral graph domain for multivariate series as most of the spatio-temporal convolutional models ignore the inter series correlations. With spectral graph convolution, we assume a prior knowledge on the graph relationship, where we cluster and process nearby relevant nodes to gain inter series correlation. In the following we will briefly introduce the mechanism of StemGNN, which

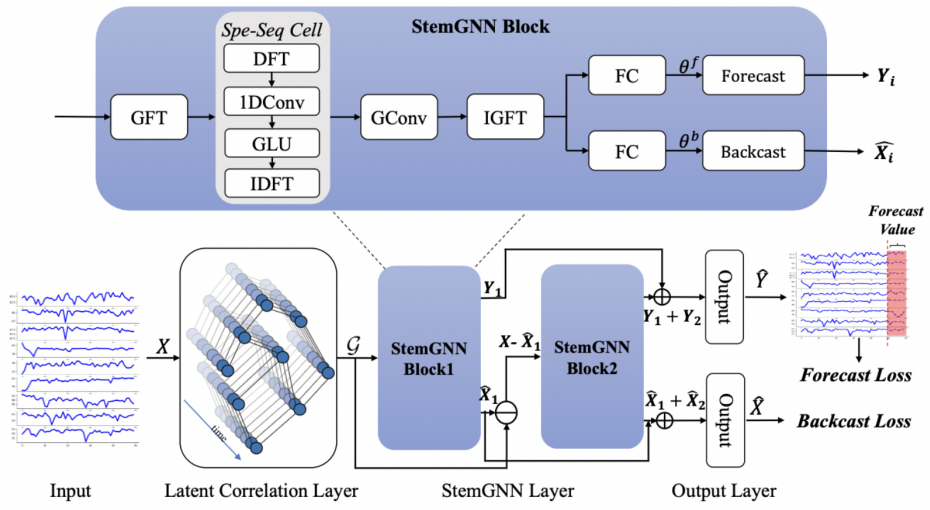

Figure 2: The overall architecture of StemGNN

consists of a latent correlation layer and a sequence of carefully-designed *Stem* blocks. A picture of the StemGNN architecture is available in Figure 2.

Spectral graph convolution network often requires prior adjacency matrix. In the paper, authors propose a self attention mechanism to capture the relationships among features, that will later form the graph structure. Such module is called *Latent Correlation Layer*. It consists in encoding the long series of information and to compute a correlation matrix. In this module, the time series input $X \in R^{NxT}$ is first fed into Gated Recurrent Unit (GRU) to learn its time representation, thus each hidden state of the GRU unit is the aggregated information from every time step. The last hidden state is then used to compute an attention matrix with learnable W^Q and W^K. Since the attention matrix computes the similarity of the hidden states of the series, the attention score can be used to build edges between features in our graph of series. The authors then symmetrize the attention matrix with $\frac{1}{2}(A + A^T)$ to become the adjacency matrix. We then compute the normalized Laplacian distance [13] with $D^{-1/2}(D - A)D^{-1/2}$ where D is the diagonal degree matrix by summing up the rows of A. With the Laplacian matrix, we can compute the orthogonal spectral representation using Chebyshev polynomial approximation. [14]

Assume $X \in R^{Nxd}$ where N is the number of multivariate and d is the dimension of each embedded series, we setup the problem as a graph structure with our graph Laplacian from latent correlation layer. Each temporal series will be projected onto orthogonal spectral domain with the decomposed Laplacian matrix U, this is performed by Graph Fourier Transform (GFT) module in the StemGNN block. Commonly we will implement graph convolution filters on the spectral representation here [3]. However, GFT itself only captures the inter series association, we would also emphasize the connection of intra series. The spectral representation of univariate temporal series will then be decomposed into frequency domain through Discrete Fourier Transform (DFT), a gated linear unit is then applied to aggregated the temporal information linearly. Both operations are in the *Spe-Seq Cell*, which is responsible to learn the intra-series representations. Then, each feature will be mapped back to temporal domain by Inverse-DFT in order to perform graph convolution. After the convolutional module, we project the features back to the input domain using a simple fully connected layer. Eventually, there will be two outputs for forecast and backcast respectively. Backcasting is similar to the idea of auto-encoding, whose rationale is to enhance the representation power. However, in our paper, we are only training with forecast loss which is Mean Squared Error (MSE) by default.

3.2 Transformers For Intra-Series Representation Learning

Transformer architecture [15] has established a new state-of-the-art in a variety of NLP tasks [8, 20]. The success of this architecture is largely due to the *self-attention* mechanism. The output of a Transformer layer is a deeply-contextualized version of the input. In NLP, for example, it can capture the different meanings that a word can have in different contexts. In the *latent correlation layer* of the StemGNN architecture, a GRU is used to learn the aggregated information of each series individually, with respect to time, and cross-attention is successively used to learn interactions and correlations between different features. Although Recurrent Neural Networks, and GRUs, have demonstrated the ability to represent the time evolving nature of a series, respecting the input ordering, they fail to contextualize the input, which can be achieved only by means of attention [2]. We decide to leverage attention mechanism also for the time-evolving representation of each series: we argue, in fact, that a better representation can help to subsequently compute the similarity among features. Learnable positional encodings are used to preserve the ordering, due to the non-recurrent nature of Transformers. A comprehensive picture of our enhanced latent correlation layer is displayed in Figure 3.

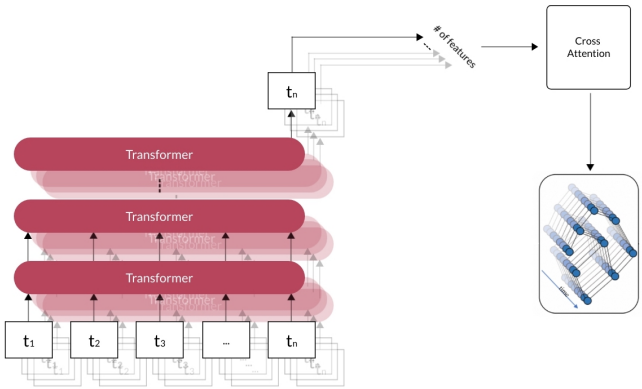

Figure 3: Our Transformer module adapted into Latent Correlation Layer.

3.3 Dynamic Time Warping

In this paper we focus on a topic of cross-series training through the implementation of StemGNN networks in which self-attention mechanism is used as a correlation layer within the networks to group correlated series of stocks. Here, we added an additional clustering layer to obtain prior knowledge on the subgroups of stock series and pass those individual grouped series through correlation layer of StemGNN to evaluate whether the prior knowledge on the subgroups contributes to accuracy improvement. To this end, we construct the additional clustering layer using Dynamic Time Warping (DTW) algorithm which finds the alignments between a pair of time series.

Clustering S&P 500 stock series of multiple observations (time steps) requires to balance between clustering quality and the number of observations in each clusters [21]. DTW clustering algorithm balances a pair of time series in such a way to minimize the distances between two series, which allows to compare different observations (time steps) of time series. In addition, DTW technique is superior over Euclidean-base methodologies as it compares a pair of series at any different time steps making the approach relatively robust in distortion and deformation [11]. The intuition of DTW begins with constructing local distance matrix C (local cost function) where C is distance between any time steps of a pair of stock series m and n. Similarly, the local distance matrix of more than three stock series (x variables) can be constructed by measuring the local distances for the x variables simultaneously.

$$C(i,j) = \sqrt{(m_i x - n_j x)^2} \tag{1}$$

Finally, DTW constructs a global cost matrix by which the algorithm finds the optimal path in such a way to minimizes the sum of the distances for a pair of time series. To determine the optimal number of clusters for S&P 500 stock series, we pair DTW with hierarchical clustering method to measure distances and alignments of 492 stock series as shown in Figure4.

For our S&P500 stock series, implementation of clustering results in three clusters. As Shown in Figure5, each group sets of stock series share similar characteristics of price behaviors from one stock to another within the same cluster. For instance, 39 stocks in cluster

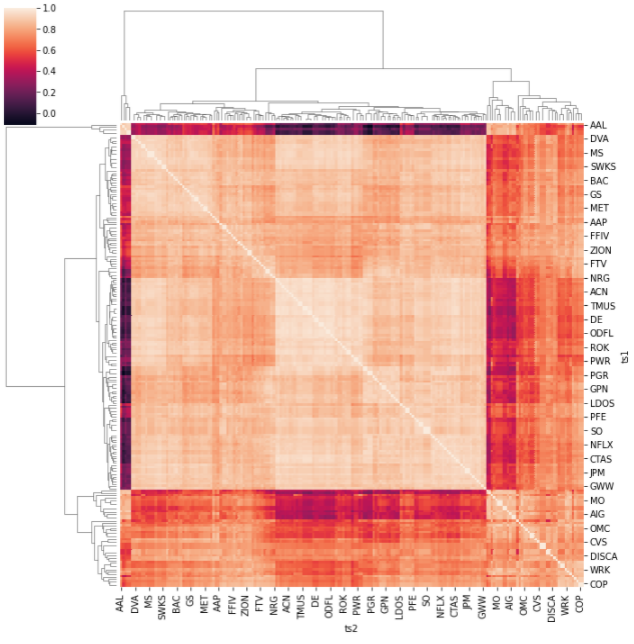

Figure 4: DTW Similarity matrix with a dendrogram imposed

Table 1: Results of the experiments on test set

Architecture	Evaluation Metric		
	MAPE	MAE	RMSE
StemGNN	65.24%	2.40	3.22
StemGNN + Transformers	50.89%	2.02	3.19
StemGNN + Transformers + Clustering (Ours)	**47.58%**	**1.94**	**3.05**

1 (Company: ExxonMobil, Kellogg's, Ford, General Electric, AIG) index appear to be loosing momentum year over year. 262 stocks in cluster 2 (Company: Accenture, American Express, Costco, Home Depot, Intel) exhibit potentials for growth after reversal from the short-term bearish trend during COVID-19 crisis. 191 stocks in cluster 3 (Company: eBay, FedEx, Facebook, Expedia, IBM) show stable stock price performance over the past five years except. In our study, when those clustered stocks series are modelled jointly, it has the effects of creating highly correlated covariates in training transformer + StemGNN networks, which our study assume to be a leading factor to improve forecast accuracy.

4 EXPERIMENT

4.1 Datasets

Our study is based on S&P 500 listed stocks for 5-year past prices ranging from 30-September 2016 to 30-September 2021. We drop off 15 stocks that have failed to meet the time steps requirements mainly due to newly listed stocks after year 2017. We obtain those

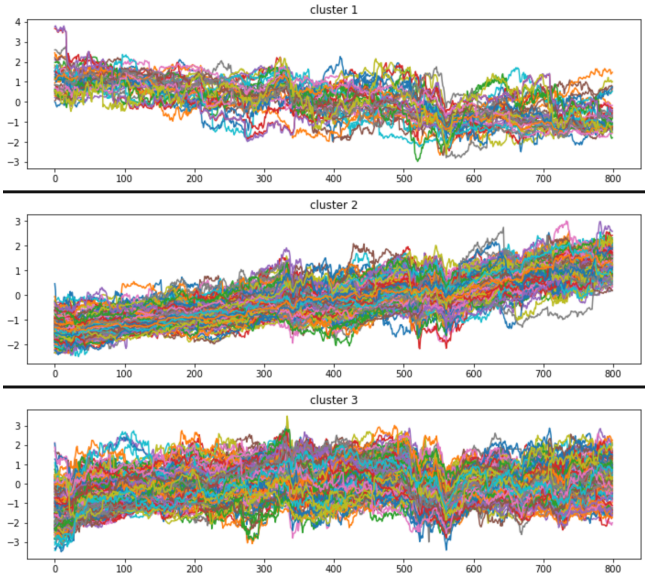

Figure 5: Normalized Adjusted Closing Price of 492 Stock Series in each Clusters

Table 2: Efficiency and memory footprint

Architecture	Time per epoch (s)	# of parameters
StemGNN	28.80	1'826'195
StemGNN + Transformers + Clustering (Ours)	8.22	1'072'331

stock series using Python YahooFinancials package and select adjusted close price as a daily stock price indicator.

4.2 Experimental Settings

In this section, we compare our work to the original StemGNN algorithm and later perform an performance study to further check if our added components really had a role in the model improvement. We setup 3 settings to evaluate the improvement in performance: **(1)** First we run our baseline using the original StemGNN setup. **(2)** Next, we switched out the GRU module in latent correlation layer with a simple Transformer module. We also argue that the our temporal data might be relatively small for the number of stocks, and we hypothesize that training with a smaller subset might be beneficial for the performance. **(3)** Therefore, we perform a dynamic time warping clustering on our stocks, and given the cluster prior information, we will train on each cluster of stocks independently. In total we have clustered all our cohort series to 3 groups, we will provide the cluster analysis in the next section. All of our result will based on the metric MAPE, MAE and RMSE of the test set. Note that for our 3^{rd} setting, we compute the micro average of the individual stock result from each cluster for fair evaluation.

From our dataset we split 70%, 20% and 10% of the samples as training set , validation set and testing set in chronological order, respectively. During the training phase, we train on the first 871 days and validate the performance with the next 249 days. The rest 10% of the data is regarded as test set and will be evaluated by the best performing model out of 5 runs. For our experiment, we use sliding window size as 12, forecasting step as 3 and batch size as 8. In our Transformer module, we have a linear layer with 16 hidden dimensions before performing positional encoding on the input. We set the hidden size of our 8-layers Transformer module as 8 and only utilize 1 head. We also have a linear layer to embed the Transformer output to the number of stocks. In addition, we normalize each time series with its mean and standard deviation as the input to our model.

We show our experiments result in Table 1. We observe that our StemGNN with cluster information method performed the best for each metric with a large margin from the baseline. We dedicate the improvement to the use of stock behaviour prior information. However, we recognise that the computation cost of clustering is subject to the size of the data and may not be as efficient to large scale data. In addition, we also observe slight improvement in the RMSE by replacing Transformer with GRU. We showed in Table 2 that training with Transformer also greatly reduce time per epoch and the number of learnable hyperparameters. We also demonstrate the prediction accuracy of some examples from Cluster 1 in Figure 7. We noticed that the model is able to predict accurately for stocks with stable trend or volatile trend for different range of price values within Cluster 1.

4.3 Hyperparameter Tuning

After we replaced the GRU module with a transformer, we took advantage of the huge speed up in training time and conducted a series of hyperparameter tuning experiments using grid search and found the optimal set of hyperparameters. As compared with the baseline setup, we reduced the batch size from 32 to 8 and also reduced the time steps fed to the Latent Correlation Layer from 12 to 4. We showed in the following subsections the improvements made from each modification and analysed the results.

4.3.1 Batch size. We tested our model on different batch sizes and the result was shown in Table 3. According to the result, mAPE were significantly reduced as we used a lower batch size than the original paper, and we believed this was caused by the difference between the dataset we used and the one used in the original work.

Originally, the hyperparameters were used to train on the PeMS07 dataset [5], which consisted of freeway performance measurements from 228 sensors, and each had recorded the traffic data of 12673 time points. However, our stock data only had 1345 time points for each of the 492 stocks, which was about 10 times smaller than the original.

Furthermore, unlike traffic data which was highly predictable due to the high correlation of its data between subsequent time steps, stock data was more unpredictable, as many external and unseen factor could affect the outcome of the next time step.

Therefore, we believed the improvement from reducing the batch size could be reasoned in 2 ways. First, reducing the batch size caused the weights update to happen between a shorter interval of training data samples, so the weights were updated more rapidly

Table 3: Results of different batch size

Batch size	MAPE
32	69.3%
4	**46.6%**

under the same number of epochs, and the model was less likely to be underfitted when our smaller dataset was used.

Second, as our dataset was inherently noisier than the original one, the training was more unstable as there existed more sharp local minima on the loss surface. As a large batch size tended to converge to sharp minimizer more easily as explained by [10], the model would not generalize well when tested on the test set even when it could achieve a low training loss, and reducing the batch size could mitigate the effect.

4.3.2 Input size in Latent Correlation Layer. During our initial runs of experiment, we noticed that the mAPE on the validation set converged to values in a large range and did not decrease further during the training. For example, we recorded an mAPE above 80% and also below 50% at the end of the training using the exact setup.

We further conducted experiments to visualize the correlation maps produced by the Latent Correlation Layer and the results were shown in Figure 6. High value lines were found in the correlation maps, indicating the model was using a few predictor stocks to predict all the other stocks. We hypothesized that there existed a few important stocks in our dataset that could represent other stocks well, therefore were selected by our model for stock price forecast. However, we further discovered the positions of the high value lines were different for all the correlation maps, showing that the predictor stocks vary between runs.

We believed training instability occurred when the model chose a different set of predictor stocks each time, as its model performance could fluctuate depending on the quality of the stocks it chose, and only a number of stock sets could generalize well in the test set.

Given the analysis, we proposed to further limit the number of most recent time steps the Latent Correlation Layer could see from 12 to 4 to minimize the influence of possible noise from the earlier time steps.

The result was shown in Table 6. As expected, the variance of mAPE calculated over 5 runs was significanly reduced when only more recent time steps were fed to the layer. Although no obvious mAPE improvements were observed, we chose to stay with this setup as our model could produce consistent results between runs and it significantly reduced the runs for us to get a better model base on the validation result.

4.4 Performance Analysis

To investigate performance of our proposing StemGNN networks which are modified with Dynamic Time Warping and Transformer layers, we analyze performance on test data across each of the clusters as shown in Table 5. From our investigation, we observe the significant decrease of MAE and RMSE in Cluster=1 which has 39 stock series. Specifically, RMSE for the Cluster=1 is reduced 4 times against the Cluster=3 whose RMSE is the highest with 3.33.

Table 4: Monthly Price Volatility of Clusters

Cluster	Size	Price volatility(month)
Cluster1	39	18.60
Cluster2	262	74.39
Cluster3	191	43.97

Table 5: Results on each cluster*

		Evaluation Metric		
Cluster	Size	MAPE	MAE	RMSE
Cluster 1	39	50.7%	0.41	0.79
Cluster 2	262	39.6%	2.05	3.20
Cluster 3	191	57.9%	2.12	3.33

* The results were evaluated on each cluster's own test set.

Table 6: Standard Deviation of MAPE over 5 Runs

Time steps in latent correlation layer	standard deviation
12	14.0
4	4.6

We conduct additional analysis to investigate significant error reductions in Cluster=1 by our proposing networks compared to the other groups. To this end, we calculate a monthly price volatility for individual stock series and compute the intra-cluster volatility by using average. The monthly price volatility of stock series is obtained with standard deviation (σ) of a sequence of stock prices for time horizons T where T is $252/12 = 21$ trading days.

$$\sigma_{monthly} = \sigma_{yearly}\sqrt{T} \tag{2}$$

In our experiment, the price volatility is lowest in Cluster=1 which has the lowest MAE and RMSE. We also compute the monthly price volatility in other clusters as shown in Table4. The volatility formula we used is to measure fluctuation of stock prices over a period of time. The lower the value, the lesser fluctuation of stock series. This allows our modified StemGNN to have training on the stable time series patterns which results in lowest MAE and RMSE especially for Cluster=1 whose stock prices are relatively less volatile as compared to other stock series.

5 CONCLUSION

In this work we tackled the very challenging problem of Stock Price Prediction, using StemGNN, an architecture that established as the state-of-the-art in a range of benchmarks. We further improved it using a well-known clustering technique to avoid very uncorrelated stocks to interact with one another, and substituted a GRU-based encoding of the series with a Transformer architecture to leverage self attention mechanism and parallelization on GPU. Our experiments confirmed our intuitions from both effectiveness and efficiency perspectives. We achieved the best performance with

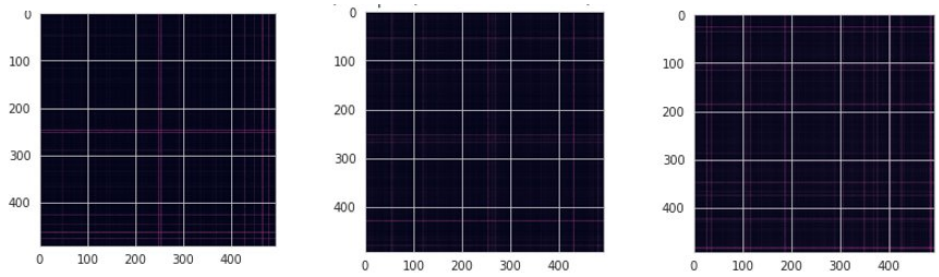

Figure 6: Correlation maps predicted in 3 separate runs using the same setup.

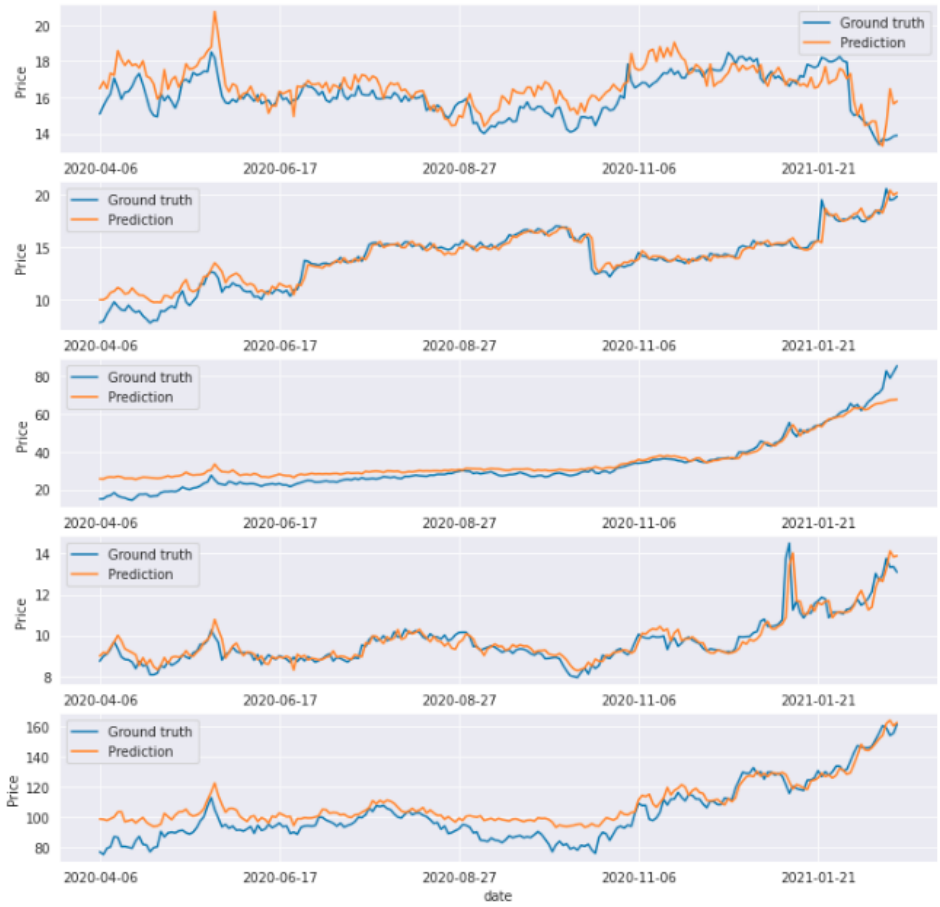

Figure 7: Stock trend examples from Cluster 1 with its prediction.

our StemGNN with clustering approach, we argue that the low price volatility in Cluster 1 has contributed to the improvement. Our modification of the StemGNN architecture also reduced the training time and number of parameters greatly. At last, we noticed that reducing the input time steps of latent correlation layer from 12 to 4 leads to a more stable training performance for each run.

REFERENCES

[1] 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (Jul 2018). https://doi.org/10.24963/ijcai.2018/505

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2015).

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. arXiv:1312.6203 [cs.LG]

[4] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2021. Spectral

Temporal Graph Neural Network for Multivariate Time-series Forecasting. *CoRR* abs/2103.07719 (2021). arXiv:2103.07719 https://arxiv.org/abs/2103.07719

[5] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[6] Thomas G. Fischer and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* 270 (2018), 654–669.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[8] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization. *CoRR* abs/2003.11080 (2020). arXiv:2003.11080 https://arxiv.org/abs/2003.11080

[9] Huseyin Ince and Theodore Trafalis. 2008. Trafalis, T.: Short Term Forecasting with Support Vector Machines and Application to Stock Price Prediction. International Journal of General Systems 37(6), 677-687. *International Journal of General Systems - INT J GEN SYSTEM* 37 (12 2008), 677–687. https://doi.org/10.1080/03081070601068595

[10] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (2016).

[11] Seulbi Lee, Jaehoon Kim, Jongyeon Hwang, EunJi Lee, Kyoung-Jin Lee, Jeongkyu Oh, Jungsu Park, and Tae-Young Heo. 2020. Clustering of Time Series Water Quality Data Using Dynamic Time Warping: A Case Study from the Bukhan River Water Quality Monitoring Network. *Water* 12, 9 (2020), 2411.

[12] Shuanglong Liu, Chao Zhang, and Jinwen Ma. 2017. CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets. 198–206. https://doi.org/10.1007/978-3-319-70096-0_21

[13] Carolyn Reinhart. 2020. The normalized distance Laplacian. arXiv:1903.04575 [math.CO]

[14] David I Shuman, Pierre Vandergheynst, and Pascal Frossard. 2011. Chebyshev polynomial approximation for distributed signal processing. *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)* (Jun 2011). https://doi.org/10.1109/dcoss.2011.5982158

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[16] Xiaohua Wang, P.K.H. Phua, and Weidong Lin. 2003. Stock market prediction using neural networks: Does trading volume help in short-term prediction?. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 4. 2438–2442 vol.4. https://doi.org/10.1109/IJCNN.2003.1223946

[17] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. arXiv:1801.07455 [cs.CV]

[18] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2020. A Transformer-based Framework for Multivariate Time Series Representation Learning. *CoRR* abs/2010.02803 (2020). arXiv:2010.02803 https://arxiv.org/abs/2010.02803

[19] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1409–1416.

[20] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2019. Semantics-aware BERT for Language Understanding. *CoRR* abs/1909.02209 (2019). arXiv:1909.02209 http://arxiv.org/abs/1909.02209

[21] Xiaodan Zhu, Anh Ninh, Hui Zhao, and Zhenming Liu. 2021. Demand Forecasting with Supply-Chain Information and machine learning: Evidence in the Pharmaceutical Industry. *Production and Operations Management* (2021).

A APPENDIX

A.1 Clustering

Composition of Stocks by Clusters and by Sectors				
Sector	Clust 1	Clust2	Clust3	Total
Consumer	8%	54%	37%	87
Energy	14%	29%	57%	14
Financials	5%	46%	49%	57
Healthcare	6%	54%	40%	80
Industrials	3%	45%	53%	40
Information Technology	8%	63%	28%	71
Materials	10%	51%	39%	80
Real Estate	13%	57%	30%	46

Table 7: Clustered S&P 500 stocks by sectors analysis