A Spatio-Temporal Transformer Model for Node Attribute Prediction in Dynamic Graphs

Namrata Banerji and Tanya Berger-Wolf

The Ohio State University, Columbus OH, USA

Abstract. Accurate multi-step forecasting of node-level attributes on dynamic graphs is critical for applications ranging from financial trust networks to traffic monitoring. Existing spatio-temporal graph neural networks typically assume a static adjacency, and seldom deal with multidimensional timeseries prediction. In this work, we propose an end-to-end *dynamic edge-biased spatio-temporal model* that ingests a multidimensional timeseries of node attributes and a timeseries of adjacency matrices, to predict multiple future steps of node attributes. At each time step, our transformer-based model injects the given adjacency as an adaptable attention bias, allowing the model to focus on relevant neighbors as the graph evolves. We further deploy a masked node/time pretraining objective that primes the encoder to reconstruct missing features, and train with scheduled sampling and a horizon-weighted loss to mitigate compounding error over long horizons. Unlike prior work, our model accommodates dynamic graphs that vary across input samples, enabling forecasting in multi-system settings such as brain networks across different subjects, financial systems in different contexts, or evolving social systems. Empirical results demonstrate that our method outperforms strong STGCN, DCRNN, and MTGNN baselines by 10–20%.

Keywords: Dynamic Graphs, Spatio-Temporal Graph Neural Networks, Transformers, Node Attribute Prediction

1 Introduction

Many real-world systems, ranging from brain connectivity networks to social trust platforms, are naturally represented as dynamic graphs, where the set of edges and node attributes evolve over time. In these settings, the underlying relationships between entities (i.e., the graph structure) change due to external stimuli or internal dynamics. For example, in functional brain networks, edges correspond to time-varying functional connections between brain regions, which reconfigure dynamically in response to cognitive states or external tasks. In Bitcoin-OTC and Alpha trust networks, the trust scores exchanged between users evolve as a result of transactions, leading to changing connectivity over time. Similarly, in dynamic social or biological systems, interactions are not only sparse but also transient, making edge evolution a critical modeling component.

While considerable progress has been made in learning from dynamic graphs, much of this work focuses on node classification or link prediction, often assuming fixed node connectivity or event-stream representations of edge changes. In contrast, predicting future node attributes, such as a node's behavioral signal, risk score, or physiological state, is both important and underexplored, especially in the presence of dynamic edge structures.

In this work, we address the problem of node attribute prediction in fully dynamic graphs, where both node features and edges change over time, but the node set remains fixed. This setting arises naturally in domains where node states are observed over time and influenced by evolving interactions. We propose a novel architecture that combines dynamic graph learning, graph attention, and temporal self-attention to model both short- and long-range spatiotemporal dependencies. We propose two variants of our model, one that learns a time-varying graph structure relevant to the node attribute prediction, and one that uses a given dynamic graph structure with edge biases to predict the node attributes.

To evaluate our method, we consider synthetic and real-world datasets including the LA traffic network and Bitcoin-OTC and Alpha trust networks, where node features represent things like traffic volume at an intersection or average trust ratings given and received. We also compare against strong baselines like STGCN[18] and MTGNN[16], showing that our model achieves superior performance on root mean squared error and mean absolute error metrics. This work is, to the best of our knowledge, the first to explicitly address node attribute prediction using dynamic graphs with fully evolving edge structures.

2 Related Work

Dynamic Graph Representation Learning. A large body of work has been devoted to learning on dynamic graphs, primarily targeting tasks such as link prediction and node classification. Methods such as EvolveGCN [13], TGAT [3], TGN [14], and DyRep [15] model temporal interactions in graphs by evolving either node embeddings or graph parameters over time. However, these methods typically focus on classification or event prediction and do not address the task of predicting continuous-valued node attributes. Furthermore, many prior works model graphs as streams of discrete events (e.g., interactions between node pairs), rather than explicitly modeling evolving graph snapshots with dense temporal node attributes. Event-stream models are flexible but cannot always handle rich node attribute time series directly (e.g., vectors of features at each time step). Our method is better suited for settings where both topology and node attributes evolve continuously and are available at regular intervals.

Time Series Forecasting with GNNs. Several methods have explored forecasting node values in spatiotemporal settings, especially in traffic and sensor networks. STGCN [18], DCRNN [11], and Graph WaveNet [17] operate on static graphs and combine temporal convolution or recurrent modules with graph convolution for short-term forecasting. These models assume fixed connectivity between nodes, making them inapplicable to domains where the underlying net-

work structure evolves. While some extensions like DGCRN [10] and AGCRN [1] incorporate latent or adaptive graphs, they often do not model fully dynamic edge sets or permit per-timestep graph changes.

Learning Graph Structure for Forecasting. MTGNN [16] addresses multivariate time series forecasting by learning graph structure jointly with temporal prediction. However, it assumes a shared graph across the entire sequence (or batch), rather than modeling a dynamic graph that changes at each timestep. Additionally, the node attributes are assumed to be single dimensional, which is suitable for the traffic forecasting use case, but limits its generalizability to real-world dynamic graphs such as social or trust networks with potentially multivariate node features.

Node Attribute Prediction in Dynamic Graphs. Surprisingly few works explicitly address multivariate node attribute prediction in dynamic graphs with changing edge structure. In this setting, both the input and output are time series of node features, and the evolving topology provides contextual relational information. Our work differs from prior approaches in three key ways:

- we predict **continuous, multidimensional** node attributes (not classification labels),
- we incorporate edge dynamics, allowing the adjacency matrix to evolve over time, and
- we combine graph attention with temporal attention, enabling simultaneous modeling of spatiotemporal dependencies.

To the best of our knowledge, this is the first architecture to address the regression task of node attribute prediction using fully dynamic graphs that doesn't assume a single global graph structure shared across inputs.

3 Method

We propose DySTFormer, a novel architecture for multistep, multivariate node attribute prediction in fully dynamic graphs, where both node features and edge structures evolve over time while the node set remains fixed. We introduce two variants of the model. The first, DySTFormer, learns time-varying graph structures and captures spatiotemporal dependencies through a combination of dynamic graph learning, graph attention, and temporal self-attention. The second variant, preDySTFormer, operates on predefined dynamic graphs rather than learning them. This distinction is motivated by application domains: in systems such as Bitcoin trust networks, where edge relationships are explicitly observed, it is advantageous to incorporate this structure directly. In contrast, for domains like the METR-LA traffic dataset, where the graph is latent, learning the structure from node attributes may be more appropriate.

3.1 Problem Formulation

Let $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ denote a graph snapshot at time t, where $\mathcal{V}$ is a fixed set of N nodes and $\mathcal{E}_t$ is the edge set at time t. Let $X_t \in \mathbb{R}^{N \times D}$ be the matrix of node

features at time t. Given a sequence $\{X_1, \ldots, X_L\}$, the goal is to predict future node features $\{X_{L+1}, \ldots, X_{L+H}\}$ (Figure 1), conditioned on both node feature evolution and the dynamic graph structure (given or inferred).

Input Representation. Each training sample consists of:

- A node attribute history tensor $X_{\text{hist}} \in \mathcal{R}^{N \times D \times L}$, where N is the number of nodes, D is the feature dimension of each node and L is the input sequence length.
- A dynamic graph sequence $A_{\text{hist}} \in \mathcal{R}^{N \times N \times L}$, representing one adjacency matrix per time step. This is optional, and in case predefined graphs are not provided, our model is able to learn the dynamic relationships that best align with the task (node attribute prediction).

The forecasting target is a trajectory $Y \in \mathcal{R}^{N \times D \times H}$ of node attributes over H future time steps.

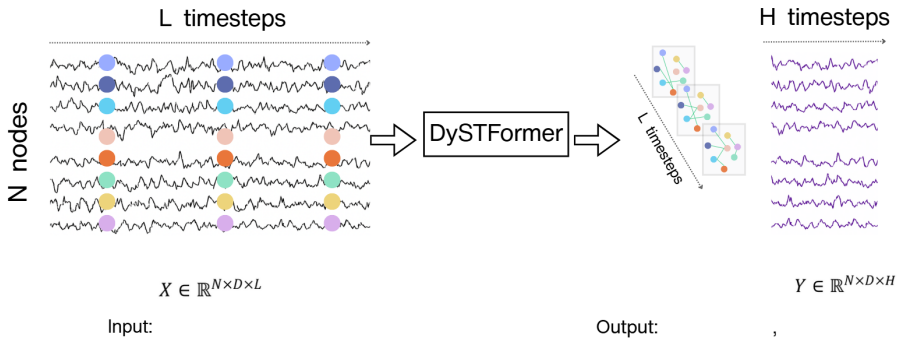

Fig. 1: High Level Problem Overview

3.2 Architecture Overview

The model architecture consists of four key components:

Dynamic Graph Learner. When the graph is not provided as input, to account for evolving connectivity, we learn a separate adjacency matrix $A_t \in \mathbb{R}^{N \times N}$ at each time step. We pass node features through an MLP-based encoder and compute pairwise attention scores:

$$A_t = \text{softmax}(E_t E_t^\top), \quad \text{where } E_t = \text{MLP}(X_t)$$

This component allows the model to dynamically infer the graph topology as a function of node features. The main distinction between *preDySTFormer* and *DySTFormer* is that *preDySTFormer* does not have this module, and operates on the given dynamic graphs directly.

Encoding and Temporal Position We begin by linearly projecting each node's input feature vector to a hidden dimension d, and add learnable temporal positional encodings:

$$Z = \text{Linear}(X_{\text{hist}}) + \text{PE}_{\text{time}}$$

This produces $Z \in \mathcal{R}^{N \times d \times L}$, which is then passed through a stack of spatiotemporal transformer layers.

Spatiotemporal Transformer Layers Each transformer layer integrates dynamic edge-aware multi-head attention by injecting per-time-step adjacency information as a learnable bias in the attention scores. This allows the model to adapt spatial attention to changing graph structure, which is critical for domains like brain networks and evolving trust graphs. At each layer, node i attends to node j at time t using:

$$\text{score}_{ij} = \frac{Q_i^\top K_j}{\sqrt{d}} + \text{Bias}_{ij}(A_t)$$

where Bias_{ij} is computed via a small MLP over $A_t[i, j]$. The output is passed through a residual feedforward block. We apply edge dropout during training to encourage robustness to noise.

Forecasting Decoder The output from the encoder, $Z_{\text{enc}} \in \mathcal{R}^{N \times d \times L}$, represents the encoded history of node features across the L input time steps, where N is the number of nodes and d is the hidden dimension. This tensor is first reshaped and permuted into a format suitable for sequence modeling, and passed through a GRU encoder to obtain an initial hidden state h_0 that summarizes the historical dynamics of each node.

The decoder then operates autoregressively over the prediction horizon H. At each future time step t, the decoder GRU generates an output z_t and new hidden state h_t based on the previous hidden state and the current input. The output is passed through a forecast-step MLP to produce the predicted node features at time t, denoted as $y_t \in \mathcal{R}^{N \times D}$, where D is the feature dimension. These outputs are collected over the entire forecast horizon to yield the final output tensor $Y_{\text{pred}} \in \mathcal{R}^{N \times D \times H}$.

Scheduled Sampling. During training, we employ *scheduled sampling* to bridge the gap between the training and inference conditions, a technique introduced by [2]. In traditional teacher forcing, the decoder is always fed the ground truth from the previous time step. However, at inference time, ground truth is not

available, and the decoder must rely entirely on its own predictions. To mitigate this train-test discrepancy, we probabilistically choose between using 1. only the decoder's previous prediction z_{t-1} and 2. a weighted combination of the ground truth Y_{t-1}^{true} and the model's own previous prediction z_{t-1} as input at each time step during training. This probability is governed by a decaying function of the training epoch, such that early in training, the decoder relies mostly on ground truth, and gradually transitions to using its own predictions as training progresses. This improves robustness and reduces error accumulation over long forecasting horizons.

The complete forecasted sequence is generated by unrolling the decoder over H time steps, using either sampled or predicted inputs, and projecting the hidden states back to the feature space through the MLP head.

Loss Functions Our loss function is a combination of Mean Absolute Error and a Variation loss:

$$\mathcal{L} = \mathcal{L}_{\text{MAE}} + \lambda \cdot \mathcal{L}_{\text{var}}$$

where λ is a weighting coefficient, and:

$$\mathcal{L}_{\text{MAE}} = \sum_{t=1}^{H} w_t \cdot \text{MAE}(Y_t^{\text{pred}},\ Y_t^{\text{true}}) \tag{1}$$

$$\mathcal{L}_{\text{var}} = \text{MSE}(Y_{t+1}^{\text{pred}} - Y_t^{\text{pred}},\ Y_{t+1}^{\text{true}} - Y_t^{\text{true}}) \tag{2}$$

The weights w_t are exponentially decaying to emphasize short-term accuracy. Variation loss penalizes differences in temporal derivatives (i.e., frame-to-frame changes) between the prediction and ground truth, and discourages oversmooth predictions by explicitly penalizing when the predicted signal lacks the expected variability over time.

Masked Pretraining To enhance the model's representation of spatiotemporal dependencies before supervised forecasting, we introduce a self-supervised masked pretraining objective. Inspired by masked language modeling in NLP [4], we randomly mask a subset of entries across nodes and timesteps in the input history tensor $X_{\text{hist}} \in \mathcal{R}^{N \times D \times L}$ and train the model to reconstruct these values using the corresponding adjacency sequence A_{hist}. This technique improves representation learning and has shown success in both sequence modeling and graph neural networks [4, 12, 6].

Masking Strategy. For each training sample, we generate a binary mask $M \in \{0, 1\}^{N \times D \times L}$ by sampling entries uniformly at random with probability $p_{\text{mask}} = 0.15$. The masked input $\tilde{X}_{\text{hist}}$ is created by zeroing out the selected entries:

$$\tilde{X}_{\text{hist}} = X_{\text{hist}} \odot (1 - M)$$

We feed $\tilde{X}_{\text{hist}}$ and A_{hist} into the encoder and decode a full reconstruction $\hat{X}_{\text{hist}}$ using the same projection head used in forecasting.

Loss Function. We compute a masked reconstruction loss that penalizes reconstruction error only at masked positions:

$$\mathcal{L}_{\text{pretrain}} = \frac{\|(\hat{X}_{\text{hist}} - X_{\text{hist}}) \odot M\|^2}{\|M\|_1 + \epsilon}$$

where ϵ is a small constant to avoid division by zero. This objective trains the model to learn generalizable spatiotemporal representations, even in the absence of forecasting supervision.

Pretraining Schedule. We first train the model for a fixed number of epochs using $\mathcal{L}_{\text{pretrain}}$ only, and then fine-tune on the forecasting task with the supervised loss $\mathcal{L} = \mathcal{L}_{\text{MAE}} + \lambda \mathcal{L}_{\text{var}}$. We observe that masked pretraining improves performance, particularly on datasets with noisy or irregular structure such as traffic and trust networks.

Training Procedure We train using Adam with a learning rate of 10^{-3} for up to 100 epochs, using early stopping based on validation RMSE. During early epochs, we apply curriculum learning by gradually increasing the forecast horizon.

4 Datasets and Baselines

Bitcoin Trust Networks : The Bitcoin Alpha and OTC trust networks from the SNAP repository [9, 8] consist of timestamped interactions, where users rate one another on a scale from -10 to 10. We derive two-dimensional node features at a daily resolution: the average rating given and the average rating received by each user. These features are used to construct time series of length 20, along with corresponding dynamic graphs. For prediction, we use the first 12 time steps as input and forecast the subsequent 8.

Traffic data : The METR-LA traffic dataset contains speed measurements from sensors distributed across the Los Angeles metropolitan area. The underlying network is static, defined by the physical distances between sensors [7]. To simulate dynamic topology, we generate semi-synthetic dynamic graphs by randomly removing 10% of the edges at each time step. We use sequences of 12 time steps as input and predict the following 12 time steps.

We split the datasets into training (70%), validation (20%), and test (10%) sets and normalize inputs. The output from the models are denormalized before calculating evaluation metrics.

Baselines and Metrics We compare against the most popular benchmark methods in this domain, namely, STGCN, DCRNN, and MTGNN, using MAE, and RMSE as evaluation metrics. We also compare our model against a fully connected long short-term memory network (LSTM) [5] that ignores graph structure and models the temporal dynamics of each node independently. It serves as a strong non-graph baseline for time series forecasting.

Table 1: Network Statistics

Dataset	**Nodes**	**Input length**	**Output length**	**Feature dimension**
Bitcoin Alpha	1254	12	8	2
Bitcoin OTC	1304	12	8	2
METR-LA	207	12	12	1

5 Results

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) results are presented in Table 2. For the Bitcoin trust datasets, node features are two-dimensional, whereas most baseline models operate on univariate time series. In such cases, we report the average of the two feature dimensions. All reported values are means over 10 independent runs, with standard deviations between 0.009 and 0.06. Our method outperforms existing approaches in all scenarios considered. The version that learns the adjacency matrix performs better in the traffic dataset, where the ground truth network is not explicitly observed, but inferred from node attribute similarities. On the other hand, *preDySTFormer* performs better on the trust networks, since it leverages explicitly observed relationships between nodes.

Table 2: Performance (MAE and RMSE) of different methods on various datasets.

Dataset	**Bitcoin Alpha**		**Bitcoin OTC**		**METR**	
Method	MAE	RMSE	MAE	RMSE	MAE	RMSE
LSTM	2.1	3.8	2.89	3.6	4.57	9.5
STGCN	1.8	3.6	1.09	4.7	4.5	9.5
DCRNN	1.81	3.47	1.3	2.8	3.9	7.56
MTGNN	2.18	4.5	2.2	3.2	3.42	7.23
preDySTFormer	**1.65**	**2.8**	**1.02**	**2.39**	**3.25**	6.9
DySTFormer	1.89	2.94	1.37	2.82	3.42	**6.12**

6 Conclusion

We introduced DySTFormer, a spatio-temporal transformer architecture designed for node attribute prediction on dynamic graphs. The model integrates a dynamic graph learner with attention-based mechanisms to jointly capture spatial and temporal dependencies. Empirical results on traffic datasets with evolving edge structures demonstrate improved predictive performance over strong spatiotemporal graph neural network (STGNN) baselines. In future work, we aim to extend this framework to more complex real-world dynamic systems, such as brain fMRI signals and animal movement networks.

References

1. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting (2020). https://doi.org/10.48550/ARXIV.2007.02842, https://arxiv.org/abs/2007.02842, version Number: 2
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 28 (2015)
3. Dai, C., Tang, Q., Ding, H.: TGAT: Temporal Graph Attention Network for Blockchain Phishing Scams Detection. In: 2024 International Conference on Computer, Information and Telecommunication Systems (CITS). pp. 1–7. IEEE, Girona, Spain (Jul 2024). https://doi.org/10.1109/CITS61189.2024.10608015, https://ieeexplore.ieee.org/document/10608015/
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2019)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
6. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. In: International Conference on Learning Representations (ICLR) (2020)
7. Jagadish, H.V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J.M., Ramakrishnan, R., Shahabi, C.: Big data and transportation engineering. In: Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE). pp. 1260–1264. IEEE (2014)
8. Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., Subrahmanian, V.: Rev2: Fraudulent user prediction in rating platforms. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. pp. 333–341. ACM (2018)
9. Kumar, S., Spezzano, F., Subrahmanian, V., Faloutsos, C.: Edge weight prediction in weighted signed networks. In: Data Mining (ICDM), 2016 IEEE 16th International Conference on. pp. 221–230. IEEE (2016)
10. Li, F., Feng, J., Yan, H., Jin, G., Jin, D., Li, Y.: Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution (2021), https://arxiv.org/abs/2104.14917
11. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR '18) (2018)
12. Liu, Y., Tang, J., Gao, J., Wang, Z., Yang, W.: Masked modeling of multivariate time series with transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 7326–7334 (2023)
13. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T.B., Leiserson, C.E.: EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs (Nov 2019). https://doi.org/10.48550/arXiv.1902.10191, http://arxiv.org/abs/1902.10191, arXiv:1902.10191 [cs]
14. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.: Temporal graph networks for deep learning on dynamic graphs. In: ICML 2020 Workshop on Graph Representation Learning (2020)

15. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Representation Learning over Dynamic Graphs (Mar 2018). https://doi.org/10.48550/arXiv.1803.04051, http://arxiv.org/abs/1803.04051, arXiv:1803.04051 [cs]
16. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks (May 2020). https://doi.org/10.48550/arXiv.2005.11650, http://arxiv.org/abs/2005.11650, arXiv:2005.11650 [cs]
17. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph WaveNet for Deep Spatial-Temporal Graph Modeling (May 2019). https://doi.org/10.48550/arXiv.1906.00121, http://arxiv.org/abs/1906.00121, arXiv:1906.00121 [cs]
18. Yu, B., Yin, H., Zhu, Z.: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. pp. 3634–3640 (Jul 2018). https://doi.org/10.24963/ijcai.2018/505, http://arxiv.org/abs/1709.04875, arXiv:1709.04875 [cs]