GEANN: Scalable Graph Augmentations for Multi-Horizon Time Series Forecasting

Sitan Yang
Forecasting Science, Amazon
New York, NY, USA
sitanyan@amazon.com

Malcolm Wolff
Forecasting Science, Amazon
New York, NY, USA
wolfmalc@amazon.com

Shankar Ramasubramanian
Forecasting Science, Amazon
New York, NY, USA
sramasub@amazon.com

Vincent Quenneville-Belair
Forecasting Science, Amazon
New York, NY, USA
quennv@amazon.com

Ronak Metha*
University of Washington
Seattle, WA, USA
ronakdm@uw.edu

Michael W. Mahoney
Forecasting Science, Amazon
New York, NY, USA
zmahmich@amazon.com

ABSTRACT

Encoder-decoder deep neural networks have been increasingly studied for multi-horizon time series forecasting, especially in real-world applications. However, to forecast accurately, these sophisticated neural forecasters typically rely on a large number of time series examples with substantial history. A rapidly growing topic of interest is forecasting time series which lack sufficient historical data—often referred to as the "cold start" problem. In this paper, we introduce a novel yet simple method to address this problem by leveraging graph neural networks (GNNs) as a data augmentation for enhancing the encoder used by such forecasters. These GNN-based features can capture complex inter-series relationships, and their generation process can be optimized end-to-end with the forecasting task. We show that our architecture can use either data-driven or domain knowledge-defined graphs, scaling to incorporate information from multiple very large graphs with millions of nodes. In our target application of demand forecasting for a large e-commerce retailer, we demonstrate on both a small dataset of 100K products and a large dataset with over 2 million products that our method improves overall performance over competitive baseline models. More importantly, we show that it brings substantially more gains to "cold start" products such as those newly launched or recently out-of-stock.

CCS CONCEPTS

• **Deep Neural Networks** → **Time Series Forecasting**; • **Graph Neural Networks** → *Graph Data Augmentation*.

KEYWORDS

Time Series Forecasting, Graph Data Augmentation, Graph Neural Networks, Scalability

ACM Reference Format:
Sitan Yang, Malcolm Wolff, Shankar Ramasubramanian, Vincent Quenneville-Belair, Ronak Metha, and Michael W. Mahoney. 2023. GEANN: Scalable Graph Augmentations for Multi-Horizon Time Series Forecasting. In *MLG '23: 19th KDD Workshop on Mining and Learning from Graphs, August 15th, 2023, Long Beach CA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

*Work performed during an internship at Amazon

MLG '23, August 15th, 2023, Long Beach CA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recent years have shown a growing interest in using deep neural networks (DNNs) for multi-horizon time series forecasting problems [2, 23, 24, 27, 38]. This is due to their flexibility of consuming various types of inputs and to their large model capacity to effectively learn on a huge amount of data, compared to traditional methods. Canonical DNNs (e.g., LSTM [18] and GRU [11]) have achieved success in many impactful real-world applications [6, 7, 44], and most recent methods in this field leverage Convolutional Neural Networks [e.g., 4, 9, 31, 39]

and Transformer architectures [e.g., 12, 21, 24, 25, 40] to further improve performance.

While a primary benefit of forecasting with DNNs is the ability to train on a large number of time series with long histories, these models also tend to rely on that scale to effectively forecast. Difficulties can arise, however, when there is a significant lack of historical data. This is often referred to as the "cold start" problem, and it has garnered recent interest in time series forecasting literature [e.g., 1, 5, 8, 28]. As many modern DNN forecasting models adopt a Seq2Seq structure [33], which assumes that the observed time series are uncorrelated, they tend to produce poor forecasts for "cold start" series. In these cases, learning relational information across series may bring additional performance gains to forecasting models.

Recent research has explored the use of Graph Neural Networks (GNNs) to capture complex inter-series relationships. While graph data augmentation has been studied extensively for general graph-based tasks such as node classification [26, 29] and edge prediction [37], its use in time series forecasting remains relatively under-explored. GNNs for time series explicitly model the relationship between observations by representing time series as nodes and their interactions as edges in a graph, and this has shown promising results in early work [22, 43]. More recently, [42] introduced using a retrieval mechanism to augment time series, achieving significant performance gains for multi-horizon forecasting. However, most major work in this field [see 45] have only used a single graph with no more than 1,000 nodes in evaluation, proposing methods which fail to scale to many practical applications. To alleviate this limitation, a few recent advances have improved the scalability of GNNs for time series forecasting by using altered, smaller graphs [15, 19, 30] and mini-batch sampling algorithms [10, 16].

In this paper, we expand on this literature by proposing a novel methodology—GEANN ("Graph Ensemble Augmented Neural Networks")—as a practical DNN solution to the "cold start" problem in large-scale time series forecasting. GEANN is a parsimonious model using GNNs as a data augmentation mechanism to enhance encoders typically used in the Seq2Seq forecasting architectures. We apply our method to the sequence structure of MQ-CNN from the family of MQ-Forecaster models [12, 39], which have shown state-of-the-art performances for time series, and especially for our target application – demand forecasting. We leverage the forecasting quality of MQ-CNN with graph-encoded information as an add-on component to improve representation learning. GEANN can scale to one or more very large graphs, which we demonstrate can lead to substantial performance improvements. Graphs in GEANN are predefined and static, and we can optionally assemble them using any domain-specific knowledge. In addition, we propose the use of pre-computed sparse graphs and their induced subgraphs to parallelize the GNN learning process on large and complex datasets.

To our best knowledge, this work is the first to show the benefit of GNNs as a data augmentation for time series forecasting in large scale real-world applications. We evaluate the proposed method in our target application – demand forecasting for a large e-commerce retailer, using both a small dataset consisting of ~100K products and a large application with over 2MM products. In both cases, we observe overall performance improvements over competitive MQ-Forecaster baselines. More importantly, we demonstrate that our method brings substantially larger gains in "cold-start" scenarios, such as new products with little to no sales history and recently out-of-stock products with sales history being incorrectly suppressed.

2 METHODS

2.1 Problem Formulation

Here and for the remainder of the paper, we denote tensors in boldface, matrices in upper case, and vectors in lower case. Let $Y \in \mathbb{R}^{N \times T}$ denote N time series of length T as targets, $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times T \times d}$ a set of d time series covariates, and $X^{(s)} \in \mathbb{R}^{N \times m}$ a set of m static covariates. Given a *context length* $C \geq 0$—i.e. the number of past observations used for modeling from the forecast time t—and a collection of *horizons* $\mathcal{H}$ to forecast in the future, we wish to generate the conditional forecast given $\mathbf{X}^{(t)}_{t-C:t} = (X^{(t)}_{t-C}, ..., X^{(t)}_{t})$, $Y_{t-C:t}$, and $X^{(s)}$ via the model

$$\widehat{Y}_{t,\mathcal{H}} = f\left(Y_{t-C:t}, \mathbf{X}^{(t)}_{t-C:t}, X^{(s)}; \boldsymbol{\theta}\right), \tag{1}$$

where $\boldsymbol{\theta}$ represents a collection of learnable parameters. To promote scalability, current deep learning architectures often consume the information of each observation i independently with the shared set of parameters $\boldsymbol{\theta}$ as

$$\widehat{y}_{i,t,\mathcal{H}} = f\left(y_{i,t-C:t}, X^{(t)}_{i,t-C:t}, x^{(s)}_{i}\ ;\ \boldsymbol{\theta}\right). \tag{2}$$

This treatment lends itself naturally to parallel computing, but it discards relational dependencies which may exist across time series, motivating the use of GNNs for improving equation (2) to learn such information.

GNN layers used for time series in this paper are generally of the form

$$g_{i,t} = \text{GNN}_\theta\left(H_t\ ;\ \mathcal{G}\right),$$

where the inputs $H_t \equiv (h_{i,t})_{i=1}^N \in \mathbb{R}^{N\times d_{\text{Enc}}}$ are intermediate embeddings, $\mathcal{G}$ is a graph describing pairwise relationships among observations, and the output is a graph-aware embedding. However, current GNN methods typically operate by modeling pairwise relations among observations across the entire graph simultaneously, leading to poor scalability for large graphs.

After the model is chosen, the parameters are tuned to optimize the loss during training as

$$\text{Loss}(\boldsymbol{\theta}) = \sum_i \sum_h \sum_t \ell(y_{i,t,h}, \widehat{y}_{i,t,h}). \tag{3}$$

The quantile loss (QL) function used in our experiments is detailed in Appendix A.

2.2 Model Architecture

Figure 1 summarizes our architecture. GEANN adopts the encoder-decoder architecture of MQ-CNN (as detailed in [39]) with our *graph ensemble module* (GEM) embedded. The encoder uses a stack of dilated temporal convolutions to summarize past targets and time-varying covariates into a sequence of hidden states $H_t \equiv (h_{i,t})_{i=1}^N \in \mathbb{R}^{N\times d_{\text{Enc}}}$. The proposed GEM component encodes H_t to additional hidden states $G_t \equiv (g_{i,t})_{i=1}^N \in \mathbb{R}^{N\times d_{\text{GNN}}}$,

$$\text{GEM}_{\boldsymbol{\theta}}(H_t; \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(R)}).$$

through R individual GNN operations, each of which on separate fixed graphs $\mathcal{G}^{(r)}$. That is, for $r = 1, ..., R$,

$$g_{i,t}^{(r)} = \text{GNN}_\theta^{(r)}\left(H_t\ ;\ \mathcal{G}^{(r)}\right).$$

The encoded states from all GNN layers are combined with trainable weights $w^{(r)} \geq 0, \sum_{r=1}^R w^{(r)} = 1$ to yield the final output:

$$g_{i,t} = \sum_{r=1}^R w^{(r)} g_{i,t}^{(r)} \quad \forall i.$$

Each graph remains static, but the node embeddings are dynamic across all time steps t.

Many prior methods and associated GNN libraries for graph learning such as DGL [36] and PyTorch Geometric [14] typically require the entire node embedding set H_t and the whole graph structure $\mathcal{G}^{(r)}$ during each forward pass, regardless of batch size, which is infeasible for large datasets. We instead propose a learning algorithm for GEANN (see details in Appendix B) which applies "top-k" neighborhood sampling and induces "L-hop" subgraphs; this uses nodes in each mini-batch as seed nodes to alleviate the scaling issue (see, e.g. [10, 16] for related techniques). We note that the graph sparsity controls the use of GPU memory during training.

The resulting embeddings G_t are then combined with H_t for augmenting the representation learning process. Notice that this process is done for each t separately during training, and the parameters are optimized together with the subsequent forecasting task.

In this paper, we adopt the same decoder as in MQ-CNN, but we note that any other decoder can be used. Formally, our GEANN architecture is as follows:

$$\begin{aligned}
h_{i,t} &= \text{Encoder}_{\boldsymbol{\theta}}^{(t)}\left(y_{i,t-C:t}, X_{i,t-C:t}^{(t)}\right),\\
h_i^{(s)} &= \text{Encoder}_{\boldsymbol{\theta}}^{(s)}\left(x_i^{(s)}\right),\\
G_t &= \text{GEM}_{\boldsymbol{\theta}}\left(H_t; \mathcal{G}^{(1)}, ..., \mathcal{G}^{(R)}\right),\\
\hat{y}_{i,t,\mathcal{H}} &= \text{Decoder}_{\boldsymbol{\theta}}(h_{i,t}, g_{i,t}, h_i^{(s)}).
\end{aligned}$$

For the GNN operation, we use L graph convolutional network (GCN) layers [17] for a L−hop neighborhood configuration, but this can be easily extended to other types of graph learning layers such as graphSAGE [16] or GAT [35].

Graph Construction. GEANN can use any predefined graph along with optionally a specified weight for each edge. Here we model each of N time series in $H_t, t = 1, \ldots, T$ as a node and their interactions as edges, and we generally consider two types of graphs: the data-driven graph, and the domain knowledge-defined graph. GEANN natively supports sparse graphs which can be efficiently stored as edge lists. For each graph constructed, we further apply an additional "top-k" operation during training to control the GPU memory usage of each mini-batch so that the process can be efficiently parallelized.

We construct a data-driven graph using a similarity metric between node i and j. In this paper, we choose the Pearson correlation coefficient as the metric, but other metrics can also be used. We adopt the same method used in [42] to base our calculations on the embedding vectors generated by a pretrained MQ-CNN model as $S_{\text{Corr}}(i,j) = \left|\text{Corr}\left(H_i^{(0)}, H_j^{(0)}\right)\right|$. Here $H_i^{(0)}$ denotes the pretrained version of $h_{i,t}, t = 1, ..., T$ from a frozen MQ-CNN model, and $\text{Corr}(\cdot,\cdot)$ indicates the

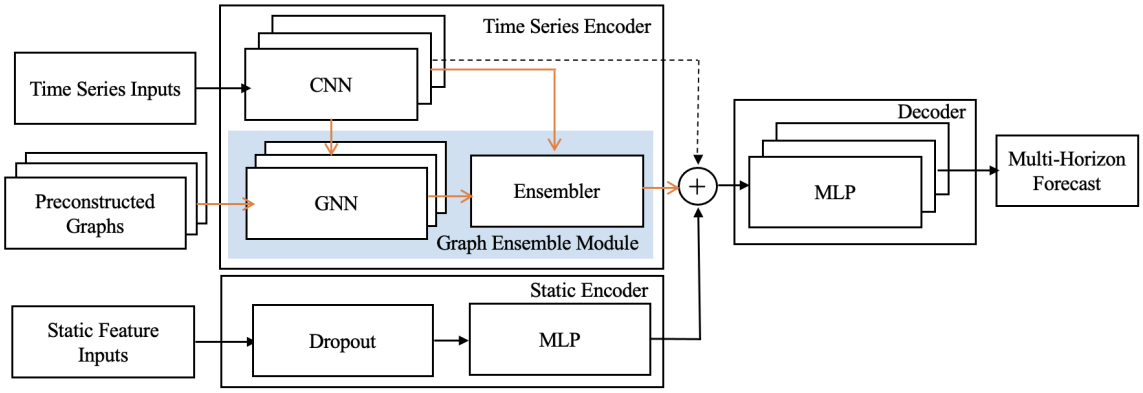

Figure 1: Based on the encoder-decoder structure of MQ-CNN [39], the embedded graph ensemble module of GEANN maps the time series encoded input $h_{i,t}$ through a number of GNN operations (details below). The outputs are then used to form the ensemble representation $g_{i,t}$, which are decoded together with $h_{i,t}$ into forecasts.

Pearson correlation coefficient. Based on the similarity metric we further obtain a k-NN embedding graph.

We also consider constructing graphs with domain-related knowledge available. For online marketplace demand forecasting, there often exists a rich set of relational information (e.g., site catalog information and customer browsing data). In this paper, we use *browse nodes*, which are attributes visible on the website of many e-commerce retailers to help customers navigate through the vast selection of products. For example, browse node set for a men's fashion sweatshirt consists of "Clothing, Shoes & Jewelry", "Men", "Clothing" and "Fashion Hoodies & Sweatshirts." These correspond to the nodes visited in the browsing tree to locate the product. Thus, products that belong to the same browse node are likely to be co-browsed by customers, providing a notion of both substitutable and complimentary goods.

We note that unlike the "ground-truth" graph used in spatio-temporal tasks such as the traffic flow forecasting [22, 43], the graph derived from browse nodes or other similar attributes only indicates one of many complex and subtle relationships existing among data, and it is inherently more difficult to show the value-add of graph learning in this case.

3 RESULTS

In this section we evaluate GEANN on two demand datasets from a large e-commerce retailer, that include time series features such as unit sales, promotions, holidays and detail page views as well as static metadata features such as catalog information. Similar datasets with the same set of features but generated in different time windows have been used in [12, 39, 42]. We have obtained five years (2016-2021) of history for time series data. Each model is trained on three years (2016-2019) of demand data, one year is held out for validation, and the final year is kept for evaluation. The task is to forecast the 50th and 90th quantile of weekly demand for up to one year at each of the 52 forecast creation time (see Appendix A for metric details).

GEANN is implemented using the Pytorch framework with 8 NVIDIA V100 Tensor Core GPUs. The model is optimized using AdamW [20] with default parameters. We limit our node neighborhood to a depth of 2 with a maximum neighborhood size of 10. For the graph encoding, we use a 2-layer GCN model with 32 hidden units.

3.1 Small Scale Experiment

The first dataset consists of 100K products with the largest number of total units sold during the training period from 5 main European marketplaces (EU5), and we compare the following architectures:

- **MQ-CNN**: the MQ-CNN model [39]
- **MQ-T**: the MQ-Transformer model [12]
- **GEANN-bw**: GEANN using a browse node graph
- **GEANN-kNN**: GEANN using a kNN embedding graph
- **GEANN-bw+kNN**: GEANN using both a browse node and kNN embedding graph

MQ-CNN and MQ-T serve as the baseline models in the comparison.[1] For GEANN, each graph contains 100K nodes. For

[1]We mainly focus our comparison on MQ-CNN, as MQ-T requires substantially more GPU memory, and we are already memory bound for GEANN. Moreover, we see that the improvement of MQ-T on MQ-CNN is orthogonal to those in GEANN, and thus they can be combined in the future study.

the browse node graph, we rank all other products related to a particular product by the number of times where they appear together in the same browse node, and we choose the top 10 products as the neighbors. For the kNN embedding graph, we consider the 10 nearest neighbors calculated on the pretrained embeddings of MQ-CNN. We also include the performance of GEANN with both graphs. Furthermore, for the purpose of ablation, we include 3 additional model configurations:

- MQ-CNN-L: MQ-CNN with the number of parameters matching that of GEANN by increasing the dilation capacity of the CNN layer
- GEANN-idm: GEANN with a "zero-neighbor" graph (i.e., adjacency matrix being an identity matrix) that contains no additional graph-related information
- GEANN-random: GEANN with a randomly connected graph with no predictive information expected

We train each model to 100 epochs using batch size of 256, and each model evaluation is computed with 5 identical runs using different random seeds. We summarize each model performance in Table 1. We observe all GEANN variants lead to noticeable overall performance improvements (~2%-3%) over baselines. Notably, the three ablations do not seem to improve upon baselines, even with similar number of parameters compared to GEANN. Hence, the predictive information in the browse node and k-NN embedding graph cannot be produced from a randomly generated graph. GEANN-idm is expected to be similar to MQ-CNN, and indeed it has on-par performance. The ensemble of the two graphs does not seem to produce further accuracy gain in our experiment.

Table 1: Performance metrics on 100K EU5 retail products. The results are rescaled so that they are relative improvements over MQ-CNN. Lower is better. The number of parameters used by each model is also included.

Model	P50 QL	P90 QL	Overall	Param
MQ-CNN	1.000	1.000	1.000	850k
MQ-T	1.060	0.999	1.029	858k
MQ-CNN-L	1.059	1.047	1.053	898k
GEANN-idm	1.005	0.1003	1.004	900k
GEANN-random	1.003	0.993	0.998	900k
GEANN-bw	0.977	**0.974**	**0.976**	900k
GEANN-kNN	0.979	0.977	0.978	900k
GEANN-bw+kNN	**0.969**	0.983	**0.976**	915k

3.2 Large Scale Application

We test on a large scale application of demand forecasting that involves over 2MM products with most sales from North America marketplaces (NA). Leveraging a graph with millions of nodes is a rare challenge considered in the previous graph-based methods [32, 41]. Again using MQ-CNN and MQ-T as baselines, we compare the test performance of GEANN-bw, GEANN-kNN and GEANN-bw+kNN. We train each model with a batch size of 512. Each epoch takes around 30 minutes for the model with a single graph and 1 hour with two graphs. Each model evaluation is averaged across 3 identical runs, and test results are summarized in Table 2. We notice that GEANN improves over baselines with the browse node graph, but in this case it degrades with the kNN graph, which is likely to confirm the predictive information from browsing data. A fundamental question related to degraded performance when using the data-driven graph is whether it is a function of the model architecture or graph estimation procedure. Further analysis of the data-driven graph suggests the data-driven graph is highly volatile when k is small, providing a low signal-to-noise ratio (detailed in Section 3.3). The two graph ensemble method performs on-par with the baselines mainly due to the underperformance of the kNN graph.

Table 2: Performance metrics on 2 million NA retail products. The results are rescaled so that they are relative improvements over MQ-CNN. Lower is better.

Model	P50 QL	P90 QL	overall
MQ-CNN	1.000	1.000	1.000
MQ-T	0.999	0.998	0.998
GEANN-bw	0.997	**0.985**	**0.991**
GEANN-kNN	1.023	1.044	1.033
GEANN-bw+kNN	**0.993**	1.001	0.997

Newly launched and Recently Out-Of-Stock Products. For GEANN-bw, in addition to overall performance gain, we also find that it significantly improves for two challenging and important groups: newly launched, and recently out-of-stock (OOS) products. Forecasting demand for new products is generally difficult due to little to no time series history. For recently OOS products, past sale history during OOS period is no longer a good signal for demand prediction due to inventory constraint. For these two cases, we show in Table 3 that GEANN brings substantially more gains (~5%) compared to the overall improvement. Our empirical analysis indicates

that these accuracy improvements tend to stem from faster calibration of demand forecasts for new and OOS products, by adjusting their forecasts based on the historic demand of similar products used as neighbors in GEANN. Hence by incorporating relational information, our model is able to leverage greater contextual understanding to quickly calibrate forecasts for cold-start products.

Table 3: Model performance comparison for newly launched and recently out-of-stock products relative to MQ-CNN.

Model	GEANN-bw		
	P50 QL	P90 QL	overall
Newly Launched	0.975	0.910	0.943
Recently OOS	0.974	0.957	0.965

3.3 Data-Driven Graph Stability

In this paper, we have considered both data-driven and domain-knowledge defined graphs. We observe that the performance of using a data-driven graph is volatile, while the domain-knowledge graph seems to produce consistent improvements across different datasets. In this section, we conduct an in-depth analysis for the data-drive graph used to provide insights for this result.

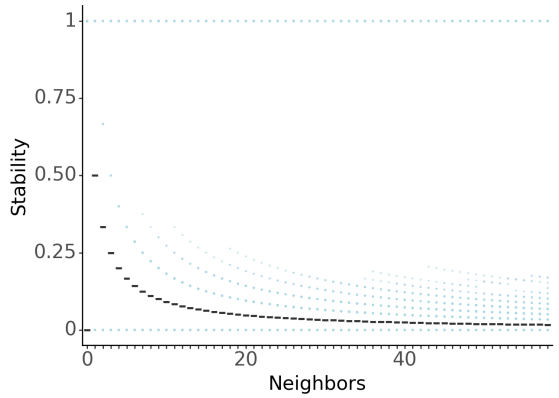

Figure 2: Boxplots of stability by number of neighbors for k-NN generated from Pearson correlation between embeddings H across 3 runs of MQ-CNN. Blue points represent outliers.

Distribution of similarity metrics. We first compare the nearest neighbor distribution for these two graphs, respectively, for the large scale dataset used in Section 3.2. Appendix Figure 3 depicts a histogram of the estimated means and standard deviations of the Pearson correlations from 10 nearest neighbors for each product in GEANN-kNN. The substantial concentration of mean correlations near unity, and similarly the concentration of standard deviation near zero, show that many of the products have nearly interchangeable rank. On the other hand, Appendix Figure 4, which shows a similar histogram as those for co-browsing counts from 10 nearest neighbors for each product in GEANN-bw, presents much stronger variation. Moreover, it is clear that a subset of products have strong similarity relative to others, with mean co-browsing counts above 35.

On k-NN graph volatility. One reason such context-based k-NN graphs may show unreliable relationships relevant to forecasts is the stability of the k-NN construction on embeddings $\mathbf{H}$. We define stability for a product a and number of neighbors k across runs r, each run generating a context $\mathbf{H}^{(r)}$, as

$$\text{Stability}(a; k, \mathcal{R}) \equiv \frac{\left|\bigcap_{r\in\mathcal{R}} \text{KNN}_r(a; k)\right|}{k}.$$

Stability is a measure of preservation of graph similarity across model training runs; a random selection of neighbors has a stability proportional to a HyperGeometric(N, k, k), where N is the number of data observations. When N is reasonably large, the stability is close to 0, and a deterministic graph has a stability of 1.0. Prior work in natural language processing field shows the stability of frequently used word embeddings are often 0.8 or greater [3].

Figure 2 shows the distribution of stability by the number of neighbors for our embedding based k-NN using Pearson correlation across 3 model runs. We find generally low stability of k-NN relationships generated from the embeddings $\mathcal{H}$. While the stability of low numbers of neighbors is relatively larger at 0.5, it quickly decreases to represent an approximately random graph as the number of neighbors increases— likely due to orderings based on correlation being overcome by model noise. Moreover, the stability is highly consistent across products. A maximal stability of approximately 0.5 suggests that the k-NN based graph may provide an inconsistent signal due to noise in the embeddings $\mathbf{H}$.

4 CONCLUSION

In this paper, we demonstrate that for demand forecasting our proposed graph-based time series forecasting method GEANN outperforms the current Seq2Seq models while maintaining

the scaling advantage. One interesting future direction is to consider different GNN architecture such as the graph attention network architecture [34]. In addition, more sophisticated graph construction methods can be used, such as choosing graphs and model parameters based on their Network Community Profile curves [13].

REFERENCES

[1] Carlos Aguilar-Palacios, Sergio Muñoz-Romero, and José luis Rojo-Álvarez. 2020. Cold-start promotional sales forecasting through gradient boosted-based contrastive explanations. *IEEE Access* 8 (2020), 137574–137586.

[2] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Franccois-Xavier Aubet, Laurent Callot, and Tim Januschowski. 2022. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *Comput. Surveys* (may 2022).

[3] Angana Borah, Manash Pratim Barman, and Amit Awekar. 2021. Are word embedding methods stable and should we care about it?. In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*. 45–55.

[4] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. 2017. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (2017).

[5] J Bottieau, Z De Grève, T Piraux, A Dubois, F Vallée, and J-F Toubeau. 2022. A cross-learning approach for cold-start forecasting of residential photovoltaic generation. *Electric Power Systems Research* 212 (2022), 108415.

[6] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang (Bernie) Wang. 2017. Probabilistic demand forecasting at scale. In *VLDB 2017*.

[7] Carlos Capistrán, Christian Constandse, and Manuel Ramos-Francia. 2010. Multi-horizon inflation forecasts using disaggregated data. *Economic Modelling* 27, 3 (2010), 666–677.

[8] Ayush Chauhan, Archiki Prasad, Parth Gupta, Amiredddy Prashanth Reddy, and Shiv Kumar Saini. 2020. Time Series Forecasting for Cold-Start Items by Learning from Related Items using Memory Networks. In *Companion Proceedings of the Web Conference 2020*. 120–121.

[9] Yitian Chen, Yanfei Kang, Yixiong Chen, and Zizhuo Wang. 2020. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* 399 (2020), 491–501.

[10] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. 257–266.

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

[12] Carson Eisenach, Yagna Patel, and Dhruv Madeka. 2020. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. https://arxiv.org/abs/2009.14799

[13] Evgeniy Faerman, Felix Borutta, Kimon Fountoulakis, and Michael Mahoney. 2017. LASAGNE: Locality And Structure Aware Graph Node Embedding. *IEEE/WIC/ACM International Conference on Web Intelligence* (10 2017).

[14] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[15] Ankit Gandhi, Aakanksha, Sivaramakrishnan Kaveri, and Vineet Chaoji. 2021. Spatio-Temporal Multi-Graph Networks for Demand Forecasting in Online Marketplaces. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV*. 187–203.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, Vol. 30.

[17] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. https://arxiv.org/abs/1506.05163

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

[19] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. 2021. https://arxiv.org/abs/2106.05150

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[21] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[22] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[23] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJiHXGWAZ

[24] Bryan Lim. 2018. Forecasting Treatment Responses Over Time Using Recurrent Marginal Structural Networks. In *Advances in Neural Information Processing Systems*, Vol. 31.

[25] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

[26] Songtao Liu, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. 2021. Local Augmentation for Graph Neural Networks. *CoRR* abs/2109.03856 (2021). arXiv:2109.03856 https://arxiv.org/abs/2109.03856

[27] Dhruv Madeka, Lucas Swiniarski, Dean Foster, Leo Razoumov, Kari Torkkola, and Ruofeng Wen. 2018. Sample path generation for probabilistic demand forecasting. In *KDD 2018 Workshop on Mining and Learning from Time Series*. https://www.amazon.science/publications/sample-path-generation-for-probabilistic-demand-forecasting

[28] Jihoon Moon, Junhong Kim, Pilsung Kang, and Eenjun Hwang. 2020. Solving the cold-start problem in short-term load forecasting using tree-based methods. *Energies* 13, 4 (2020), 886.

[29] Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J Kim. 2021. Metropolis-Hastings Data Augmentation for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 19010–19020.

[30] Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski. 2022. Multivariate Time Series Forecasting with Latent Graph Inference. https://openreview.net/forum?id=JpNH4CW_zl

[31] Omer Berat Sezer and Ahmet Murat Ozbayoglu. 2018. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing* 70 (2018), 525–538.

[32] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete graph structure learning for forecasting multiple time series. *arXiv preprint arXiv:2101.06861* (2021).

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[34] Petar Velivcković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. https://arxiv.org/abs/1710.10903

[35] Petar Velivcković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[36] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[37] Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Siddharth Bhatia, and Bryan Hooi. 2021. Adaptive Data Augmentation on Temporal Graphs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 1440–1452.

[38] Ruofeng Wen and Kari Torkkola. 2019. Deep generative quantile-copula models for probabilistic forecasting. In *ICML 2019 Workshop on Time Series*. https://www.amazon.science/publications/deep-generative-quantile-copula-models-for-probabilistic-forecasting

[39] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A Multi-Horizon Quantile Recurrent Forecaster. https://arxiv.org/abs/1711.11053

[40] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. 2020. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317* (2020).

[41] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. arXiv:2005.11650 [cs.LG]

[42] Sitan Yang, Carson Eisenach, and Dhruv Madeka. 2022. MQRetNN: Multi-Horizon Time Series Forecasting with Retrieval Augmentation. KDD. https://doi.org/10.48550/ARXIV.2207.10517

[43] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[44] J N Zhang and Kazumitsu Nawata. 2018. Multi-step prediction for influenza outbreak by an adjusted long short-term memory. *Epidemiology and Infection* 146 (2018), 809 – 816.

[45] Daniel Zügner, Victor Garcia Satorras, Tim Januschowski, Stephan Günnemann, and Jan Gasthaus. 2021. A study of joint graph inference and forecasting. In *ICML 2021 Time Series Workshop*.

A DISTRIBUTIONAL QUANTILE FORECASTS

We refer to distributional forecasts as quantile forecasts, and we consider our target application (demand forecasting) as predicting certain quantiles of the future demand distribution at a weekly grain for up to one year in the future. A natural metric measuring how accurate a quantile forecast f at the q-th quantile with respect to the true demand d is the quantile loss L (QL), as defined below:

$$L_q(d, f) = q(d - f)_+ + (1 - q)(f - d)_+, \tag{4}$$

where $(\cdot)_+ = \max(\cdot, 0)$. This metric is typically aggregated across samples and time horizons, and it is weighted by their actual demand. The weighted P50 and P90 QL are used to measure the quality of distributional forecasts as the 50th and 90th percentile of the demand distribution. The same metric has been used for previous work on multi-horizon time series forecasting [12, 24, 39, 42].

B GEANN LEARNING ALGORITHM

In this section, we describe the learning algorithm of GEANN. See the details in Algorithm 1. While stochastic gradient descent (SGD) is the optimization algorithm shown, in practice other methods such as Adam [20] is often used. The graph ensemble module first identifies the time series in the current mini-batch (denoted as $\mathcal{M}_b$). For each graph $\mathcal{G}^{(r)}$, the module determines a subgraph $\overline{\mathcal{G}}^{(r)}$ of $\mathcal{G}^{(r)}$ induced by the "L-hop-out neighborhood" of the seed nodes $\mathcal{M}_b$, and containing all nodes $\overline{\mathcal{M}}^{(r)}$ reached by traversing at most L edges on $\mathcal{G}^{(r)}$. This is also the set of nodes reached by applying an L-layer graph convolution on seed nodes [16]. In this process, $\mathcal{G}^{(r)}$ and $\overline{\mathcal{G}}^{(r)}$ are guaranteed to produce the same output for seed nodes in $\mathcal{M}_b$, ensuring that a gradient estimated with elements of $\mathcal{M}_b$ remains unbiased. For large graphs, the neighborhood memberships can be pre-computed and stored offline while only retrieving nodes of $\overline{\mathcal{G}}^{(r)}$ online in each mini-batch, effectively reducing the use of GPU memory. Notice that only the graph representations of seed nodes are used in the decoder and backpropagation step. We note that the sparsity of $\mathcal{G}^{(r)}$

Algorithm 1 GEANN Learning Algorithm

Require: Number of epochs E, mini-batch size m, number of GNN layers L, candidate graphs $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(R)}$, context length C, maximum horizon H, training data $\left(\mathbf{y}_{i,1:T}, \mathbf{x}_{i,1:T}, \mathbf{c}_i\right)_{i \in [N]}$, loss function ℓ, learning rate η, initial parameters $\boldsymbol{\theta}$.

for $i_{\text{ep}} \in \{1, \ldots, E\}$ **do**
  Partition $[N]$ into mini-batches $\mathcal{M}_1, \ldots, \mathcal{M}_{N/m}$
  for $b \in \{1, \ldots, N/m\}$ **do**
    $h_{i,t} = \text{Encoder}_\theta\left(\mathbf{y}_{i,t-C:t}, \mathbf{X}_{i,t-C:t}, X_i^{(s)}\right)$
    for $r \in \{1, \ldots, R\}$ **do**
      $\overline{\mathcal{M}}^{(r)}, \overline{\mathcal{G}}^{(r)} = \text{GetHopSubgraph}(\mathcal{M}_b, \mathcal{G}^{(r)}, L)$.
      $h_{i,t}^{(r)} = \text{Encoder}_\theta\left(\mathbf{y}_{i,t-C:t}, \mathbf{X}_{i,t-C:t}, X_i^{(s)}\right)$
      $\mathbf{H}_t^{(r)} = (h_{i,t}^{(r)})_{i \in \overline{\mathcal{M}}^{(r)}}$.
      $\left(g_{i,t}^{(r)}\right)_{i \in \overline{\mathcal{M}}^{(r)}} = \text{GNN}_\theta^{(r)}\left(\mathbf{H}_t^{(r)}, \overline{\mathcal{G}}^{(r)}\right)$.
    end for
    $\mathbf{g}_{i,t} = (g_{i,t}^{(1)}, \ldots, g_{i,t}^{(R)})$ for all $i \in \mathcal{M}_b$.
    $\left(\hat{y}_{i,t,\mathcal{H}}\right) = \text{Decoder}_{\boldsymbol{\theta}}(h_{i,t}, \mathbf{g}_{i,t})$.
    $\text{Loss}(\boldsymbol{\theta}) = \sum_{i \in \mathcal{M}_b} \sum_t \ell(y_{i,t,\mathcal{H}}, \hat{y}_{i,t,\mathcal{H}})$.
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla \text{Loss}(\boldsymbol{\theta})$.
  end for
end for
return $\boldsymbol{\theta}$.

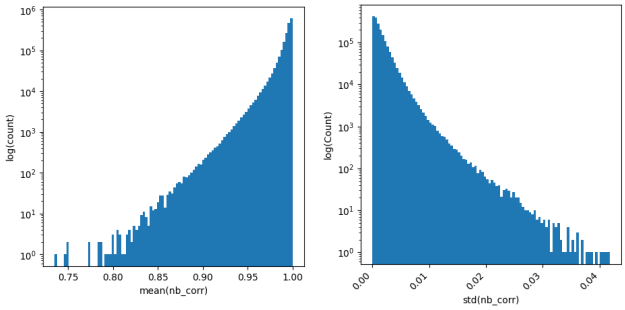

Figure 3: Histogram of the estimated mean and standard deviation of the Pearson correlations in GEANN-kNN.

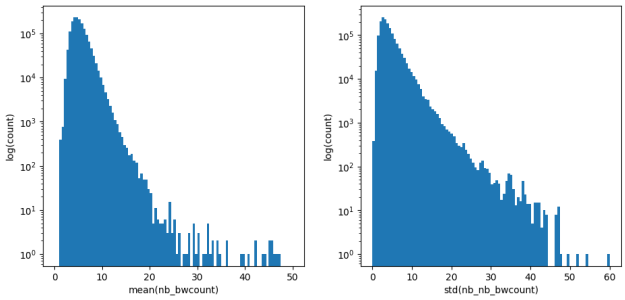

Figure 4: Histogram of the estimated mean and standard deviation of the co-browsing counts in GEANN-bw.

is critical for the learning algorithm to be efficiently trained in parallel. This directly controls the size of $\overline{\mathcal{M}}^{(r)}$. Therefore in this process, we typically apply a "top-k" operation on the given graph so that if $\mathcal{M}_b$ has m nodes, the upper bound for the number of nodes in $\overline{\mathcal{M}}^{(r)}$ is $m(1+k^L)$. Consequently we can determine the value of (m, k) based on memory constraints on the GPU.

C ADDITIONAL FIGURES FOR DATA-DRIVE GRAPH ANALYSIS

In this section, we include additional figures for the data-drive graph stability analysis conducted in Section 3.3. See Figure 3 and Figure 4. In particular, we calculate using the kNN embedding graph the means and standard deviations of the Pearson correlations from 10 nearest neighbors of each product, and we compare with those calculated from the browse node graph. We note that the score associated in the browse node graph is the co-browsing count, i.e., the number of times a certain product and its neighbor product belong to the same node in the browsing tree, and we use this score to rank all neighbor products for applying the "top-k" operation.