Application of Voronoi Neighborhood Weighted Graph Convolutional Networks for City-Sized Car Traffic Prediction

Przemysław Bielecki[a], Tomasz Hachaj[a,*] and Jarosław Wąs[a]

[a]Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, AGH University of Krakow, Al. Mickiewicza 30, 30-059 Krakow, Poland
ORCID (Przemysław Bielecki): https://orcid.org/0000-0002-4776-3051, ORCID (Tomasz Hachaj): https://orcid.org/0000-0003-1390-9021, ORCID (Jarosław Wąs): https://orcid.org/0000-0003-2964-745X

Abstract. The application of graph convolutional neural networks for traffic prediction is a standard procedure; however, this approach is rarely used under the assumption that the exact city plan is unknown and the prediction area is a city-sized region. This paper fills this gap by proposing and evaluating the Sample and Aggregate-Voronoi method (SAGE-Voronoi), which utilizes the novel concept of Voronoi Neighborhood Weighted Graph-based convolutional networks to predict car traffic in cities. It demonstrates the usefulness of this method for short-time predictions using real sensors data from the moderate-sized city of Darmstadt. We have utilized data from 35 days from March 1, 2024, resampled to 10-minute intervals. During this period, there were 104 crossings with active sensors. The results obtained are compared with those of other neural network algorithms. The proposed approach is not limited to spatiotemporal traffic data and can be utilized in other similar domains. The SAGE-Voronoi graph neural network enables the reliable prediction of varying car traffic among network nodes. It also better fitted the non-typical data in our dataset, showing its better generalization abilities than the basic SAGE network. The source code and dataset used in our experiments are available for download, enabling the results to be fully reproduced.

Keywords: Traffic Prediction, Voronoi Neighborhood, Graph Neural Network, Convolutional Network, Sensor Data, Short-Term, City-Sized data.

1 Introduction

Traffic prediction is one of the most important issues in intelligent transportation systems [16]. Many cities nowadays have sensor-based infrastructure that enables real-time traffic monitoring. Thanks to this high-fidelity data, civil engineers and scientists can visualize actual traffic and perform reasoning and learning from historical data. The contemporary machine learning approaches enable a better understanding of traffic patterns and make short- and long-term forecasts on the future of car traffic in the monitored area [9]. Because of obvious reasons, in urban areas, the motion of the vehicles is constrained by streets, which can be modeled as graph data structures. Therefore, a natural approach to data-driven traffic modeling is to apply graph-based methods such as graph convolutional neural networks.

* Corresponding Author. Email: thachaj@agh.edu.pl

1.1 State-of-the-art

The application of Graph Neural Networks for city traffic forecasting is a straightforward choice [13, 18, 2, 3]. However, it becomes more challenging if there is no detailed information about the route graph. In this particular situation, we need to estimate the connections between nodes representing the points at which traffic is measured. The problem of connectivity between objects whose spatial coordinates are known is often solved by the Voronoi diagram. Therefore, it is unsurprising that the Voronoi-based approach is often used to solve such problems. Over 30 years ago, in 1993, authors in [1] proposed the construction of a Voronoi diagram over a set of points representing patterns in feature space to make it easier to derive alternative neural network structures to achieve the desired pattern classification.

More contemporary approaches use Voronoi schema to produce data structural models for neural networks. For example, these models have been used for protein modeling [14, 12], neighborhood analysis for robotics tactile features related to contact depth [5] and path planning [17], or general-purpose clustering [8]. In paper [7], the spatiotemporal graph convolutional network based on Voronoi diagrams is used for traffic crash prediction. Paper [19] demonstrates the effectiveness of the spatiotemporal-based predictions of the integration of Voronoi tessellations with spatiotemporal deep learning models, such as Long Short-Term Memory (LSTM).

1.2 Novelty of this paper

As can be seen from the state-of-the-art survey, the application of graph convolution neural networks for traffic prediction is a standard procedure; however, it is rarely used when the exact city plan is unknown and the prediction area covers an entire city. This paper fills this gap by proposing and evaluating a method that uses the novel concept of Voronoi Neighborhood Weighted Graph-based convolutional networks for city-scale traffic prediction, more specifically for forecasting traffic volume at intersections. The method is demonstrated to be useful for short-time predictions using real sensor data

from a moderate-sized city. The results obtained are compared with other neural network algorithms. The proposed approach is not limited to spatiotemporal traffic data and can be utilized in other similar domains.

2 Materials and methods

The methodology used for prediction of the car traffic has two components: (i) construction of a Voronoi Neighborhood Weighted Graph (VN-WG) capturing spatial relations between sensors, and (ii) a spatiotemporal neural network combining graph convolution with recurrent modeling. Spatial embeddings are obtained with GraphSAGE, temporal dependencies with an LSTM layer, and the final prediction with a fully connected layer. The following subsections present the GraphSAGE formulation, the VN-WG construction, and the hybrid SAGE-Voronoi model.

2.1 Sample and aggregate network layer

The sample and aggregate method (GraphSAGE), as described in [11] is a reliable and popular method for inductive node embedding. It incorporates node features into the learning algorithm and can learn the topological structure of each node's neighborhood. It can be defined in the following way:

$$SAGE(F, W, A) = ReLU\left(\begin{bmatrix} (F_{(n,m,if)} \times W_{(if,of)})_{(n,m,of)} \\ pin(F_{(n,m,if)}, A) \times W_{(if,of)})_{(n,m,of)} \end{bmatrix}_{(n,m,2\cdot of)}\right) \quad (1)$$

Where: F - input features (tensor of observations), W - weight tensor (trainable parameters), A - adjacency matrix of graph G, n - number of vertices in graph G, m - input sequence length (number of samples in time series), if - input features count (number of features per vertex), of - output features of SAGE count, pin - permutation invariant pooling operator (often maximum, mean or sum). ReLU (Rectified Linear Unit) introduces nonlinearity to the solution.

SAGE layer produces low-dimensional tensor representations for all graph nodes in the form of a tensor with dimensionality $(n, m, 2 \cdot of)$, which is a concatenation of the features tensor F multiplied by the weight tensor W and the features tensor from the specific neighborhood of each node aggregated by permutation invariant pooling pin multiplied by the same weight tensor W. Next, the embedding propagates to a temporal modeling layer, such as LSTM or Gated Recurrent Unit (GRU). The final prediction is formed by a fully connected layer. Various approaches can be used to design a node's neighborhood in the graph. Assuming that we are dealing with real-world spatiotemporal data, the most intuitive approach is either to utilize the known topology of the graph with a distance-based threshold or, if the graph is unknown, model the graph structure only by distances between nodes, as in [20]:

$$a_{ab} = \begin{cases} e^{-\frac{d_{ab}^2}{\sigma^2}} & if \quad a \neq b, e^{-\frac{d_{ab}^2}{\sigma^2}} \geq \epsilon \\ 0 & otherwise \end{cases} \quad (2)$$

Where a_{ab} is a coefficient in the adjacency matrix between nodes indexed a and b, σ and ϵ are domain-specific parameters that depend on the real-world distances. As it can be challenging to rationally estimate the slope of (2) that is guided by σ, the simplified approach is often used:

$$a_{ab} = \begin{cases} 1 & if \quad a \neq b, e^{-\frac{d_{ab}^2}{\sigma^2}} \geq \epsilon \\ 0 & otherwise \end{cases} \quad (3)$$

which produces a binary adjacency matrix. The binary adjacency matrix also simplifies the pin operator in (1) because it does not have to take edge weight into account:

$$pin_a(F, A) = \Theta(F_b \forall A_{ab} \neq 0) \quad (4)$$

where Θ is a permutation invariant pooling function (see explanation below 1).
However, in that approach, we lose some information about graph topology.

2.2 Voronoi Neighborhood Weighted Graph

Definition 1 (Voronoi Neighborhood Weighted Graph)**.** *Let $P = \{p_1, \ldots, p_n\}$ be the set of sensor locations in the plane. Define the Voronoi adjacency graph $G_V = (V, E_V)$, where the set of nodes $V = P$ and the edge $\{p_i, p_j\}$ belongs to E_V if the corresponding Voronoi cells share a boundary. For a parameter $d_{\max} \in \mathbb{N}$, the Voronoi Neighborhood Graph is defined as $G = (V, E)$ with*

$$\{p_i, p_j\} \in E \quad \Leftrightarrow \quad dist_{G_V}\{p_i, p_j\} \leq d_{\max},$$

where $dist_{G_V}$ is the shortest-path length in G_V. A weighted version $G = (V, E, W)$ is obtained by assigning to each edge $\{p_i, p_j\} \in E$ a weight w_{ij} derived from $dist_{G_V}(p_i, p_j)$ using one of the scaling rules (5)–(7).

2.3 Voronoi neighborhood graph calculation

Let us assume that we are registering data at a finite number of points (sensors) with known coordinates, and that the data are time series. Also, let us assume that we anticipate the influence of spatial relations between nodes on time series values, and that the strength of the influence is positively correlated with the proximity between points (sensors). The intuitive approach to model the spatial relationship between these points is to present them in the form of a graph G_V derived from the Voronoi diagram. The nodes of this graph are the sensor locations; an edge connects two points if their Voronoi cells share a boundary.

The Voronoi neighborhood graph G is derived from G_V and has an additional parameter d_{max} - maximal neighborhood size. G consists of all the nodes from G_V. Two nodes of G are connected if there is a path in a graph G_V of length no greater than d_{max}.

In order to calculate graphs G_V and G we can apply Delaunay triangulation because the Delaunay triangulation of a discrete point set corresponds to the dual graph of the Voronoi diagram [6]. The proposed algorithm for calculating G_V and G is presented in Algorithm 1.

Figure 1 illustrates the process of calculating the Voronoi neighborhood. In this image, a Voronoi diagram has been generated from a set of points. To calculate the neighborhood for a certain point indicated in red, we evaluate all cells around it with an increasing diameter. The level of the neighborhood between the red and blue points is color-coded. We repeat this procedure for all points in order to calculate the adjacency matrix of G.

The next step in our approach is to rescale the adjacency matrix A generated by Algorithm 1 so that the farther the path between the nodes in graph G_V is, the smaller the values in A. In other words, the weights of the edges in A should be inversely proportional to the

Algorithm 1 Calculate Voronoi neighborhood graph

Require: P - a set of n points that represent the spatial position of measurements (for example, the position of sensors at road crossings), d_{max} - maximal neighborhood size
$T \leftarrow Delaunay(P)$ ▷ perform Delaunay tessellation, returns data structure T which for each $p_i \in P$ holds information about each $p_j \in P$ that has a common edge
$A \leftarrow [0]_{n \times n}$ ▷ initialize adjacency matrix of size $n \times n$ with zeros for the graph that will be generated for each of the n points in P
procedure CALCULATE_A(i, k, T, A, d, d_{max}) ▷ Calculate the adjacency matrix where i - initial point index, k - neighbor point index, T - Delaunay tessellation structure, A - adjacency matrix, d - actual neighborhood distance.
  for $j \in T[k]$ **do**
    if ($A[i,j] = 0$ or $A[i,j] > d$) and $j \neq i$ **then**
      $A[i,j] \leftarrow A[j,i] \leftarrow d$
    end if
    if $d < d_{max}$ **then**
      Calculate_A($i, j, T, A, d+1, d_{max}$)
    end if
  end for
end procedure
for $p_i \in P$ **do** ▷ Fill adjacency matrix for each $p_i \in P$
  Calculate_A($i, i, T, A, 1, d_{max}$)
end for
return A

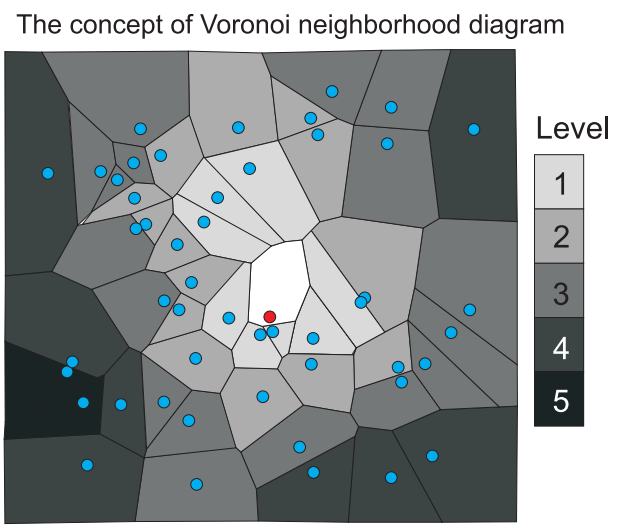


Figure 1: This figure illustrates the concept of calculating the Voronoi neighborhood. The Voronoi diagram is generated from a set of points. In order to calculate the neighborhood for a certain point indicated in red, we evaluate all cells around it with increasing diameter. The level of the neighborhood between the red and blue points is color-coded.

path length between nodes in G_V. In order to achieve this, we can apply one of several possible approaches:

- Linear scaling:

$$a'_{a,b} = \begin{cases} \frac{1}{a_{a,b}} & if \quad a_{a,b} \neq 0 \\ 0 & if \quad a_{a,b} = 0 \end{cases} \tag{5}$$

- Exponential scaling:

$$a'_{a,b} = e^{1-a_{a,b}} \tag{6}$$

- Binary thresholding:

$$a'_{a,b} = \begin{cases} 1 & if \quad a_{a,b} \neq 0 \\ 0 & if \quad a_{a,b} = 0 \end{cases} \tag{7}$$

After applying Algorithm 1 and one of the approaches (5)-(7) we can use the adjacency matrix A' in SAGE layer (1). The pin operator in (1) that takes edge weight into account becomes:

$$pin_a(F, A) = \Theta(F_b \cdot A_{ab}) \tag{8}$$

2.4 *Voronoi neighborhood in graph neural network: SAGE-Voronoi*

The graph convolutional neural network proposed in this paper is composed of a SAGE layer (1) with the pin operator (8) for graph embedding, followed by an LSTM layer for temporal modeling. The final, third layer is the fully connected layer that calculates the network response by performing a linear combination of the LSTM outputs. We will refer to this network later in this paper as SAGE-Voronoi. The loss function used for training is a mean squared error (MSE).

2.5 *Dataset*

To evaluate our approach, we used a city-scale car traffic dataset from Darmstadt, which is available to download from https://datenplatform.darmstadt.de/verkehr/apps/opendata/#/ (Access date April 27, 2025). The data is updated every minute and is provided in CSV format. The collection contains the values of traffic volume values for individual intersections. This reliable dataset was used in previous research [10, 16].

In order to download data, we have used the script available in repository https://github.com/browarsoftware/darmstadt_download (Access date April 27, 2025). We have utilized data from 35 days from March 1, 2024, resampled to 10-minute intervals. During this period, 104 crossings with active sensors were taken into account. We added up the measurements taken by all the sensors at each crossing, so we did not consider the direction of car traffic. As a result, the adjacency matrix A' is symmetric. If, during the 35 days, some sensor did not provide traffic data, we replaced readings with zero (we did not apply any procedure for filling in missing data). We have split the dataset into the train, validation, and test subsets in proportions 0.5, 0.2, and 0.3, respectively, starting from the earliest to the latest time periods. The test and validation data were used during training. Evaluation of method performance was made on the test dataset. Each subset was randomly shuffled using a fixed seed. We repeated our experiments 10 times, changing the seed each time. Each of the three dataset features was standardized by removing the mean and scaling to unit variance.

Figure 2 shows the locations of car crossings on a map of the city of Darmstadt. Red crossings are those discussed in detail in Sections 3 and 4.

3 Results

We have implemented our method using Python 3.8 programming language and the machine learning libraries Tensorflow 2.8, Keras

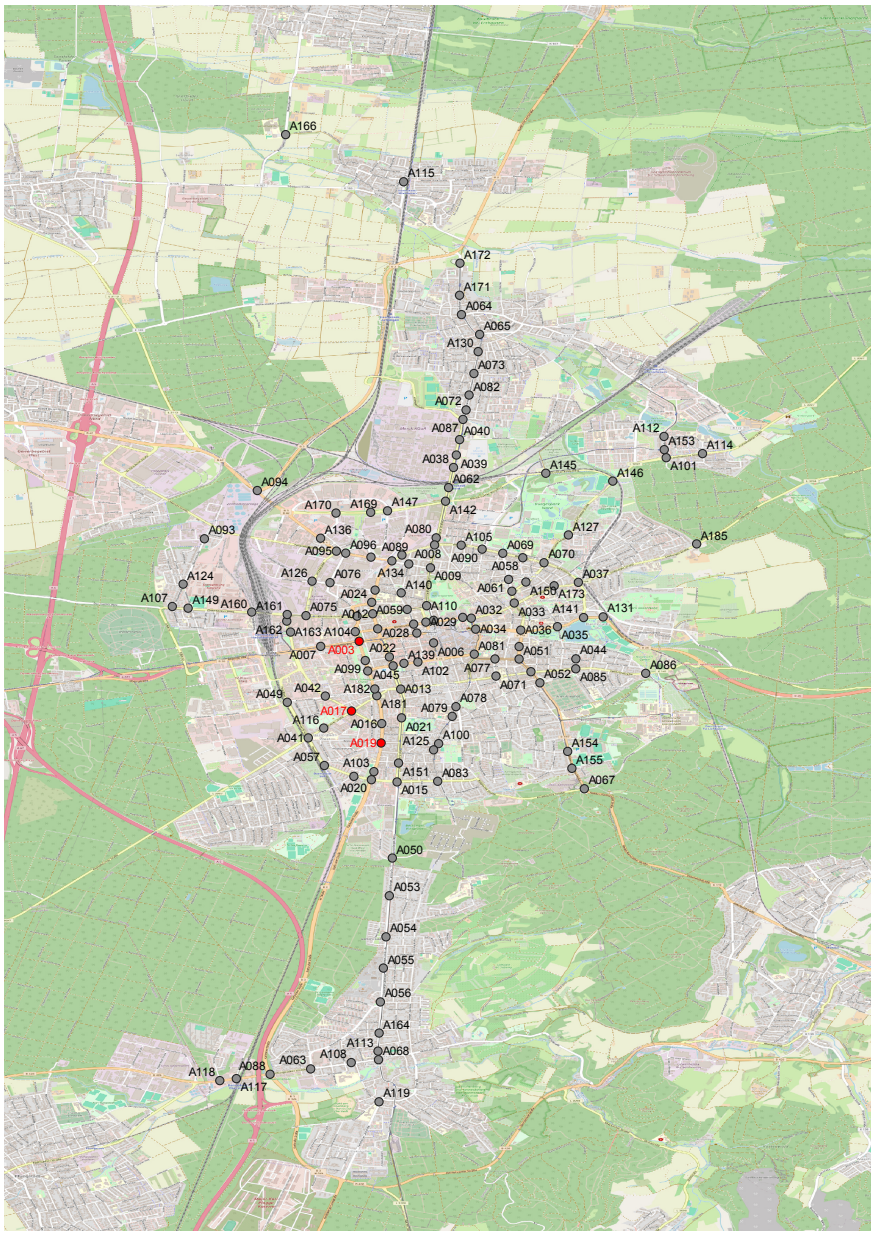


Figure 2: Positions of car crossings on the city map of Darmstadt. The red crossings are those discussed in detail in Sections 3 and 4. Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org [15].

2.8, and Scipy 1.8, in which the Delaunay tessellation is implemented. Our implementation significantly extends the source code https://keras.io/examples/timeseries/timeseries_traffic_forecasting/ (Access date April 27, 2025). The source codes and dataset of our experiments can be downloaded from https://github.com/bielprze/VN-WGCN (Access date April 27, 2025), and the experiments are fully reproducible.

In order to evaluate the proposed SAGE-Voronoi method, we have utilized the dataset described in Section 2.5. We have considered three short-term forecast horizons: 1 sample horizon (10 minutes ahead), 2-sample horizons (20 minutes ahead), and 3-sample horizons (30 minutes ahead). We have tested three adjacency scalers, as defined in equations (5)-(7). The results of the test SAGE-Voronoi network were compared to the original SAGE approach and the simple, Pure LSTM approach. Both the SAGE and SAGE-Voronoi had 64 LSTM units with the length of W in (1) set to 10. The networks were trained using the RMSprop optimizer [4] with a learning rate of 0.0002 for 40 epochs. The maximum neighborhood size d_{max} in Algorithm 1 was set to 5.

A Pure LSTM network consists of an LSTM layer with 200 units, a connected dense layer with 200 units with ReLu activation, and a final dense layer with a size equivalent to the forecast horizon. The network was trained using the Adam optimizer with a learning rate of 0.0001 for 200 epochs. The loss function was mean squared error (MSE). The meta parameters of SAGE-family networks were initially suggested by the creators of the original SAGE implementations, while the Pure LSTM was the result of parameter tuning, which, due to limited space, will not be described in this work.

We have evaluated four error functions: MSE, relative mean squared error (RMSE), mean absolute error (MAE), and mean rel-

Table 1: The evaluation results of the Pure LSTM neural network on the test dataset described in Section 2.5.

Forecast Horizon	Mean MSE	Mean RMSE	Mean MAE	Mean MRE
10 minutes (1 sample)	10792.890	94.784	39.915	0.300
20 minutes (2 samples)	11327.710	98.502	40.619	0.304
30 minutes (3 samples)	11758.250	100.989	41.530	0.314

Table 2: The evaluation results for the SAGE neural network on the test dataset described in Section 2.5.

Forecast Horizon	Mean MSE	Mean RMSE	Mean MAE	Mean MRE
10 minutes (1 sample)	6356.707	70.725	25.044	0.241
20 minutes (2 samples)	7373.572	76.669	27.536	0.261
30 minutes (3 samples)	8201.786	81.129	30.072	0.287

Table 3: The evaluation results of the SAGE-Voronoi neural network on test dataset described in Section 2.5.

Forecast Horizon	Adjacency Scaler	Mean MSE	Mean RMSE	Mean MAE	Mean MRE
10 minutes (1 sample)	(5)	6192.241	69.946	24.644	0.229
10 minutes (1 sample)	(6)	6221.831	70.089	24.820	0.232
10 minutes (1 sample)	(7)	6266.784	70.308	24.666	0.234
20 minutes (2 samples)	(5)	7324.215	76.407	27.389	0.263
20 minutes (2 samples)	(6)	7414.280	76.876	27.619	0.263
20 minutes (2 samples)	(7)	7326.109	76.422	27.373	0.261
30 minutes (3 samples)	(5)	8114.812	80.676	29.743	0.275
30 minutes (3 samples)	(6)	8248.948	81.283	30.106	0.283
30 minutes (3 samples)	(7)	8063.123	80.383	29.499	0.275

ative error (MRE). Results for the Pure LSTM network are presented in Table 1, for the SAGE network in Table 2, and for SAGE-Voronoi in Table 3. All results were averaged over 10 repetitions with different random seeds (see Section 2.5); hence the table headers report "Mean MSE," "Mean RMSE," "Mean MAE," and "Mean MRE." In Figure 3, we present detailed traffic forecast values for three selected crossings that are representative of our dataset. These crossings are A003, A017, and A019.

4 Discussion

As shown in Tables 1-3, SAGE-family graph networks outperform the Pure LSTM architecture. Passing information about the topology of the nodes clearly improves the prediction capability of the SAGE and SAGE-Voronoi architectures. A non-linear fully connected layer in the Pure LSTM approach is insufficient to deduce this information from the training dataset. The Mean MRE prediction of Pure LSTM never dropped below 0.3 while the Mean RMSE was around 0.9 and the Mean MAE around 0.4. It is also worth noting that there is a positive correlation between the values of MSE, RMSE, MAE and MRE.This means that an increase or decrease in one of these metrics is also reflected in an increase or decrease in the others (this is obvious in the case of MSE and RMSE). The MRE metric is especially important because it shows the error rate as a percentage of the actual value. In the case of the Darmstadt dataset, the traffic varies from dozens to thousands of cars per sample, so a relative measure is more appropriate for judging prediction quality. Both networks obtained very similar results in the case of SAGE and SAGE-Voronoi. For a forecast horizon of 10 minutes (1 sample) and 30 minutes (3 samples), the results of SAGE-Voronoi were slightly better for all adjacency scaling methods considered. For one and three samples, prediction Mean MRE in SAGE-Voronoi dropped by 0.012 compared to the SAGE approach while using linear scaling (5). For the 20-minute prediction (2 samples), SAGE-Voronoi has slightly worse results in the case of exponential scaling (6) for all error metrics, while the other two adjacency scaling resulted in slightly better results in all metrics beside Mean MRE which has identical values for (7). Therefore, we can conclude that, in most cases, applying the Voronoi neighbourhood graph improvement proposed in this paper has a positive influence on the prediction of the SAGE graph neural network architecture.

Figure 3 presents a more detailed visualization of the three networks' performance. These three crossings were selected because they exhibit significantly different scales of average movement per unit of time. As can be seen, both SAGE and SAGE-Voronoi perform very similarly; however, tthe quantitative error measures clearly demonstrate the advantage of the Voronoi approach. The Pure LSTM approach is visibly and quantitatively inferior to the other methods. An interesting phenomenon also occurs at the A017 crossing, where there is heavy traffic from April 1 to April 2 (see (c) and the enlarged fragment in (d)). The SAGE-Voronoi was able to predict the anomalous traffic more accurately.

Summarizing the proposed SAGE-Voronoi graph neural network allows reliable prediction of varied car traffic among network nodes. It also better fitted the non-typical data in our dataset, showing its better generalization abilities than the basic SAGE network.

5 Conclusion

The Voronoi Neighborhood allows modeling of real-world scenarios in which the measurement sensors are not uniformly distributed over an area. In the case when those sensors are situated in the road crossings, considering the neighborhood to be distance-based (see equation (8)) may lead to incorrect conclusions about the graph topology. The results indicate that, applying the Voronoi framework significantly improves the predictive ability of a neural network. The

results obtained are promising, and further research will address the full potential of the proposed Voronoi-based framework. There are no methodological obstacles to applying the proposed methodology to non-symmetric graphs. Future work will investigate predictive ability on larger datasets with longer time series. Since the proposed method is not limited to traffic data, it may also be applied and evaluated in other spatiotemporal domains.

References

[1] N. Bose and A. Garga. Neural network design using voronoi diagrams. *IEEE Transactions on Neural Networks*, 4(5):778–787, 1993. doi: 10.1109/72.248455.

[2] S. Cheng, Z. Wang, B. Yang, and K. Nakano. Convolutional neural network-based lane-change strategy via motion image representation for automated and connected vehicles. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):12953–12964, 2024. doi: 10.1109/TNNLS.2023.3265662.

[3] N. Davis, G. Raina, and K. Jagannathan. Grids versus graphs: Partitioning space for improved taxi demand-supply forecasts. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6526–6535, 2021. doi: 10.1109/TITS.2020.2993798.

[4] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, and S. Elmougy. Improving the efficiency of rmsprop optimizer by utilizing nestrove in deep learning. *Scientific reports*, 13(1):8814, 2023.

[5] W. Fan, M. Yang, Y. Xing, N. F. Lepora, and D. Zhang. Tac-vgnn: A voronoi graph neural network for pose-based tactile servoing. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10373–10379, 2023. doi: 10.1109/ICRA48891.2023.10160288.

[6] S. Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean Geometry*, pages 193–233. 1997. doi: 10.1142/9789814355858_0006. URL https://www.worldscientific.com/doi/abs/10.1142/9789814355858_0006.

[7] J. Gan, Q. Yang, D. Zhang, L. Li, X. Qu, and B. Ran. A novel voronoi-based spatio-temporal graph convolutional network for traffic crash prediction considering geographical spatial distributions. *IEEE Transactions on Intelligent Transportation Systems*, 25(12):21723–21736, 2024. doi: 10.1109/TITS.2024.3452275.

[8] C. Gentile and M. Sznaier. An improved voronoi-diagram-based neural net for pattern classification. *IEEE Transactions on Neural Networks*, 12(5):1227–1234, 2001. doi: 10.1109/72.950151.

[9] B. Gomes, J. Coelho, and H. Aidos. A survey on traffic flow prediction and classification. *Intelligent Systems with Applications*, 20:200268, 2023. doi: https://doi.org/10.1016/j.iswa.2023.200268.

[10] Ł. Gosek, F. Muras, P. Michałek, and J. Wąs. Traffic prediction based on modified nagel-schreckenberg model. case study for traffic in the city of darmstadt. In *International Conference on Parallel Processing and Applied Mathematics*, pages 478–488. Springer, 2019. doi: https://doi.org/10.1007/978-3-030-43222-5_42.

[11] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[12] I. Igashov, K. Olechnovič, M. Kadukova, C. Venclovas, and S. Grudinin. Vorocnn: deep convolutional neural network built on 3d voronoi tessellation of protein structures. *Bioinformatics*, 37(16):2332–2339, 02 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab118. URL https://doi.org/10.1093/bioinformatics/btab118.

[13] W. Jiang and J. Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117921. URL https://www.sciencedirect.com/science/article/pii/S0957417422011654.

[14] K. Olechnovič and Č. Venclovas. Voroif-gnn: Voronoi tessellation-derived protein–protein interface assessment using a graph neural network. *Proteins: Structure, Function, and Bioinformatics*, 91(12):1879–1888, 2023.

[15] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

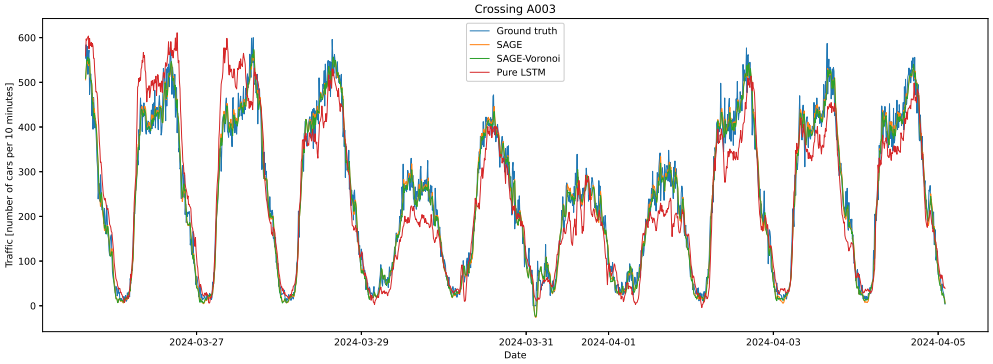
[16] A. Patelli, J. R. Hamilton, V. Lush, and A. Ekart. A gentler approach to urban traffic modelling and prediction. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022. doi: 10.1109/CEC55065.2022.9870273.

[17] F. Qian, W. Liu, H. Bao, and X. Shi. A cnn-based fast generalized voronoi diagrams framework for path planning. In *2024 International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–5, 2024. doi: 10.1109/ICNSC62968.2024.10759920.

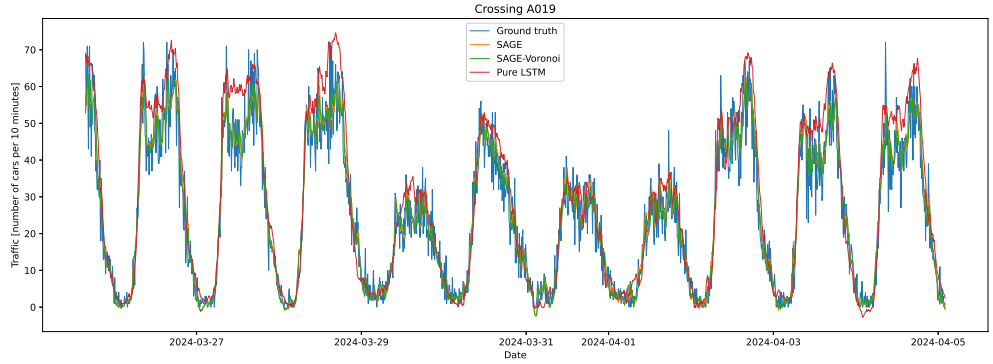
[18] E. Sant'Ana da Silva, H. Pedrini, and A. L. d. Santos. Applying graph neural networks to support decision making on collective intelligent transportation systems. *IEEE Transactions on Network and Service Management*, 20(4):4085–4096, 2023. doi: 10.1109/TNSM.2023.3257993.

[19] H. Wang, H. Zhou, and S. Cheng. Dynamical system prediction from sparse observations using deep neural networks with voronoi tessellation and physics constraint. *Computer Methods in Applied Mechanics and Engineering*, 432:117339, 2024. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2024.117339. URL https://www.sciencedirect.com/science/article/pii/S0045782524005942.

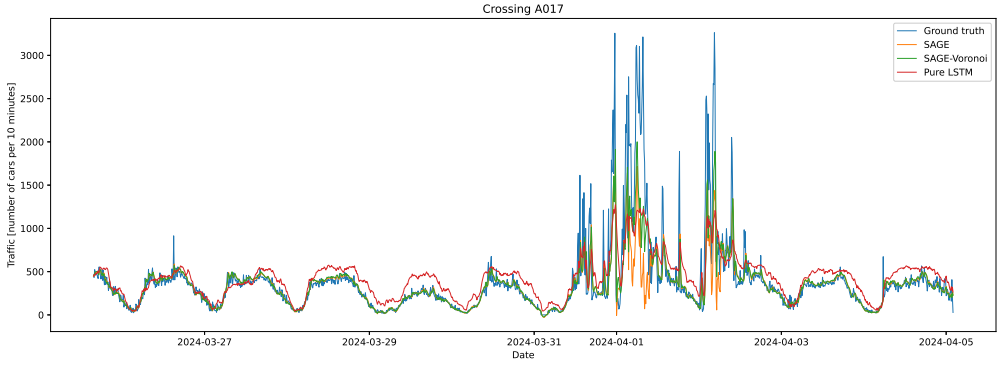
[20] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3634–3640. AAAI Press, 2018. ISBN 9780999241127. URL https://dl.acm.org/doi/10.5555/3304222.3304273.



(a) Errors for SAGE $(850.383, 29.161, 22.221, 0.150)$, SAGE-Voronoi $(809.970, 28.460, 21.580, 0.145)$ and Pure LSTM $(3391.328, 58.235, 43.948, 0.263)$.



(b) Errors for SAGE $(35.861, 5.988, 4.402, 0.331)$, SAGE-Voronoi $(34.095, 5.839, 4.283, 0.325)$ and Pure LSTM $(62.588, 7.911, 5.783, 0.390)$.



(c) Errors for SAGE $(87823.811, 296.350, 109.077, 0.234)$, SAGE-Voronoi $(60322.241, 245.605, 95.932, 0.226)$ and Pure LSTM $(83675.687, 289.267, 165.341, 0.649)$.

(d) Errors for SAGE $(864463.345, 929.765, 678.609, 0.547)$, SAGE-Voronoi $(530029.143, 728.031, 525.195, 0.503)$ and Pure LSTM $(630720.066, 794.178, 582.447, 0.756)$.

Figure 3: The detailed results for the selected crossings. Under each plot, we present mean error values between the original dataset and evaluated networks. The error orders are the same as in Tables 1. The networks are ordered as follows: SAGE, SAGE-Voronoi, Pure LSTM.