Temporal Graph Neural Networks for Stock Return Prediction

Andreas Varvarigos
Department of Electrical and Computer Engineering
Yale University
New Haven, CT 06520
`andreas.varvarigos@yale.edu`

1 Introduction

Predicting stock returns is a fundamental challenge in financial modeling with critical implications for investment strategies, portfolio management, and risk assessment. Traditional time-series models, such as Long Short-Term Memory (LSTM), effectively capture temporal dependencies but fail to incorporate interdependency relationships between stocks. This limitation often results in suboptimal predictions, as the dynamic and interconnected nature of financial markets necessitates models capable of simultaneously learning both temporal and spatial dependencies.

We propose a graph-based approach using Graph Neural Networks (GNNs) to explicitly model inter-stock relationships, leveraging the S&P 500 dataset. The graph nodes represent stocks, with features derived from historical returns, while edges capture inter-stock correlations or learnable relationships. Our methodology explores three graph construction approaches: sector-independent graphs, fully connected graphs, and hypergraphs with weak inter-sector connections. We compare Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) for inter-stock relationship modeling, combined with either a window-based method, where we only consider previous return values within a fixed window and discard the rest, or an LSTM-based temporal aggregation to incorporate the entire historical information. The proposed framework incorporates a shallow Multi-Layer Perceptron (MLP) to process the GNN-generated embeddings, ultimately predicting stock returns.

Extensive experiments demonstrate that incorporating temporal-spatial modeling and inter-sector relationships significantly improves predictive accuracy. Notably, fully connected graphs emerged as the best-performing graph construction approach, highlighting the importance of modeling complex inter-sector dependencies. Furthermore, the temporal GNN framework, leveraging learnable correlations, achieved the best results, outperforming both window-based GNNs and LSTM-only baseline. This study shows that combining temporal and spatial information through GNNs leads to superior predictive performance in stock return prediction.

To ensure reproducibility of the experiments and facilitate further research, all code, scripts, and instructions for running the experiments are available in the following GitHub repository: https://github.com/varvarigos/stock_return_pred_using_temporal_gnns/tree/main.

2 Related Work

Traditional time-series models, such as LSTMs [6] and Autoregressive integrated moving average (ARIMA) [1], have been widely used in stock price prediction [3, 8]. These models excel at capturing temporal dependencies within sequential data, but they fail to account for the complex spatial relationships between stocks, such as interdependencies across sectors or companies.

GNNs provide a powerful alternative by leveraging graph structures to model these relationships. GNNs have been explored for stock movement prediction by incorporating sector and geographical proximity into the graph structure [4]. However, their approach primarily focused on learning inter-company relation embeddings rather than directly predicting stock returns.

Recent developments in temporal graph models have further expanded the applicability of GNNs. Temporal Graph Convolutional Network (T-GCN) [9] combines GCNs with Gated Recurrent Units (GRUs) [2], enabling the modeling of both spatial and temporal dependencies. Subsequent models, such as A3T-GCN [7] and AST-GCN [5], incorporate attention mechanisms and external attributes to enhance performance in dynamic systems like traffic forecasting. These advancements highlight the potential of temporal GNNs to capture long-term dependencies and adapt to evolving systems. However, the authors in [7] and [5] focuses on non-financial domains, namely traffic flow forecasting, leaving a gap in the application of temporal GNNs to stock return prediction. My work extends this type of analysis by adapting such models to the financial domain, incorporating learnable edge weights to better capture nuanced inter-stock relationships and leveraging temporal GNNs to integrate both spatial and temporal information and predict stock returns. By evaluating different graph construction approaches and exploring the trade-offs between window-based and temporal aggregation methods, this study demonstrates the utility of GNNs for robust stock return prediction.

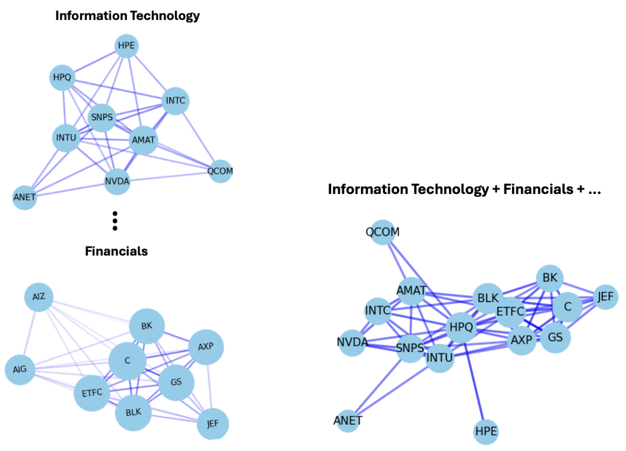
3 Method

I model the stock market as a graph $G = (V, E)$, where V represents a set of stocks, and the set of edges E represents relationships between these stocks. The graph construction approaches are detailed below:

1. **Sector-Independent Graphs:** Separate graphs are constructed for each stock market sector (Figure 1a), assuming minimal, and thus, negligible inter-sector interactions, where stocks in one sector do not influence those in another. In other words, this representation ignores the influence one sector may have on another.
2. **Fully Connected Graphs:** A single graph includes all sectors (Figure 1b), connecting every stock to all other correlated stocks. This setup assumes inter-sector interactions are significant, allowing the GNN to capture both intra- and inter-sector relationships for a comprehensive market view. The name fully connected is used expansively in the sense that I consider the correlations between all pairs of stocks, but in fact the graph may not be fully connected.
3. **Hypergraphs with Weak Inter-Sector Connections:** A "hypergraph" is constructed where companies are clustered into sector-specific subgraphs, each sector represented by a dedicated node. All stocks within a sector are connected to their respective sector node, while stocks from different sectors are not directly connected. Sector nodes themselves will be connected (Figure 1c), representing weaker but existing inter-sector relationships, balancing computational efficiency with the ability to capture cross-sector correlations.

The node features for the graph are derived from historical stock returns. These features are either constrained to a specific time window—weekly (7 days), monthly (30 days), or half-year (180 days)—or aggregated over the entire historical data available (limited to up to 1 year to avoid over-smoothing). Aggregation over the full history uses a shallow LSTM with one hidden layer to condense the variable-length time-series feature vector into a fixed-size embedding. Edge weights are defined either by simple correlations or by learnable mappings. The learnable mappings are produced using a shallow MLP with three linear layers, each followed by ReLU activation functions. These edge weights capture the relationships between stocks and contribute to the GNN's ability to propagate relevant information across the graph.

To process the graph, either a GCN or a GAT model is employed. Both architectures consist of two hidden layers and produce a 20-dimensional embedding for each stock. The embedding is then passed through another shallow MLP (three linear layers, with ReLU activation after each except the last layer, as the stock return prediction can be negative). The final output is a scalar value representing the predicted next-day return for each stock. The architecture is designed to be inductive, as each sample corresponds to a unique graph. This is essential due to the dynamic nature of the dataset: the inclusion of temporal models means that predictions for later dates incorporate more historical data, and missing data (e.g., for stocks that were not consistently active in the market) lead to variable graph structures. To establish a baseline for comparison, I also trained a standalone LSTM model without the use of GNNs. This model processes the time-series data directly to predict the next-day stock return without taking into consideration any stock interdependencies. This comparison evaluates the effectiveness of incorporating GNNs to model inter-stock relationships alongside temporal dependencies.



(a) Independent Graphs for Each Sector.

(b) Fully Connected Graph Companies.

(c) Independent Sector Graphs with Weak Inter-Sector Connections.

Figure 1: Illustration of three approaches for constructing stock relationship graphs: (a) Separate graphs for each sector; (b) Single graph for all sectors encompassing the stocks of all sectors; (c) Independent sector graphs with weak inter-sector connections. The correlation threshold is set at 0.6 (links with correlation below this threshold are ignored in all cases (a), (b), (c)), with stronger correlations indicated by higher edge opacity, and larger node sizes representing greater market influence (i.e., higher sum of correlations). Note that the stocks of only two sectors (IT and FIN) are displayed for easy visualisation purposes.

4 Experiments

4.1 Dataset

The S&P 500 dataset, used in the experiments, consists of daily stock returns for 505 companies over 1006 trading days. These companies are categorized into 11 sectors, providing a sectoral structure for graph construction. Edges between stocks are defined either by their correlation values or by learnable mappings computed through a shallow MLP. Node features are derived from prior returns, constrained to a chosen window size for the window-based method, or aggregated across the entire historical data up to 360 days for the temporal method. Missing values in the dataset are handled by discarding samples where the number of missing entries exceeds the number of valid ones within the relevant window, ensuring data integrity.

The graph construction approaches greatly impact computational efficiency and predictive capabilities. Figure 2 and Table 3 of Appendix summarize the key characteristics of the graphs used in each approach in terms of node degree distribution. Approach 1 (sector-independent graph) is the simplest design with separate graphs for each sector, resulting in a median degree of 65 and 13,301 edges. However, it excludes inter-sector relationships, limiting its ability to capture cross-sector dynamics. Approach 2 (fully connected graph) with a median degree of 505 and 127,187 edges, is the most expressive but computationally expensive. Approach 3 (hypergraph with weak inter-sector connections) balances complexity and expressiveness by incorporating inter-sector connections via a hypergraph structure. With a median degree of 66 and 13,861 edges, it maintains the computational efficiency of Approach 1 while as it uses a similar number of edges while capturing meaningful cross-sector influences. While Approach 2 offers the potential for greater predictive insights, experiments assess whether its complexity justifies the trade-offs compared to the more efficient Approaches 1 and 3.

4.2 Training Details

The training process employs the Adam optimizer along with a `ReduceLROnPlateau` learning rate scheduler, which reduces the learning rate by half whenever the validation loss fails to improve for 4 consecutive epochs. Training is conducted over a maximum of 40 epochs, with early stopping implemented to terminate training if validation loss does not improve for 11 epochs, thereby mitigating overfitting. The tuned hyperparameter values are detailed in Table 3 in the Appendix.

For reproducibility, random seeds for both the `PyTorch` and `random` libraries are set to 42. Prior to training, the dataset is preprocessed by scaling stock returns to fall within the range [-1, 1]. This scaling method was determined to yield better performance compared to standard normalization methods such as scaling to zero mean and unit variance. Finally, the dataset is split into training, validation, and test sets according to 80/10/10 ratio.

4.3 Results

I will now present the results, starting with ablation studies for each method—window-based GNNs and temporal GNNs using LSTM. Following this, I will compare the best-performing configurations of both methods against each other and the LSTM-only baseline. The performance is evaluated using two metrics: the Mean Squared Error (MSE) loss and the Mean Directional Accuracy (below referred to as Sign Accuracy), which measures the % of predictions where the model correctly determines the sign of the stock returns, effectively capturing periods of growth and decline.

4.3.1 Window-based GNNs

In Table 1, the results for the GCN architecture are presented across various window sizes, weight types, and graph construction approaches. The findings reveal that graph construction approach 2, the fully connected graph, consistently outperforms the others in terms of test metrics (both in MSE and Sign Accuracy), regardless of the weight type or window size. This highlights the intricate and complex interconnections between stocks, even across different sectors, showcasing the importance of modeling comprehensive inter-sector relationships for enhanced predictive power. When comparing window sizes, a one-month window emerges as the best performer in most configurations, while the one-week window is the least effective. Using too many past returns (e.g., six months) tends to dilute the importance of recent, more relevant data, leading to underperformance. On the other hand, a smaller window size (e.g., weekly) fails to capture sufficient information, underscoring the necessity of a balanced window that incorporates both short-term trends and broader market dynamics. Additionally, using learnable correlations consistently outperforms direct correlations, as the model effectively extracts nuanced relationships that are not immediately apparent from raw correlation values. The best-performing window-based GCN configuration utilizes a monthly window, graph construction approach 2, and learnable correlations, and achieves a test loss and Sign Accuracy of 0.0329 and 81.26% correspondingly, demonstrating its robustness in understanding market trends to predict whether a stock is growing or shrinking.

Graph Approach	**Weight Type**	**Window** (Days)	**Train** (MSE, Sign Acc.)	**Val** (MSE, Sign Acc.)	**Test** (MSE, Sign Acc.)
1	Correlation	7	$(0.0446 \pm 0.0, 76.77 \pm 0.0)$	$(0.1024 \pm 0.0, 74.8 \pm 0.0)$	$(0.0982 \pm 0.0, 70.95 \pm 0.0)$
		30	$(0.0687 \pm 0.0, 66.53 \pm 0.0)$	$(0.0599 \pm 0.0, 66.43 \pm 0.0))$	$(0.0553 \pm 0.0, 66.21 \pm 0.0)$
		180	$(0.0439 \pm 0.0, 76.47 \pm 0.0)$	$(0.0474 \pm 0.0, 76.12 \pm 0.0)$	$(0.0478 \pm 0.0, 75.91 \pm 0.0)$
	Learnable Corr.	7	$(0.0333 \pm 0.0, 80.61 \pm 0.0)$	$(0.0363 \pm 0.0, 80.93 \pm 0.0)$	$(0.0456 \pm 0.0, 78.23 \pm 0.0)$
		30	$(0.031 \pm 0.0, 81.75 \pm 0.0)$	$(0.037 \pm 0.0, 80.95 \pm 0.0)$	**(0.0329 ± 0.0, 81.28 ± 0.0)**
		180	$(0.0303 \pm 0.0, 82.38 \pm 0.0)$	$(0.0309 \pm 0.0, 82.62 \pm 0.0)$	$(0.0319 \pm 0.0, 81.11 \pm 0.0)$
2	Correlation	7	$(0.092 \pm 0.0, 35.69 \pm 0.0)$	$(0.078 \pm 0.0, 37.48 \pm 0.0)$	$(0.800 \pm 0.030, 72 \pm 1)$
		30	$(0.085 \pm 0.0, 66.53 \pm 0.0)$	$(0.067 \pm 0.0, 66.22 \pm 0.0)$	$(0.06 \pm 0.0, 66.21 \pm 0.0)$
		180	$(0.0464 \pm 0.0, 75.1 \pm 0.0)$	$(0.0555 \pm 0.0, 74.37 \pm 0.0)$	$(0.0578 \pm 0.0, 71.58 \pm 0.0)$
	Learnable Corr.	7	$(0.0333 \pm 0.0, 80.6 \pm 0.0)$	$(0.0363 \pm 0.0, 80.95 \pm 0.0)$	$(0.0458 \pm 0.0, 78.11 \pm 0.0)$
		30	$(0.031 \pm 0.0, 81.76 \pm 0.0)$	$(0.037 \pm 0.0, 80.95 \pm 0.0)$	$\boxed{\mathbf{(0.0329 \pm 0.0, 81.26 \pm 0.0)}}$
		180	$(0.0303 \pm 0.0, 82.39 \pm 0.0)$	$(0.0309 \pm 0.0, 82.62 \pm 0.0)$	$(0.0336 \pm 0.0, 81.1 \pm 0.0)$
3	Correlation	7	$(0.0478 \pm 0.0, 76.22 \pm 0.0)$	$(0.099 \pm 0.0, 74.35 \pm 0.0)$	$(0.1055 \pm 0.0, 69.88 \pm 0.0)$
		30	$(0.1279 \pm 0.0, 66.38 \pm 0.0)$	$(0.083 \pm 0.0, 66.35 \pm 0.0)$	$(0.056 \pm 0.0, 66.03 \pm 0.0)$
		180	$(0.0439 \pm 0.0, 76.51 \pm 0.0)$	$(0.044 \pm 0.0, 75.92 \pm 0.0)$	$(0.0487 \pm 0.0, 75.63 \pm 0.0)$
	Learnable Corr.	7	$(0.0337 \pm 0.0, 80.4 \pm 0.0)$	$(0.0368 \pm 0.0, 80.64 \pm 0.0)$	$(0.0464 \pm 0.0, 77.9 \pm 0.0)$
		30	$(0.0314 \pm 0.0, 81.58 \pm 0.0)$	$(0.0377 \pm 0.0, 80.68 \pm 0.0)$	**(0.0334 ± 0.0, 81.06 ± 0.0)**
		180	$(0.0306 \pm 0.0, 82.23 \pm 0.0)$	$(0.0311 \pm 0.0, 82.42 \pm 0.0)$	$(0.0322 \pm 0.0, 80.86 \pm 0.0)$

Table 1: Mean $\pm$ standard deviation of (MSE, Sign Accuracy) for Train, Validation, and Test sets using the **window-based GCN** method, computed by running each experiment three times. The best test metric for each graph approach (across varying windows and weight types) is highlighted in bold, and the overall best-performing approach is enclosed in a box. Note that standard deviations appearing as 0 are due to rounding to the 6th digit.

When adding attention, the results in Table 4 of Appendix show a consistent performance degradation compared to the corresponding configurations without attention. This decline is likely due to over-smoothing caused by the influence of more distant neighbors, which is further amplified by the density of the graphs. The model is also shown to be more prone to overfitting, which can be attributed to the increase in parameters due to the attention weights. Despite this, the same trends observed in the GCN architecture hold true: the best performance with attention is achieved using a monthly window, graph construction approach 2, and learnable correlations. Overall, the window-based GCN remains the best-performing windowed-GNN approach.

4.3.2 Temporal GNNs

We now consider temporal GNNs and observe from Table 2 that similar principles apply as with the window-based GNNs. However, for the majority of configurations, temporal GNNs outperform their corresponding window-based counterparts. This highlights the ability of the LSTM to effectively capture significant historical return patterns that the window-based approaches could not fully leverage. The best-performing configuration uses a Temporal GCN with learnable correlation weights on fully connected graphs, achieving a test Sign Accuracy of 83.1% and a test MSE loss of 0.0285. In comparison, the best-performing window-based GNN achieved a test Sign Accuracy of 81.26% and a test MSE loss of 0.0329, demonstrating the added benefit of incorporating temporal dependencies.

Graph Approach	**Weight Type**	**Attention**	**Train** (MSE, Sign Acc.)	**Val** (MSE, Sign Acc.)	**Test** (MSE, Sign Acc.)
1	Correlation	No	$(0.0562 \pm 0.0, 67.97 \pm 0.0)$	$(0.0997 \pm 0.0, 62.26 \pm 0.0)$	$(0.0988 \pm 0.0, 72.0 \pm 0.0)$
		Yes	$(0.0368 \pm 3e-06, 78.85 \pm 0.034335)$	$(0.0423 \pm 5.6e-05, 77.3 \pm 0.139544)$	$(0.0667 \pm 0.000933, 74.82 \pm 0.331425)$
	Learnable Corr.	No	$(0.0301 \pm 2e-06, 81.48 \pm 0.00119)$	$(0.0356 \pm 8e-06, 80.2 \pm 0.005343)$	**(0.0631 ± 0.0009, 77.29 ± 0.1702)**
		Yes	$(0.0367 \pm 2e-06, 78.71 \pm 0.024217)$	$(0.0425 \pm 4.8e-05, 77.27 \pm 0.1211)$	$(0.0667 \pm 0.0009, 74.79 \pm 0.3112)$
2	Correlation	No	$(0.0565 \pm 0.0, 67.97 \pm 0.0)$	$(0.0495 \pm 0.0, 60.99 \pm 0.0)$	$(0.0522 \pm 0.0, 71.47 \pm 0.0)$
		Yes	$(0.0379 \pm 2e-06, 78.59 \pm 0.047196)$	$(0.0437 \pm 5.7e-05, 77.27 \pm 0.182799)$	$(0.0691 \pm 0.00103, 74.47 \pm 0.406811)$
	Learnable Corr.	No	$(0.033 \pm 0.0, 81.50 \pm 0.0)$	$(0.034 \pm 0.0, 81.20 \pm 0.0)$	$\boxed{\mathbf{(0.0285 \pm 0.0, 83.1 \pm 0.0)}}$
		Yes	$(0.036 \pm 0.0, 80.58 \pm 4.2e-6)$	$(0.0376 \pm 1e-06, 80.47 \pm 2.6e-05)$	$(0.0455 \pm 1e-06, 79.37 \pm 8.4e-05)$
3	Correlation	No	$(0.0562 \pm 0.0, 67.8 \pm 0.0)$	$(0.996 \pm 0.0, 62.07 \pm 0.0)$	$(0.1001 \pm 0.0, 71.94 \pm 0.0)$
		Yes	$(0.0358 \pm 0.0, 79.79 \pm 0.00144)$	$(0.0376 \pm 0.0, 79.54 \pm 0.00044)$	$(0.0458 \pm 0.0, 78.48 \pm 0.00041)$
	Learnable Corr.	No	$(0.0316 \pm 0.0, 81.56 \pm 0.0)$	$(0.0341 \pm 0.0, 80.57 \pm 1.6e-05)$	**(0.0421 ± 0.0, 80.06 ± 5e-05)**
		Yes	$(0.0361 \pm 0.0, 79.96 \pm 2.8e-05)$	$(0.0379 \pm 0.0, 79.64 \pm 6.5e-05)$	$(0.0462 \pm 0.0, 78.44 \pm 0.00102)$

Table 2: Mean $\pm$ standard deviation of MSE (Train, Validation, Test) and Sign Accuracy (Train, Validation, Test) using **temporal GNN** approach with LSTM, running each experiment three times. The best test metric for each graph approach (across varying weight types with/without attention) is highlighted in bold, and the overall best-performing approach is enclosed in a box.

4.3.3 Comparison with best-performing architectures and baseline

As shown in Table 5 of Appendix, the baseline LSTM-only architecture achieves a test Sign Accuracy of 81.42%, slightly below the best window-based GNN model. However, it records a lower test MSE of 0.0293 compared to 0.0329 for the window-based GNN. Despite these competitive results, both approaches are surpassed by the best temporal GNN model, which outperforms them in both test Sign Accuracy and test MSE (see Figure 4 of Appendix). This demonstrates that while using either temporal information or inter-sector relationships independently yields strong performance for stock return prediction, combining these methods provides additional predictive power.

5 Conclusion

This work examines the performance of GNNs for the stock price prediction task, highlighting the potential of leveraging both temporal and relational information through temporal GNNs. By incorporating LSTMs, temporal GNNs are shown to handle long-term dependencies more effectively than window-based GNNs, where incorporating excessive historical information (e.g., six months) was found to overshadow the importance of recent data, with a one-month window emerging as the optimal choice. The results also emphasize the significance of inter-sector relationships, as fully connected graphs consistently outperformed other graph construction approaches, indicating how stocks from different sectors contribute meaningfully to predictive performance improvement despite the added computational complexity.

Future work could focus on adapting advanced temporal graph models, such as T-GCN and A3T-GCN, to this task to further enhance predictive capabilities. Exploring architectures like Graphormer, which leverage self-attention mechanisms, could provide additional improvements by capturing nuanced stock relationships. Additionally, investigating alternative edge definitions beyond correlation, such as graphical lasso, may offer new insights into inter-stock dependencies and enhance the versatility of GNN-based approaches for financial modeling.

References

[1] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] Eugene F Fama. Random walks in stock market prices. *Financial Analysts Journal*, 51(1):75–80, 1995.

[4] Fei Feng, Xiangnan He, Jinhui Tang, and Tat-Seng Chua. Temporal relational modeling with graph neural networks for stock prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–354, 2019.

[5] Shengnan Guo, Yu Lin, Nan Feng, Chao Song, and Hongyan Wan. Ast-gcn: Attention-based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4228–4235, 2021.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Yuqing Liang, Liang Wang, Yongheng Ding, Yu Sun, and Xiaokang Song. A3t-gcn: Attention temporal graph convolutional network for fraud detection in e-commerce. *Neurocomputing*, 450:135–146, 2021.

[8] Ruey S Tsay. *Analysis of financial time series*. John Wiley & Sons, 2005.

[9] Ling Zhao, Yuan Song, Hao Deng, Yu Liu, Hong Li, Tie Yi, and Feng Li. T-gcn: A temporal graph convolutional network for traffic prediction. In *IEEE transactions on intelligent transportation systems*, volume 21, pages 3848–3858. IEEE, 2019.

Appendix

Figure 2 illustrates the density of node degrees for the three graph representation approaches, while Figure 3 compares their key metrics such as median, mean, and maximum degrees, as well as the number of edges. These figures are used to analyze the structural differences and complexity of the graph representations.

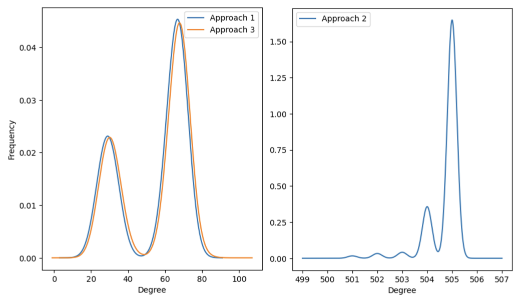


Figure 2: Density of node degrees for the three different graph representation approaches.

Metric	Appr. 1	Appr. 2	Appr. 3
Med Deg.	65	505	66
Mean Deg.	53.68	504.71	54.72
Max Deg.	69	505	80
Min Deg.	25	501	26
# of Edges	13,301	127,187	13,861

Figure 3: Comparison of median, mean, max, min degrees, and number of edges for the three graph representation approaches.

Table 3 lists the hyperparameters used during the training process, including their values and a brief description. It is referenced when discussing the experimental setup and helps clarify the configurations used during training and how to achieve the best-performing models.

Hyperparameter	Value	Description
`thres_corr_edge_drop`	0.6	Dropping edges below correlation threshold.
`epochs`	40	Maximum number of training epochs.
`lr`	1×10^{-3}	Learning rate for the GCN/GAT model.
`lr_mlp`	1×10^{-4}	Learning rate for the post-GNN shallow MLP.
`lr_edge_mlp_mapper`	5×10^{-4}	Learning rate for the edge weight MLP mapper.
`lr_lstm`	1×10^{-4}	Learning rate for the LSTMs.
`ReduceLROnPlateau.patience`	4	# epochs to wait before reducing the lr on val loss plateau.
`ReduceLROnPlateau.factor`	0.5	Factor by which the lr is reduced when val loss plateaus.
`EarlyStopping.patience`	11	# epochs to wait before training stops on val loss decrease.

Table 3: Hyperparameters used in the training process. The description column provides a brief explanation of each hyperparameter. Note that other hyperparameters, such as window size and edge types (learnable vs. plain correlation), are studied in the ablation studies in Section 4.3.

Table 4 presents the mean and standard deviation of MSE and Sign Accuracy for the Train, Validation, and Test sets using the window-based GAT method, over three runs. It was seen that the GAT model consistently underperformed compared to the GCN architecture, likely due to increased sensitivity to over-smoothing and overfitting caused by attention mechanisms in dense graph settings.

Graph Approach	Weight Type	Window (Days)	Train (MSE, Sign Acc.)	Val (MSE, Sign Acc.)	Test (MSE, Sign Acc.)
1	Correlation	7	$(0.0397 \pm 0.0, 77.8 \pm 0.0)$	$(0.0424 \pm 0.0, 78.51 \pm 0.0)$	$(0.0524 \pm 0.0, 75.17 \pm 0.0)$
		30	$(0.0358 \pm 0.0, 79.08 \pm 0.0)$	$(0.0447 \pm 0.0, 77.69 \pm 0.0)$	$(0.0413 \pm 0.0, 77.59 \pm 0.0)$
		180	$(0.0286 \pm 0.0, 82.24 \pm 0.0)$	$(0.0397 \pm 0.0, 78.87 \pm 0.0)$	$(0.042 \pm 0.0, 77.68 \pm 0.0)$
	Learnable Corr.	7	$(0.0397 \pm 0.0, 77.8 \pm 0.0)$	$(0.0424 \pm 0.0, 78.51 \pm 0.0)$	$(0.0524 \pm 0.0, 75.17 \pm 0.0)$
		30	$(0.0358 \pm 0.0, 79.08 \pm 0.0)$	$(0.0447 \pm 0.0, 77.69 \pm 0.0)$	(**0.0413 ± 0.0**, 77.59 ± 0.0)
		180	$(0.0286 \pm 0.0, 82.24 \pm 0.0)$	$(0.0397 \pm 0.0, 78.87 \pm 0.0)$	(0.042 ± 0.0, **77.68 ± 0.0**)
2	Correlation	7	$(0.0436 \pm 0.0, 76.48 \pm 0.0)$	$(0.0461 \pm 0.0, 77.61 \pm 0.0)$	$(0.0565 \pm 0.0, 73.77 \pm 0.0)$
		30	$(0.0322 \pm 0.0, 80.97 \pm 0.0)$	$(0.0439 \pm 0.0, 78.16 \pm 0.0)$	$(0.0408 \pm 0.0, 78.97 \pm 0.0)$
		180	$(0.027 \pm 0.0, 83.75 \pm 0.0)$	$(0.0385 \pm 0.0, 80.19 \pm 0.0)$	$(0.0403 \pm 0.0, 78.92 \pm 0.0)$
	Learnable Corr.	7	$(0.0436 \pm 0.0, 76.48 \pm 0.0)$	$(0.0461 \pm 0.0, 77.61 \pm 0.0)$	$(0.0565 \pm 0.0, 73.77 \pm 0.0)$
		30	$(0.0322 \pm 0.0, 80.97 \pm 0.0)$	$(0.0439 \pm 0.0, 78.16 \pm 0.0)$	$\boxed{(0.0408 \pm 0.0, \mathbf{78.97 \pm 0.0})}$
		180	$(0.027 \pm 0.0, 83.75 \pm 0.0)$	$(0.0385 \pm 0.0, 80.19 \pm 0.0)$	(**0.0403 ± 0.0**, 78.92 ± 0.0)
3	Correlation	7	$(0.0403 \pm 0.0, 77.4 \pm 0.0)$	$(0.043 \pm 0.0, 78.25 \pm 0.0)$	$(0.0534 \pm 0.0, 74.72 \pm 0.0)$
		30	$(0.0366 \pm 0.0, 79.38 \pm 0.0)$	$(0.0465 \pm 0.0, 77.76 \pm 0.0)$	$(0.0426 \pm 0.0, 77.89 \pm 0.0)$
		180	$(0.0274 \pm 0.0, 82.54 \pm 0.0)$	$(0.0401 \pm 0.0, 78.51 \pm 0.0)$	$(0.0431 \pm 0.0, 76.93 \pm 0.0)$
	Learnable Corr.	7	$(0.0403 \pm 0.0, 77.4 \pm 0.0)$	$(0.043 \pm 0.0, 78.25 \pm 0.0)$	$(0.0534 \pm 0.0, 74.72 \pm 0.0)$
		30	$(0.0366 \pm 0.0, 79.38 \pm 0.0)$	$(0.0465 \pm 0.0, 77.76 \pm 0.0)$	**(0.0426 ± 0.0, 77.89 ± 0.0)**
		180	$(0.0274 \pm 0.0, 82.54 \pm 0.0)$	$(0.0401 \pm 0.0, 78.51 \pm 0.0)$	$(0.0431 \pm 0.0, 76.93 \pm 0.0)$

Table 4: Mean $\pm$ standard deviation of (MSE, Sign Accuracy) for Train, Validation, and Test sets using the **window-based GAT** method, computed by running each experiment three times. The best test metric for each graph approach (across varying windows and weight types) is highlighted in bold, and the overall best-performing approach is enclosed in a box. Note that standard deviations appearing as 0 are due to rounding to the 6th digit.

Table 5 shows the performance of the best window-based GNN, temporal GNN, and LSTM-only baseline in terms of MSE and Sign Accuracy. The temporal GNN outperformed both the window-based GNN and LSTM-only baseline, achieving the lowest test MSE and the highest Sign Accuracy.

Model	Test (MSE, Sign Acc.)
Best Window-Based GNN	$(0.0329 \pm 0.0, 81.51 \pm 0.0)$
Best Temporal GNN	**(0.0285 ± 0.0, 83.1 ± 0.0)**
Baseline LSTM-Only	$(0.0293 \pm 0.0, 81.42 \pm 0.0)$

Table 5: Comparison of the **best window-based GNN, temporal GNN, and baseline LSTM-only** approaches. Metrics are presented as mean $\pm$ standard deviation for MSE and Sign Accuracy across Train, Validation, and Test sets.

Figure 4 illustrates the training and validation accuracies for the best-performing method, a Temporal GCN with learnable parameters on fully-connected interdependency graphs. This plot highlights the model's ability to generalize effectively without overfitting.

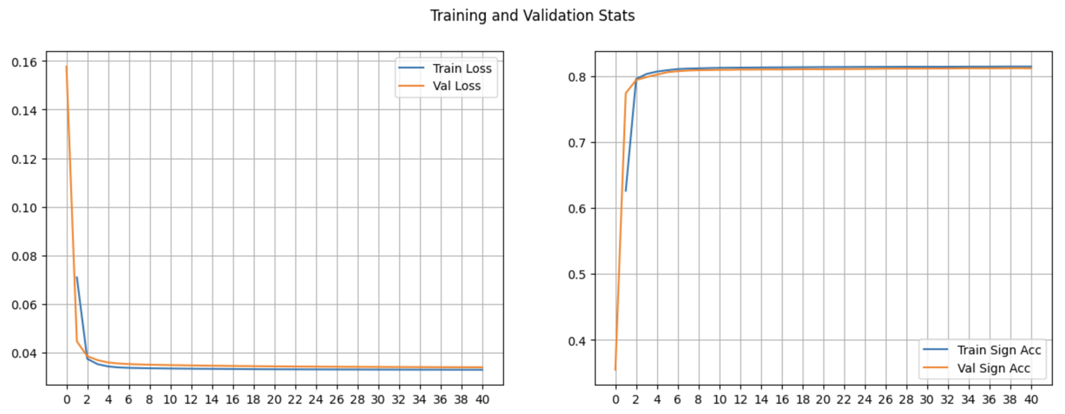


Figure 4: Training and validation accuracies for best performing method - Temporal GCN with learnable parameters on fully-connected interdependency graphs.