

This item is the archived peer-reviewed author-version of:

Towards graph-based forecasting with attention for real-world restaurant sales

Reference:






Moura Henrique Duarte, Kholkin Leonid, D'eer Lynn, Mets Kevin, De Schepper Tom.- Towards graph-based forecasting with attention for real-world restaurant sales

... IEEE ... International Conference on Data Mining workshops - ISSN 2375-9232 - Los Alamitos, IEEE Computer Soc, (2023), p. 560-569

Full text (Publisher's DOI): <https://doi.org/10.1109/ICDMW60847.2023.00079>

To cite this reference: <https://hdl.handle.net/10067/2047290151162165141>

Towards graph-based forecasting with attention for real-world restaurant sales

Henrique Duarte Moura^{*}, Leonid Kholkine[†], Lynn D’eer[‡], Kevin Mets[§] and Tom De Schepper[¶]

University of Antwerpen - imec, Belgium

Email: { *henrique.duartemoura, †leonid.kholkine, ‡lynn.deer, §kevin.mets, ¶tom.deschepper }@uantwerpen.be

Abstract—Forecasting is the process of predicting future events or values based on past data to support, for instance, strategic business decisions. This historical data often takes the form of multivariate time series. Recently, graph networks emerged as a powerful forecasting framework by considering spatiotemporal relations in the data. In this paper, we describe the adoption of a forecasting model built in an encoder-decoder architecture. The encoder projects the spatiotemporal relations into a lower dimension subspace, whereas the decoder uses the generated embeddings to produce the predicted value for a given period. We test the model on three datasets with data from different areas. Two datasets are public benchmarks with data from disease spread and street traffic scenarios which are used to evaluate the performance of our architecture. Additionally, we also evaluate our model on a new private company-owned dataset with product sales on a chain of restaurants. This dataset is more diverse, especially in terms of variety across different restaurants. We provide a deep dive on the challenges posed by it. We experiment with various state-of-the-art models to compare the performances on all datasets. The results show that our model obtains overall good predictions, outperforming all SOTA models, in terms of accuracy, by 5.2% and 9.8%, on the benchmark datasets. On the real-world restaurant data, our model outperforms all benchmarks and surpasses the second best by 140%.

Index Terms—Time series, Forecasting, Deep Learning, Graph Networks.

I. INTRODUCTION

Forecasting models support strategic decisions in several sectors, as they use historical multivariate time series to predict future trends and, thus, effectively help in decision making. Sales forecast shows several challenges, as many factors can affect sales [1]. Some factors are part of the retailer’s marketing plan (such as pricing and promotions). There can be “secondary” effects like interaction or cannibalization, delisted or promoted substitute or complementary products. Sales can be affected also by external factors (such as sporting events, seasons and holidays), and by unpredictable factors and anomalous events (*e.g.*, COVID, strikes or terrorist attacks). There are also some extra factors that may need to be revealed or even anticipated (*e.g.*, competitive behavior, local/national economy, weather, etc). Complex relationships between variables, including static and dynamic variables, and predictable and latent relationships, make it possible to extract more information from time series.

Modeling these complex relationships and behavior is essential, not only to characterize the latent dependence between variables, but also to model temporal relationships. Traditional methods, such as autoregressive integrated moving average

(ARIMA), and vector auto-regression (VAR) approaches, focus mainly on modeling relationships between a variable, past observations of itself and past observations of other variables. Deep learning models can handle non-linear relationships in the data and, therefore, obtain remarkable prediction results. However, it is difficult to provide these models with spatial relationships between time series, even with methods based on recurrent networks. Therefore, these methods are inefficient in processing the space-time dependencies of time series. Thus, a new representation of the input data can allow these dependencies to be considered.

Graph networks allow characterizing the complex relationships between several objects of interest, where each one is considered as a graph node. These nodes have characteristics that vary over time, for example, the sales in different stores. The relationships between the nodes are represented by edges, which can reflect a static or dynamic relationship, *i.e.*, the edges can remain the same for the whole time series or change from time to time. This data representation can be processed by a group of neural networks called graph neural networks (GNN), which can be directly applied to graphs and provide an easy way to do node-level, edge-level, and graph-level prediction tasks.

In this paper, we focus on an end-to-end graph-based deep learning model and apply this in a real-world restaurant setting. The main contributions of our work are summarized as follows:

- We develop a spatiotemporal encoder-decoder architecture with an attention mechanism, where the encoder projects the spatiotemporal relations into a lower dimension subspace, and the decoder uses the generated embeddings to produce the predicted forecast value for a given period;
- We execute extensive experiments on two benchmark datasets on disease spread and street traffic. These very different datasets were used to validate the model’s performance in the forecasting task. The results show that our model outperforms the benchmark state-of-the-art (SOTA) models by 5.2% and 9.8%, respectively;
- We introduce a private dataset on restaurant sales for which our forecasting model ranks second. We discuss in detail the challenges and constraints with real-world data and potential mitigations. Our model outperforms the benchmark state-of-the-art (SOTA) models by 140% when compared to the second best model.

The paper has the following structure: An overview of related work can be found in Section II. Section III contains a descriptive analysis of the real-world restaurant dataset. In Section IV, the proposed forecasting model architecture is presented. We discuss the experimental results in Section V and the challenges of forecasting on real data in Section VI. This work concludes with Section VII.

II. RELATED WORK

Predicting future events using mathematical forecasting models is a classic problem researched for decades. These models use historical (*i.e.*, timestamped) observations to predict future values over a period of time or at a specific point in the future. As these future values are unavailable, they can only be estimated. Early studies mainly employ model-driven approaches such as ARIMA [2], VAR [3], and Kalman filters [4], which are still widely adopted in real-world applications. However, they fail to deal with complex nonlinear variables as they assume that the time series is stationary, which is only satisfied under limited conditions. On the other hand, data-driven methods employ machine learning to automatically discover patterns in historical data. Algorithms like support vector regression [5, 6], k-nearest neighbors, and tree-based regressions, *e.g.*, XGBoost [7, 8, 9], among others, can handle complex non-linear data. Due to the great advances in deep learning, research with sequence-by-sequence methods, such as deep belief networks, stacked autoencoders and, mainly, recurrent neural networks (RNN), proved practical to harvest time dependencies in time series [10]. However, spatial correlations are often neglected by RNNs. New methods based on convolutional neural networks (CNN) were created and, more recently, integrated into RNNs [11]. However, CNN applications are limited to grid structures.

Several forecasting problems have a topology nature as a graph. Graph convolutional networks (GCN) is a variant of a CNN, which operates directly on graphs [12]. This model became part of graph-based forecasting models, such as in spatiotemporal graph convolutional networks [13]. Other neural networks can be combined to improve forecasting performance. For example, *Guo et al.* [14] created a GCN with attention mechanisms. GCN can be combined with RNN, as proposed in [15, 16, 17, 18, 19, 20, 21]. A3T-GCN [17] improves T-GCN [19] by adding an attention layer to re-weight the influence of historical states before the prediction layer. This idea was incorporated in our proposal. GCN and RNN blocks can be arranged in an encoder-decoder structure, as proposed in GC-LSTM [22] and DyGrPr [23]. They project, in the encoding phase, each time-step into a d -dimensional space considering both the dynamic and the structure of the graph. The learned representation is fed to a decoder for prediction. MTGNN [24] and Graph Wavenet [25] are similar, where the former builds inter-variate relationships using a graph-learning module and the latter creates a soft graph where each pair of nodes has a continuous probability of being connected. Instead of a gated temporal convolution, our proposed model uses a convolutional LSTM, which is a similar module as

AGCRN [18]. Our proposal follows a similar idea by learning spatiotemporal features and then feeding the concatenated features into a prediction layer (decoder) mediated by an attention mechanism.

Even the input graph can be pre-processed before feeding a GNN, *e.g.*, creating a motif-based hypergraph based on the original graph [26] or using graph Fourier transform operator to work on spectral domain [27]. Also, uncertainty can be incorporated into GNNs, as shown by *Fu et al.* [28]. Ordinary differential equations (ODE) can be incorporated to graph forecasting by creating a continuous analog to GNN to process both spatial and temporal dependencies using tensor-based ODE [29] or graph neural controlled differential equation [30]. Our proposed model does not use these methods, but they can be incorporated since they are orthogonal to the architecture adopted in this paper. For example, *Deng et al.* [31] proposes a normalization module, which could be incorporated in a future work, specially to deal with the low-frequency versus high-frequency components of the time series. There are also forecasting methods using heterogeneous graphs [32], which is a graph with multiple node and/or edge types. This method predicts values associated with nodes or edges, the structure of the graph (adjacency), or a global value related to the graph. As we focus on homogeneous graphs, we will not elaborate on this.

In general, forecasting models proposed for sales often require more information than is available, such as periodic inventory and waste auditing, and external information (*e.g.*, weather, events). Several forecasting strategies have been proposed in the literature. Graph-based methods stand out for their greater capacity for data representation.

III. REAL-WORLD RESTAURANT SALES DATA

One of the challenges in the fresh food sector is good product demand forecasting, which helps achieve a balance between waste and shortage. The **RESTOS** dataset is a private dataset, which contains the aggregated daily sales by product for some restaurants of a fresh food chain. The time series represents the product sales between December 10, 2018 and March 6, 2022.

Given the historical sales for a certain product in a certain restaurant, we want to predict the sales quantities for this product in this restaurant for the next k days. In addition to the quantities sold, the dataset contains several features used as covariates. A group of them relates to information external to the restaurant, such as weather information (minimum and maximum temperature and daily precipitation volume), the daily number of COVID cases, and bank or school holidays. There is also information about the product, *e.g.*, if it was on sale and on promotion on a particular day. The product graph used for graph-based learning was created using co-purchase similarity, *i.e.*, how likely two products are sold together. Product pairs are matched for all restaurants, and the relative number of occurrences is used as the weight.

The variability in product sales time series lengths poses challenges for demand forecasting, driven by diverse restaurant

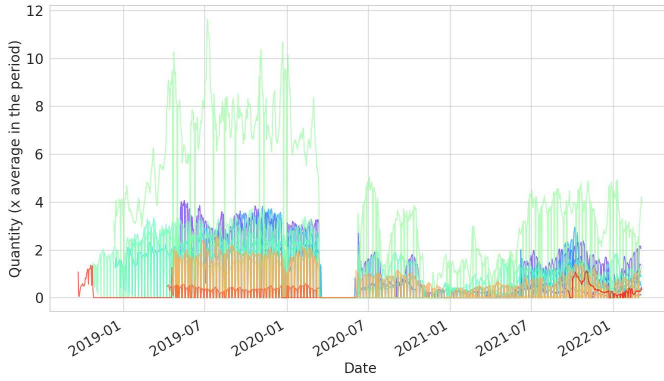


Fig. 1. Example of **RESTOS** time series. Each curve corresponds to the smoothed sales of the same product at different restaurants.

types, locations, and operating hours. COVID-related closures further complicated the data, which we tried to capture by adding a feature indicating non-sales during restaurant closing days. Figure 1 depicts these complexities, with a pandemic-related sales hiatus, varied reopening times, and divergent sales patterns across restaurants. The plot contains the sales of the same product on several restaurants. In Section VI, we delve into these real-world challenges in restaurant sales data analysis.

IV. SPATIOTEMPORAL REGRESSIVE FORECASTING WITH ATTENTION

This section states the problem we are facing in this paper, and describes the architecture proposed to solve it.

Problem definition: In this paper, demand forecasting is used to predict future demand according to historical sales states on restaurants. Since demand is hard to measure, we use sales as a proxy. Generally, demand can refer to previous sales, weather, local events, etc.

a) Definition 1: Sales graph: The topological structure of sales in a restaurant r is described as $G^{(r)} = (V^{(r)}, E^{(r)})$, where $V^{(r)} = \{v_1^{(r)}, v_2^{(r)}, \dots, v_N^{(r)}\}$ is the set of products sold in a restaurant r , and N is the number of products. $E^{(r)}$ is the set of edges, which reflects that both products are sold on the same ticket. The whole connectivity information is stored in the adjacent matrix $\mathbb{R}^{N \times N}$, where rows and columns are indexed by products, and the value of each entry indicates the connectivity between corresponding products. The entry value is 0 if there is no existing link between products and 1 (unweighted graph) or non-negative (weighted graph) if otherwise. In the latter case, the weight can reflect, for example, the probability of both products being sold together.

b) Definition 2: Feature matrix X : The covariates in sales forecasting can be composed (semi-)static features, *e.g.*, the number of cash registers or tables in the restaurant, and dynamic features, *e.g.*, weather information (temperature, rain, etc), local events, if it is a promotion or not, etc. These values can be represented as feature matrix $X \in \mathbb{R}^{T \times N \times F}$, where F is the number of node attribute features, and T is the length of historical time series (input data), *i.e.*, the number

of days/weeks/months used to evaluate the prediction function (train the prediction model).

c) Definition 3: Model: Then, modeling the sales forecasting temporal and spatial dependencies can be viewed as learning a mapping function f on the basis of the sales $G^{(r)}$ and feature matrix $X^{(r)}$ for restaurant r . Sales $Y_{t+1}^{(r)}, \dots, Y_{t+T'}^{(r)}$ of future T' timesteps are calculated by $\left[Y_{t+1}^{(r)}, \dots, Y_{t+T'}^{(r)} \right] = f \left(G_{t-T+1}^{(r)}, \dots, G_t^{(r)}, X_{t-T+1}^{(r)}, \dots, X_t^{(r)} \right)$.

Architecture: We introduce spatiotemporal regressive forecasting with attention (**STeRFA**), consisting of a set of layers that process graph information to provide a prediction as shown in Figure 2. The architecture can be seen as an encoder-decoder, where the first two blocks shown in the figure are the encoder, and the rest corresponds to the decoder. The model takes two required inputs and one optional input. The mandatory inputs are $A_t \in \mathbb{R}^{T \times N \times N}$, which represents the $N \times N$ adjacency matrix of the graph T timesteps (*i.e.*, from time $t-T+1$ up to t) and $X_t \in \mathbb{R}^{T \times N \times F}$, a matrix with the time series values for the F variables used in the forecast. The optional entry is named *key* (explained later in the section).

The first layer processes the spatial information, *i.e.*, the relationships between the nodes. This layer uses a graph neural network operator from Morris et al. [33] named hierarchical local k-GNN that hierarchically combines representations learned at different granularities. We always use 1-GNN with two graphconv layers. Notice that adding the hierarchical component reduces the indistinguishability problem highlighted by Deng et al. [31]. Thus, we shall explore this aspect in future work. The second layer processes temporal information. The temporal layer uses convLSTM proposed by Seo et al. [20], which replaces the Euclidean 2D convolution with the graph convolution. A residual connection transmits to the temporal layer the original information X_t in the graph. It is combined with the output of the spatial layer X_t' , which can be understood as an information enrichment layer for the temporal layer \hat{X}_t' . The output from the first (X_t') and second layers (X_t'') are combined to create an embedding \hat{X}_t'' that contains spatiotemporal information that can be processed by the decoder. Since the literature highlights that attention mechanisms help prediction models to focus on the most relevant information, attention was added to the proposed model to control the input of the decoder layer. The attention layer receives X_t''' as input, which is the same value as \hat{X}_t'' with the temporal T and feature F dimensions combined in a dimension of size F''' . Another input of the attention layer is the key. There are two options: an explicit key k_r provided as input for each graph or, if k_r is not given, the model assumes its value as \hat{X}_t'' (the spatiotemporal embeddings generated by the previous layers). This external key is important in cases where we have graphs that represent different groups of information. The result from the attention feeds a regression layer, which produces the final forecast $Y_t \in \mathbb{R}^{N \times T'}$, *i.e.*, T' days of forecasts (length of the output data) for each of the N nodes (object of interest). This regression layer uses a multi-layer perceptron with two hidden layers of size 512 and ReLU

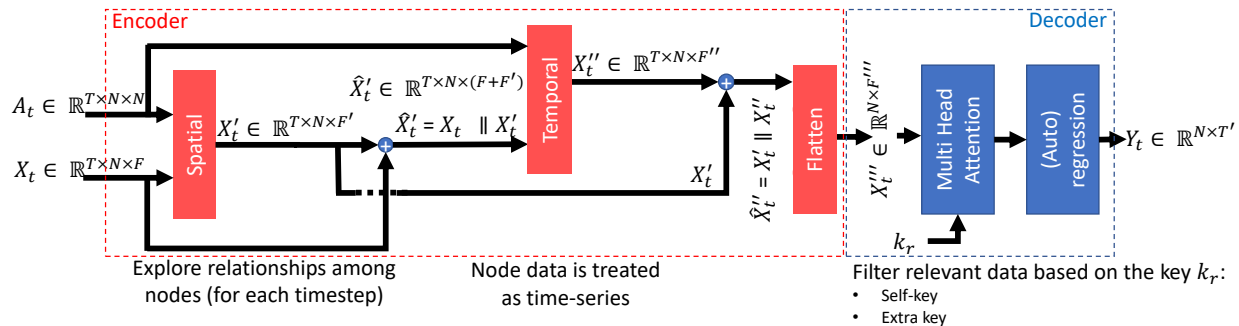


Fig. 2. SpatioTEmporal Regressive Forecasting with Attention (STeRFA)

activation functions, with creates a non-linear combination of the enriched features of different timesteps.

In a restaurant, sales often involve purchase of multiple products simultaneously, influenced by factors like personal preferences, offers, product seasonality, and events. The first two layers of the model reduce problem complexity, enhance forecasting features, and uncover relationships between sales over time, such as seasonal trends. The encoder-decoder structure extracts meaning from spatiotemporal data and encodes predictions, enabling end-to-end training using available inputs and outputs.

V. EXPERIMENTS

This section starts with the description of the experimental setup, followed by the results of predictive performance on three datasets.

A. Experimental setup

We describe the computer resources used in the experiments, the public datasets, training and testing creation, the benchmark models, the hyperparameters, optimizer and learning rate, and the evaluation metric used for evaluation.

1) *Computer environment*: Models were trained and evaluated on a virtual machine with 64 GB of RAM, 2 Intel(R) Xeon(R) Silver 4108 CPU @ 1.80GHz, and one GeForce GTX 1080 Ti. The machine runs Ubuntu 20.04 LTS with Python 3.10 and PyTorch 1.12.1.

2) *Public datasets*: We consider two public datasets for evaluating performance:

1) **Chickenpox dataset**: It contains the weekly number of chickenpox cases reported by general practitioners composed of county-level time series with the disease evolution in Hungary [34]. The time series are joined in a graph with an edge between two counties if they share a geographic border. Data covers the weeks between January 2005 and January 2015. The graph has 20 nodes and 61 edges. The values in the time series are standardized. Input time series of a 4-weeks length are used to predict the next week.

2) **METR-LA dataset**: It contains Los Angeles County highway traffic data obtained by 207 vehicle loop detectors, collected between March 1, 2012 and June 30,

2012. The adjacency matrix of the network graph was created as defined by *Lu et al.* [35]. We divided the data into 5-minute windows, as small traffic flow time intervals are easily affected by outliers, while if the intervals are too large, the data provide little information for prediction [35]. The input length of the time series is 60 minutes (12 data points), and the model is trained to predict the next 60 minutes (the largest range in [35]).

We're testing STeRFA against public datasets to see if it matches SOTA models. Public datasets are simpler than **RESTOS**, which covers multiple restaurants. Public sales datasets, to the best of our knowledge, are relatively basic and lack real sales data or the necessary information for creating graphs.

3) *Train and test dataset*: All the datasets were divided into two non-intersecting parts: train and test sets. The train set is divided as 90% for training and the remaining 10% for validation. The test set contains the most recent value and is not used during training or hyperparameter tuning.

4) *Benchmark models*: As we saw in Section II, there are several forecasting models using graphs. To limit the number of benchmark models, we have selected models listed in the papers that describe the two public datasets. From Rozemberczki et al. [34], we selected three graph-based forecasting models as a benchmark, named DyGrPr [36, 23], DCRNN [21], and STGCN [13]: DyGrPr obtained the best result among the three, DCRNN uses recurrent networks and STGCN uses spectral convolution networks. The latter two models are also evaluated on [35]. In addition, we added a newer variation of STGCN, called ASTGCN [37].

5) *Hyperparameters*: We used Neural Network Intelligence [38] for hyperparameter optimization, as it is an active project with a large option of tuners. The tuner used is Tree-structured Parzen Estimator, which is a lightweight sequential model-based optimization algorithm, with no extra dependency and it supports all search space types [39]. Table I shows the hyperparameters that have been set. We list the parameter name, whether the parameter applies to a specific model or to the optimizer and the data type of the parameter. Finally, the last column lists ranges (values separated by an hyphen) or enumerations (between braces). If a hyperparameter is not listed, it is set to the default value.

TABLE I
HYPERPARAMETER SEARCH GRID USED FOR TUNING THE MODEL

Hyperparameter name	Applies to	Type	Values
Number of hidden channels	STCONV	Integer	1 – 64
Kernel size		Integer	{1, 2, 4, 8, 16}
Order of Chebyshev filter		Integer	1 – 8
Normalization scheme for the graph Laplacian		String	{no, sym, rw}*
Number of GCN blocks	ASTGCN	Integer	{1, 2, 3}
Number of Chebyshev filter		Integer	1 – 8
Order of Chebyshev filter		Integer	1 – 8
Time strides during temporal convolution		Integer	{1, 2}
Normalization scheme for the graph Laplacian		String	{no, sym, rw}*
Number of units	DCRNN	Integer	{1, 2}
Order of Chebyshev filter		Integer	1 – 8
Number of convolutional layers	DyGrPr	Integer	2 – 8
Number of RNN layers		Integer	2 – 4
Aggregation on convolutional layer		String	{add, avg, max}**
Spatial - Learn an additive bias	STeRFA	Bool	{True, False}
Temporal - Number of RNN layers		Integer	2 – 4
Attention - Number of attention heads		Integer	1 – 8
Regression - Hidden layer size		Integer	[512]
Dropout		Regularization	Float
β_1 ***	Optimizer	Float	{0.9, 0.95, 0.99}
β_2		Float	{0.9, 0.95, 0.99}
Weight decay		Float	{0, 0.01}
Learning rate		Float	0.01

NOTES:

* No normalization, symmetric or random-walk normalization

** Reduction after message passing on GCN: addition, average or maximum.

*** More information at <https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html>

6) *Optimizer and learning rate:* Among the large number of optimizers currently available, we chose AdamW [40], which extends stochastic gradient descent and runs repeated cycles of adaptive momentum estimation. Learning rate (LR) controls how quickly or slowly a neural network model learns a problem. In general, a large LR allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. On the other hand, a smaller LR may allow the model to learn more (globally) optimal weights but may take significantly longer to train. We opt to use a decaying strategy that reduces LR, *i.e.*, after 33 epochs LR is divided by 10. The initial learning rate is 0.01. Some optimizer parameters are treated as hyperparameters as shown on Table I.

7) *Evaluation metric:* The root mean square error (RMSE) is used to measure prediction error [41]. RMSE is the standard deviation of the prediction errors (the difference between the actual and the predicted values). It measures how spread out the residuals are, *i.e.*, how concentrated the data is around the line of best fit. RMSE was selected as it has an advantage in exacerbating large errors. Then, minimize RMSE is the goal.

B. Experimental results

This section compares the forecasting performance of our model compared to the benchmark models on all three datasets. In addition, we performed ablation of our model and showed how it affects performance.

1) *Performance on datasets:* The results obtained with the **Chickenpox** dataset using all considered models can be found in Table II. STeRFA shows an improvement of the result of 5.2%.

Table III shows the results obtained with the **METR-LA** dataset. Again, STeRFA outperforms the benchmark models,

TABLE II
PERFORMANCE METRIC DURING TEST RESULTS USING THE **CHICKENPOX** DATASET FOR THE BASELINE AND PROPOSED MODELS.

Model	Time per epoch (in seconds)	RMSE test	Improvement
STeRFA	15.7086	0.8963	—
DyGrPr	12.9048	0.9433	5.2%
STGCN	1.4220	0.9502	6.0%
ASTGCN	2.0000	1.0905	21.7%
DCRNN	142.4904	1.3603	51.8%

with a 9.8% improvement compared to the second ranked model STGCN. However, we highlight that the latter was trained for 50 epochs, while STeRFA was trained for 30 epochs. Thus, STeRFA is 10x slower in order of magnitude. The number of weights to be trained in our model depends on the input data size, and the number of products and features. It is especially affected by the quadratic complexity of the attention mechanisms, which makes the training of STeRFA slower than that of the benchmarks.

TABLE III
PERFORMANCE METRIC DURING TEST RESULTS USING THE **METR-LA** DATASET FOR THE BASELINE AND PROPOSED MODELS.

Model	Time per epoch (in seconds)	RMSE test	Improvement
STeRFA	19320.0999	0.5256	—
STGCN	874.3749	0.5770	9.8%
ASTGCN	73.7699	0.5958	13.4%
DCRNN	7378.9430	0.7077	34.7%
DyGrPr	844.5275	0.7840	49.2%

The graph on the public datasets refer to one county or

an area. **RESTOS** contains several restaurants that sell the same range of products. Thus, they share the same spatial topology (equal adjacency matrix). Our model was trained using three combinations of keys: (a) the self-attention option (X_t''), (b) the restaurant identifier (k_r) and (c) both values concatenated. In (a), a restaurant identifier was added to all nodes on the input graph. Table IV shows the results obtained with the **RESTOS** dataset using the best graph-based models, all trained for 20 epochs. The result shown in the table refers to a model using k_r as key. Using the restaurant identifier drastically improves the forecasting error, which is almost 2.5 times lower than for DyGrPr. A drawback of STeRFA is the training time, which is much higher than the DyGrPr. We discuss this performance difference in section VI-0b.

TABLE IV
PERFORMANCE METRIC DURING TEST RESULTS USING THE **RESTOS**
DATASET FOR THE BASELINE AND PROPOSED MODELS.

Model	Time per epoch (in seconds)	RMSE test	Improvement
STeRFA w/resto key (b)	2069.7797	6.5449	—
STeRFA w/mixed key (c)	1934.9073	11.6237	77.6%
DyGrPr	162.5629	15.8365	142.0%
STeRFA – self key (a)	1934.9073	23.8927	265.1%
DCRNN	8.3123	25.4177	288.4%
ASTGCN	55.8376	25.4504	288.9%
A3TGCN	67.0844	25.6768	292.3%

2) *Model ablation*: We performed an ablation study to investigate how the performance of the model is affected by removing certain components to understand the contribution of that component to the overall performance. This study was performed on the **Chickenpox** dataset only, since it was simpler and, thus, easier to retrain. We tested three configurations of Figure 2: “no ablation”, *i.e.*, the complete model, “no temporal”, where the Temporal layer is replaced by the identity function and “no spatial”, where the Spatial layer is replaced by the identity function. The RMSE are, respectively, 0.8963, 0.9850, and 0.9939. Thus, the removal of layers negatively influences the performance of the model. Since this model works with a time series, it might seem that removing the Temporal layer should affect performance more significantly than removing the Spatial layer. However, it is important to remember that the encoder uses a combination of features at different timestamps, *i.e.*, it allows the creation of an autoregressive model even with the removal of the temporal layer. Thus, we can assume that the function of these two layers is to encode a set of features in a lower dimension that better represents the behavior of the time series, allowing the encoder part of the model to produce a result with a lower RMSE, and, that the spatial encoding cannot be compensated by the encoder as well as it can do for the temporal component.

VI. DISCUSSION ON REAL-WORLD DATA

We assess if GNNs perform well on our private dataset, which hasn’t been used with graph models before. This unique representation offers improved insights into data variability and relationships not captured by traditional methods. We

conduct two assessments: (1) Suitability of the proposed product graph, and (2) Consistency of the model’s forecasts across all restaurants.

a) *Graph structure*: To evaluate how changes to the graph structure affect performance, we propose to change its structure by varying the number of edges and edge weights. We considered three edge configurations: “Fully connected”, where each product is connected to all other products, “Random edges”, where 10% of possible edges are randomly replaced and “Same edges”, which considers the product co-occurrences as described in Section III. For each configuration of edges in the resulting graph, the weights can be altered so that they all have the value 1 or that the value assigned to each edge is randomly chosen between 0 and 1 (inclusive). Recall that the edge weights are used in the graph Laplacian used by the spectral graph convolutional operator. The idea behind this procedure is to confirm (or not) the quality of edges and the assigned weights. If the quality of the proposed graph is good, we expect the errors obtained with the models trained with the variations to be greater than these of the base model.

Using DyGrPr (from Table IV), we trained it with 114 different product graphs generated from the combinations. The best result uses a graph with randomly changed 10% of the edges that received a weight equal to 1 (the other edges remain unchanged) and provides an RMSE equal to 14.7080, *i.e.*, an improvement of 7.7%. Three reasons explain this behavior: (1) the behavior of sales in different restaurants is quite different, thus removing an edge can prevent noise from propagating in the exchange of messages between nodes that occurs during training; (2) some products are not sold in certain restaurants or during certain periods due to commercial decisions for this restaurant, thus removing the edges for other products with active sales and at the same time maintaining this edge for products that are also not sold, facilitates the forecasting process; and (3) the graph is static, *i.e.*, the edges and weights remain the same for all timesteps, and this does not reflect sales decisions for each restaurant (*e.g.*, promotions). These reasons can be minimized by using a distinct graph for each restaurant and/or a dynamic graph (varies with time). Another approach is suggested by Yun et al. [42]. A graph transformer can learn the (static or dynamic) graph from the data. This idea is suitable to our problem since missing/spurious connections in noisy graph results in ineffective (graph) convolution with wrong neighbors, hindering the performance. Our current graph is homogeneous but the available data can be easily converted to heterogeneous form since, *e.g.*, a new set of edges can be created based on similar category, which is now a feature of the node (product). However, we decided on a simpler approach since it is the first time this data is used in such a fashion. Also, the expert opinion of the business partner is that the relation between the products does not change much, which can be perceived by pairing the top-seller products throughout the year. Besides that, a more complex graph network increases the time for the training and forecasting phases. Thus, we will pursue this line of research in future work.

TABLE V

PERFORMANCE DURING TEST RESULTS USING SELECTED RESTAURANTS.

Model	All restos (baseline)	Resto-A		Resto-B	
		RMSE	Improv	RMSE	Improv
STeRFA – self-key	23.8927	11.9722	99.6%	0.2732	8645%
DyGrPr	15.8365	8.0100	97.7%	1.7329	814%

b) Restaurant-specific forecasting: The **RESTOS** results in Table IV are provided by feeding the GNN with data from all restaurants. Thus, generating one model that can forecast sales for all restaurants is the best strategy from a business point of view. Products are identified since one product corresponds to a specific graph node. However, the restaurants are identified by features concatenated to each node embedding. We hypothesized that this set of features may not identify each restaurant well enough, having the opposite effect of injecting noise, affecting the performance. This statement is reinforced by the differences that we observed when we used STeRFA with a key that uniquely identifies the restaurant (best result) and the other options where the key fed to the attention layer is a mix of the key with the result or when no key is presented. To check that, we trained STeRFA with self-key input on the attention layer and DyGrPr with data that comes from only one restaurant. We trained the models on two restaurants with different sales patterns. *Resto-A* is standalone while *Resto-B* is in-store: besides having different customer profiles (*Resto-A* is open to the general public), both restaurants were affected differently by COVID restrictions. Table V shows the results. The rows contain model, identified in the first column. The second column shows the RMSE of the model trained with all restaurants (baseline). The following columns show the RMSE and percent improvement relative to the baseline when trained only with *Resto-A* or *Resto-B*. Both models showed improvement when trained with only one restaurant (on par with or better than the best model in Table IV), meaning these models cannot discriminate well between restaurants when training with the whole dataset, worsening the general result. We argue, therefore, that distinguishing between restaurants is an important aspect to improve the model’s performance as obtained by our model with the key. In addition, from a business perspective, there is a trade-off between the cost of maintaining multiple forecasting models and accuracy. We see that STeRFA with the restaurant id is a good option, as it allows training a single model with good performance. Here, we also observed different results for the two restaurants, which reinforces our observation that the sales of some restaurants are more easily forecasted, implying the need for a better method to identify the restaurants. Improving the identification of restaurants as well as the model’s ability to identify different cycles and trends depending on location are matters for future research.

c) Challenges of real-world data: Figure 3 helps to explain the results obtained for all datasets and models, particularly with STeRFA. The graphs show the individual object of interest (county, sensor, or product) on the X-axis. The Y-axis displays the mean absolute percentage error (MAPE).

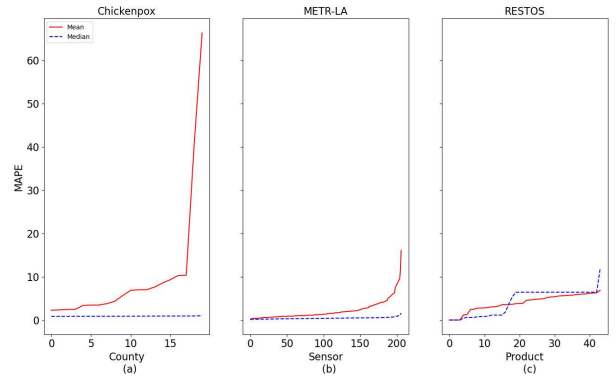


Fig. 3. Distribution of MAPE results with STeRFA on all datasets.

This metric is mostly used for checking forecast accuracy. An optimal forecast will provide zero error. Two curves are shown for each graph: the average (in red) and the median (in blue) MAPE. Figure 3 (c) shows the results for only one restaurant. Comparing the three graphs, we observe that the dashed curve is more horizontal and closer to zero in (a) and (b) than in **RESTOS** (c). This reflects the quality of the forecasts as the RMSE values showed in the previous section, *i.e.*, the model provides better predictions on the two public datasets.

When comparing the patterns of the curves, we see that the first two are similar: for most counties and sensors, the median MAPE is very close to zero. This shows that the model manages to provide good predictions in a large number of cases, with forecasts for some being affected by outliers. However, in Figure 3 (c) the pattern is different. For most products, the median MAPE is rather high. The average MAPE indicates that the results are not necessarily affected by outliers, but rather show that the model has much more difficulty in learning the product behavior through time.

To illustrate this, we present, in Figure 4, the MAPE distribution considering each of the products sold in one of the restaurants in the **RESTOS** dataset with a 1-day forecast period using STeRFA. Each curve corresponds to a product and shows the errors obtained in each of the 1286 sequences of 180 days generated in the dataset. The graph qualitatively presents the model’s difficulty in predicting each of the products. The closer the curve is to zero, the better. As discussed in Section III, the time series length for each product-restaurant pair varies, which implies the imputation of a large volume of data with zeros, thus the same product in each restaurant has a different evolution pattern. Furthermore, the behavior also depends on the customers served by a restaurant, thus, even in periods when two restaurants sell the same product, they show quite different trends and cycles.

d) Cross-correlation: Finally, cross-correlation is a way of measuring the similarity between a time series and a lagged version of another time series [43, 44]. In other words, it can tell us whether a time series is a leading indicator for another time series. An analysis of the distributions of the graph nodes’ time-series pairs can provide a broader view of the reason for

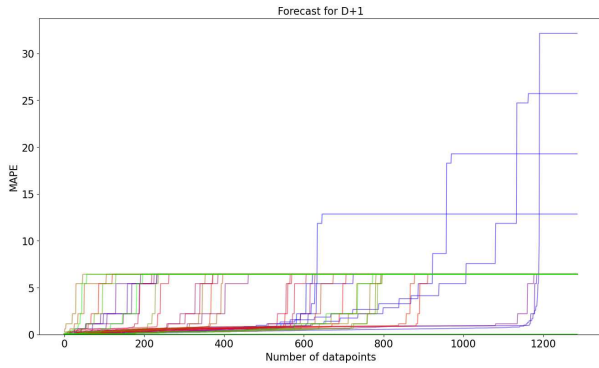


Fig. 4. Distribution of MAPE results with STeRFA for **RESTOS** dataset for all products on one restaurant.

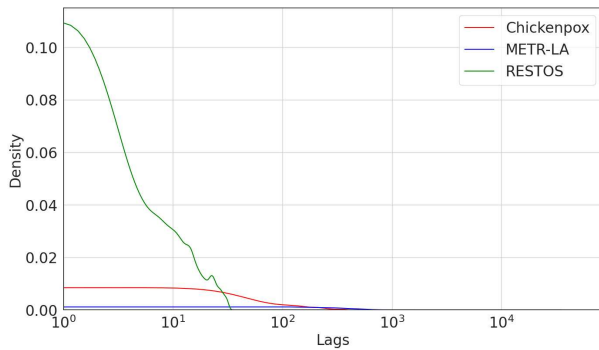


Fig. 5. Distribution of main cross correlation lag on the **Chickenpox**, **METR-LA** and **RESTOS** datasets

the performance of the prediction models in each of the studied datasets.

Figure 5 displays kernel density estimation of cross-correlations between node time series lags on a log scale for a single restaurant. A left peak indicates strong explanatory relationships between time series. As the peak shifts rightward, model complexity increases due to the need to correlate distant data blocks. Larger lag values also add complexity, requiring the GCN to remember older values with fewer training data points. For a single restaurant, within 35 days, all time series pairs correlate, suggesting patterns related to opening hours, promotions, or consumer behavior. This underscores the model’s need to handle shorter sequences compared to the public datasets, emphasizing spatial correlation over temporal.

e) Operational aspects: Training time is a crucial factor in model selection, but it should not be assessed in isolation. Our best model has a longer training time compared to the second-best model. However, improving the latter’s performance needs training multiple instances across different restaurants, resulting in increased OPEX, including technical staff, training time, and computational resources. Model training is a batch and offline process, with the frequency of updates being a business decision that balances cost and forecast quality. None of the tested models was designed for online updating, and the evaluation of performance degradation and update methods is a future research work. Other business con-

siderations include forecast generation time, typically during night hours, and the required computational resources. Our prototype, deployed in a partner’s environment (VM using Intel(R) Xeon(R) E5-2673 1 core @ 2.30GHz server with 4GB RAM and no GPU), provides daily forecasts for 3 restaurants with an average per-restaurant forecast time of 112.075 *ms*. The total execution time per restaurant, including program loading, database queries, forecasting, and central system updates, is 10.27 *s* on average, with queries accounting for most of the time (9.57 *s*). This configuration allows forecasting for over 100 restaurants per hour.

VII. CONCLUSION AND FUTURE WORK

This study explores GCN and RNN models for time series forecasting, introducing **STeRFA**, which leverages attention for spatiotemporal embeddings to capture relevant data automatically. We calibrated and validated our model using publicly available well-studied datasets. Extensive experiments on diverse data confirm our approach’s effectiveness, outperforming baseline models on both public and private datasets. We also delve into the challenges encountered in real-world applications in detail.

The main idea of using a graph approach on **RESTOS** is to capture spatiotemporal relationships that are not tackled by traditional forecasting methods. Most forecasting approaches in the literature are one-dimensional, *i.e.*, the forecast mainly depends on historical sales and other features [45]. These approaches do not consider the effect of other sales. They fail, therefore, to capture the correlations between different restaurants and products in situations such as (i) competition between similar products, (ii) out-of-stock effect on a competing product, (iii) launching of new products/competing offers and (iv) cold start offers or offers with very limited historical sales data. For business efficiency, it’s desirable to maintain a single forecasting model for all restaurants. The proposed model incorporates a restaurant identifier in its pipeline, enabling it to make more accurate predictions compared to baseline models.

Our research suggests room for improving the graph creation process for the **RESTOS** dataset. Currently, a single topology is used, overlooking potential variations in sales patterns among different restaurants and their unique product relationships, which requires further investigation. Additionally, adapting the loss function to business requirements, such as prioritizing $D + 1$ forecasting errors over $D + 5$, could enhance performance. The model shows varying success across different restaurants, hinting at the potential benefits of transitioning from a homogeneous to a heterogeneous graph for richer data representation. We plan to explore this avenue in future research.

Acknowledgements.: This research is funded by the imec ICON project A14Foodlogistics (Agentschap Innoveren en Ondernemen project nr. HBC.2020.3097).

REFERENCES

- [1] R. Fildes, S. Ma, and S. Kolassa, "Retail forecasting: Research and practice," *International Journal of Forecasting*, vol. 38(4), pp. 1283–1318, 2022.
- [2] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [3] W. W. Wei, *Multivariate time series analysis and applications*. John Wiley & Sons, 2018.
- [4] C. K. Chui and G. Chen, *Kalman filtering*. Springer, 2017.
- [5] T.-T. Chen and S.-J. Lee, "A weighted ls-svm based learning system for time series forecasting," *Information Sciences*, vol. 299, pp. 99–116, 2015.
- [6] H. Nie, G. Liu, X. Liu, and Y. Wang, "Hybrid of ARIMA and SVMs for short-term load forecasting," *Energy Procedia*, vol. 16, pp. 1455–1460, 2012.
- [7] Y. Turgut and M. Erdem, "Forecasting of retail produce sales based on XGBoost algorithm," in *Conference on Industrial Engineering and Its Application Areas*. Springer, August 2022, pp. 27–43.
- [8] A. Massaro, A. Panarese, D. Giannone, and A. Galiano, "Augmented Data and XGBoost Improvement for Sales Forecasting in the Large-Scale Retail Sector," *Applied Sciences*, vol. 11(17), p. 7793, 2021.
- [9] Z. Shilong and X. Dairu, "Machine learning model for sales forecasting by using XGBoost," in *ICCECE*. IEEE, 2021, pp. 480–483.
- [10] A. Almalaq and G. Edwards, "A review of deep learning methods applied on load forecasting," in *ICMLA*. IEEE, 2017, pp. 511–516.
- [11] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied soft computing*, vol. 90, p. 106181, 2020.
- [12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv*, 2016.
- [13] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv*, 2017.
- [14] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *AAAI*, vol. 33(1), 2019, pp. 922–929.
- [15] C. Pan, J. Zhu, Z. Kong, H. Shi, and W. Yang, "DC-STGCN: Dual-channel based graph convolutional networks for network traffic forecasting," *Electronics*, vol. 10(9), p. 1014, 2021.
- [16] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Transfer learning with graph neural networks for short-term highway traffic forecasting," in *ICPR*. IEEE, 2021, pp. 10 367–10 374.
- [17] J. Bai, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, and H. Li, "A3T-GCN: Attention temporal graph convolutional network for traffic forecasting," *ISPRS*, vol. 10(7), p. 485, 2021.
- [18] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17 804–17 815, 2020.
- [19] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE T-ITS*, vol. 21(9), pp. 3848–3858, 2019.
- [20] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*. Springer, 2018, pp. 362–373.
- [21] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv*, 2017.
- [22] J. Chen, X. Wang, and X. Xu, "GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction," *Applied Intelligence*, pp. 1–16, 2022.
- [23] A. Taheri and T. Berger-Wolf, "Predictive temporal embedding of dynamic graphs," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 57–64.
- [24] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *ACM SIGKDD*, 2020, pp. 753–763.
- [25] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv*, 2019.
- [26] C. Zhang, S. Zhang, J. James, and S. Yu, "An enhanced motif graph clustering-based deep learning approach for traffic forecasting," in *GLOBECOM*. IEEE, 2020, p. 6.
- [27] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong *et al.*, "Spectral temporal graph neural network for multivariate time-series forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17 766–17 778, 2020.
- [28] J. Fu, W. Zhou, and Z. Chen, "Bayesian spatio-temporal graph convolutional network for traffic forecasting," *arXiv*, 2020.
- [29] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ode networks for traffic flow forecasting," in *ACM SIGKDD*, 2021, pp. 364–373.
- [30] J. Choi, H. Choi, J. Hwang, and N. Park, "Graph neural controlled differential equations for traffic forecasting," in *AAAI*, vol. 36(6), 2022, pp. 6367–6374.
- [31] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, "St-norm: Spatial and temporal normalization for multivariate time series forecasting," in *ACM SIGKDD*, 2021, pp. 269–278.
- [32] Y. Wang, Z. Duan, Y. Huang, H. Xu, J. Feng, and A. Ren, "MTHetGNN: A heterogeneous graph embedding framework for multivariate time series forecasting," *Pattern Recognition Letters*, vol. 153, pp. 151–158, 2022.
- [33] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E.

- Lenssen, G. Rattan, and M. Grohe, “Weisfeiler and leman go neural: Higher-order graph neural networks,” in *AAAI*, vol. 33(1), 2019, pp. 4602–4609.
- [34] B. Rozemberczki, P. Scherer, O. Kiss, R. Sarkar, and T. Ferenci, “Chickenpox cases in Hungary: a benchmark dataset for spatiotemporal signal processing with graph neural networks,” *arXiv*, 2021.
- [35] H. Lu, D. Huang, Y. Song, D. Jiang, T. Zhou, and J. Qin, “St-trafficnet: A spatial-temporal deep learning network for traffic forecasting,” *Electronics*, vol. 9(9), p. 1474, 2020.
- [36] A. Taheri, K. Gimpel, and T. Berger-Wolf, “Learning to represent the evolution of dynamic graphs with recurrent models,” in *WWW conference*, 2019, pp. 301–307.
- [37] J. Zhu, Q. Wang, C. Tao, H. Deng, L. Zhao, and H. Li, “AST-GCN: Attribute-augmented spatiotemporal graph convolutional network for traffic forecasting,” *IEEE Access*, vol. 9, pp. 35 973–35 983, 2021.
- [38] Microsoft, “Neural Network Intelligence,” 1 2021. [Online]. Available: <https://github.com/microsoft/nni>
- [39] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Advances in neural information processing systems*, vol. 24, 2011.
- [40] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv*, 2019.
- [41] L. Wasserman and L. Wasserman, “Models, statistical inference and learning,” *All of Statistics: A Concise Course in Statistical Inference*, pp. 87–96, 2004.
- [42] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, “Graph transformer networks,” *Advances in neural information processing systems*, vol. 32, 2019.
- [43] A. Papoulis, “The Fourier integral and its applications,” *Polytechnic Institute of Brooklyn, McGraw-Hill Book Company Inc.*, 1962.
- [44] R. N. Bracewell, *The Fourier transform and its applications*. McGraw-Hill New York, 1986, vol. 31999.
- [45] A. Gandhi, S. Kaveri, and V. Chaoji, “Spatio-temporal multi-graph networks for demand forecasting in online marketplaces,” in *ECML PKDD*. Springer, 2021, pp. 187–203.