ConvLSTM Neural Network based on Hexagonal Inputs for Spatio-Temporal Forecasting of Traffic Velocities

Francisco Bahamondes[1,†], Billy Peralta[1,*,†], Orietta Nicolis[1], Andres Bronfman[1] and Alvaro Soto[2]

[1]Universidad Andres Bello, Facultad de Ingeniería, Santiago, 7500971, Chile
[2]Pontificia Universidad Católica de Chile, Departamento de Ciencias de Computación, Santiago, 7820436, Chile

Abstract
The spatial-temporal prediction of transit speeds is of great importance today as it allows for the anticipation and mitigation of vehicular congestion, thereby improving traffic efficiency. In machine learning, models such as ConvLSTM or Transformers enable reasonable predictions at the spatio-temporal level. However, these models typically assume a square grid configuration, which can limit the use of more convenient configurations in transportation, such as hexagonal grids. We propose a ConvLSTM neural network adapted to hexagonal grid sequences for transit speed prediction, incorporating a transformation of the hexagonal input to allow the use of standard spatial temporal architectures based on square grids. This work validates the proposed model through experiments comparing our approach with baseline methods using traffic data from freight transportation in the Metropolitan Region of Santiago, Chile. The results indicate that using hexagonal sequences improves the mean absolute error (MAE) in predicting freight traffic speeds by 2.7% compared to the base spatio-temporal ConvLSTM prediction model. For future work, we propose using larger databases and adapted transformers.

Keywords
Spatio-temporal prediction, Hexagonal inputs, ConvLSTM, Traffic velocities

1. Introduction

The prediction of transit speeds emerges as a critical component in addressing road congestion, offering a way to anticipate and mitigate real-time setbacks [1], essential for refining the distribution industry and the last mile. By projecting transit speeds at different times and locations, transportation companies can fine-tune the routes of their fleets, minimizing delays and cutting operational costs [2, 3]. This knowledge also enables drivers to make better decisions regarding their itineraries, avoiding bottlenecks and ensuring more agile and effective deliveries. This has a tangible impact on customer satisfaction and overall supply chain efficiency.

In recent years, there has been a notable increase in the application of machine learning (ML) techniques to address traffic speed prediction. Thanks to the availability of real-time data, such as GPS information from vehicles, sensor data, and online traffic, ML algorithms can analyze complex and dynamic patterns in traffic, allowing for more accurate speed predictions. In this problem, classical techniques such as Multiple Linear Regression [4], ARIMA [5], Random Forests [4], Support Vector Machines (SVM) [6], and MLP neural networks [7] have been applied. However, more modern models often utilize deep learning techniques

Conversely, deep learning (DL) models have also been employed for diverse tasks like crowd mobility prediction [8, 9, 10] or traffic prediction [11, 12, 13, 14]. In traffic prediction task, some networks commonly used are Long Short-Term Memory (LSTM) Neural Networks and Gated Recurrent Unit (GRU) networks. These models are ideal for modeling sequential data, such as time series, allowing for efficient capture of both short and long-term dependencies. Although current models are increasingly powerful, they naturally assume a square grid, meaning the information is represented by matrices or tensors. However, in the context of transportation, hexagonal grids offer significant advantages over traditional square inputs, particularly in terms of processing efficiency and accuracy in representing spatial patterns. The hexagonal geometry allows for greater connectivity and uniform coverage of the input space with fewer sampling points, reducing information distortion. This is because each hexagonal point has six equidistant neighbors, unlike the four or eight neighbors in a square grid, which facilitates better data interpolation and a more accurate representation of shapes and patterns. While existing spatio-temporal prediction models can approximate hexagonal inputs, this often results in a loss of

STRL'24: Third International Workshop on Spatio-Temporal Reasoning and Learning, 5 August 2024, Jeju, South Korea
*** Corresponding author.
† These authors contributed equally.
f.bahamondesscholtba@uandresbello.edu (F. Bahamondes); billy.peralta@unab.cl (B. Peralta); orietta.nicolis@unab.cl (O. Nicolis); abronfman@unab.cl (A. Bronfman); asoto@ing.puc.cl (A. Soto)
0000-0002-0877-7063 (F. Bahamondes); 0000-0002-5457-2157 (B. Peralta); 0000-0001-8046-6983 (O. Nicolis); 0000-0002-3122-3237 (A. Bronfman); 0000-0001-9378-397X (A. Soto)

performance.

In this work, we propose processing a sequence of hexagons where each cell contains the traffic speed of vehicles using a specialized library. The adaptation to a standard ConvLSTM network, designed to work with square data, involves transforming the hexagonal data into a compatible square structure. To achieve this, the operations of upsampling, padding, and shifting are applied in series to preserve the original neighborhood of the hexagons in the square structure. Then, a custom kernel is applied to convolve the data and extract relevant features, which allows maintaining the hexagonal structure. The use of hexagonal inputs allows greater efficiency in terms of computation, according to [15], since they require fewer parameters to achieve comparable coverage of the input domain. This can translate into faster training and lower resource consumption.

The contributions of our article are the following:

- We present a hexagonal grid-based representation for spatial-temporal data corresponding to vehicle speeds;
- We conduct comparative experiments that include standard baseline machine learning models along with the technique proposed in this work;
- We make the source code of this work available to facilitate the replicability of experiments.

Section 2 outlines relevant prior work. Section 3 details the proposed methodology. Section 4 presents and discusses the results of our experiments. Lastly, Section 5 summarizes our main conclusions.

2. Related work

Spatial-temporal prediction often use a combination of recurrent and convolutional networks such as ConvLSTM (Convolutional Long Short-Term Memory), which merges the spatial analysis capabilities of CNNs with the ability of LSTMs to capture temporal relationships. Recently, Transformer neural models have been applied [16, 17]. A notable feature of these networks is their ability to model long-range dependencies in sequential data.

In the literature, numerous works are focused on the spatial-temporal prediction of transit speeds, congestion, and transportation using deep neural networks. Lai et al. [18] used an improved ConvLSTM model (eConvLSTM), which incorporates advanced linear features. A Traffic Pattern Attention (TPA) block and a Squeeze-and-Excitation (SE) block are introduced to optimize the accuracy in predicting traffic matrices, thus surpassing existing baseline models. Bogaerts et al. [14] presented Graph CNN-LSTM, a hybrid architecture composed of three components; a CNN, an LSTM neural network, and a FFNN. This structure succeeds in predicting traffic over short temporal horizons (5 minutes) as well as long-term (up to 4 hours) through multi-stage predictions using data provided by the DiDi Chuxing Gaia open data initiative, and demonstrated superiority over cutting-edge ITS algorithms, such as k-NN, SVM, or LSTM. The work of DeepSTCL [19] implements a ConvLSTM network within a deep learning framework for travel demand prediction, standing out for its ability to capture spatial-temporal dynamics and surpass traditional methods like AR and ARIMA. Its focus on analyzing proximity, period, and trend patterns results in more accurate predictions and better interpretation of complex travel demand data, proving its superiority with real data from DIDI in Chengdu. Zhang et al. [20], introduced an LSTM-XGBoost model for short-term traffic flow prediction, addressing challenges such as periodicity and overfitting by combining LSTM with dropout layers and XGBoost to enhance accuracy and generalization. Validated with traffic data from Shenzhen, the model shows significant improvements in accuracy and scalability, highlighting its contribution to optimizing traffic prediction and efficient control. Duan et al. [21], introduced an enhanced hybrid CNN-LSTM model through a *greedy* algorithm for urban traffic flow prediction using GPS data from taxis. This work combines spatial and temporal feature extraction to improve prediction accuracy and efficiency. Validated with data from Xi'an, the model achieves shorter training times and greater accuracy compared to previous methods, offering an effective solution to the complexity of urban traffic data. Xu et al. [22], proposed a spatio-temporal deep learning framework, integrating ConvLSTM and Graph Convolutional Network (GCN), for precise traffic speed prediction. By extracting temporal features with ConvLSTM and spatial features with GCN, the framework significantly improves predictive performance against baseline methods, demonstrating its efficacy in the advanced analysis of large traffic data collected through the Internet of Things (IoT). Hu et al. [23] present the AB-ConvLSTM model, designed to accurately predict large-scale traffic speed in urban road networks. This model combines the ConvLSTM network, an attention mechanism, and Bi-LSTM networks to extract spatial-temporal and periodic features. The results show that AB-ConvLSTM consistently outperforms other models in predicting urban traffic speed, highlighting its ability to capture historical significance and effectively extract daily and weekly periodic functions.

Regarding hexagonal models, they have typically been applied to spatial prediction tasks. Hexagdly [24] facilitates the use of convolutional neural networks (CNNs) in this field without the need for data preprocessing. The main advantage of this approach lies in its adaptation to hexagonal grids through specific convolution and pool-

ing operations, overcoming the limitations of traditional square convolution kernels.

Previous works focus on the combination of different techniques and architectures to improve accuracy and generalization in traffic prediction considering square inputs, while works considering hexagonal inputs propose prediction at a spatial level. In this work, the geometric and topological advantages of hexagonal inputs are exploited [25]. These allow for better coverage and connectivity in capturing the spatial characteristics of traffic, resulting in a more efficient and accurate representation of temporal and spatial dynamics.

3. Proposed method

The general approach to processing sequences of data grid sequences using spatio-temporal neural networks assumes that the data is represented by square grids. However, it is not clear how to apply these models to data represented by hexagonal grids. Particularly, the neighborhood of a cell is different; while a hexagonal cell has six neighbors, a square cell has eight neighbors. However, the use of hexagonal grids in convolutional networks enhances prediction accuracy [26] due to the reduced anisotropy of hexagonal filters [27]. Despite this, the reviewed spatio-temporal neural models do not consider this type of configuration.

In this work, we propose a ConvLSTM-based method for spatial-temporal prediction utilizing hexagonal grids, applied specifically to cargo vehicle speed data. This method comprises three key steps outlined as follows: (i) Initially, we transform the transit speed data onto hexagonal grids represented in Cartesian coordinates. (ii) Subsequently, we sequence the data in hexagonal patterns while preserving the hexagonal constraint by considering equivalent square grids. (iii) Lastly, we employ a ConvLSTM network with a hexagonal constraint (HexConvLSTM) to train on the preprocessed speed data. Now we will detail these steps.

3.1. Cartesian Representation

In this work, we first group the traffic speed data into regular hexagons using a methodology that generates a hexagonal grid. The implementation of this method results in the generation of N regular hexagons, where N is determined by a spatial resolution parameter. This generation produces a hexagonal grid where each hexagon contains the measurements that the area encompasses. In Fig. 1 we show an example of a hexagonal mesh considering 21 hexagons within the experimental region. Fortunately, this hexagonal organization is typically facilitated by specialized libraries; in our case, we used the H3 library from Uber [28]. This library generates a georeferenced hexagonal grid from boundary coordinates, where the number of hexagons depends on the H3 resolution parameter.

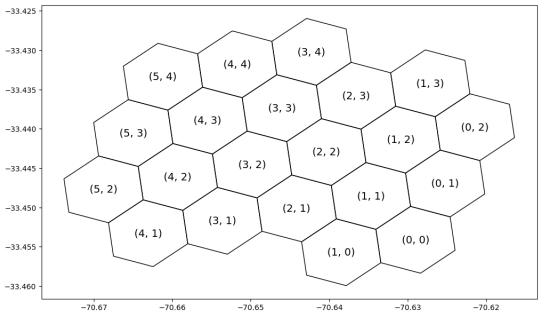
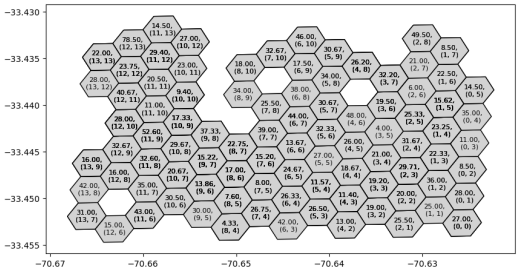
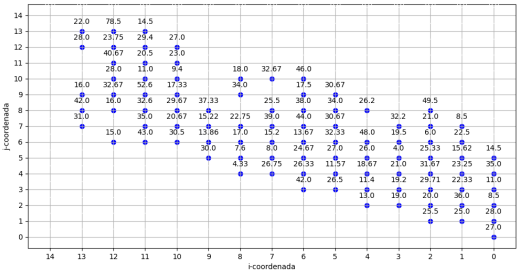


Figure 1: Grid with 21 hexagons covering the area of interest in Santiago, Chile.

Each hexagon is identified by a unique index that encodes its position. When mapping these indices to a Cartesian coordinate system (i, j) for visualization or computational purposes, hexagons sharing a common coordinate will form a line that traverses the grid in a diagonal direction. This is due to the nature of hexagonal packing, where each hexagon touches six others in an arrangement that naturally forms diagonals when represented in a 2D coordinate system (see Fig. 2).



(a) Hexagonal Grid



(b) Cartesian Grid

Figure 2: Hexagonal and Cartesian representation of July 21 at 10:00 a.m.

While in a square grid, a cell typically has eight direct

neighbors (up, down, left, right, and the four diagonals), in a hexagonal grid, each cell is adjacent to six neighbors. Therefore, the hexagonal neighborhood structure significantly alters the spatial distances between cells.

3.2. Square Preprocessing

Given that the data from hexagonal cells are represented as ordered pairs (i, j), the hexagonal grid can be represented as a square grid, that is, in the form of matrices. However, in a square grid, a cell has 8 neighbors, while hexagonal cells have 6 neighbors. Therefore, it is necessary to prepare the data so that a convolution operation, provided by ConvLSTM, respects the hexagonal constraint.

This pre-processing is performed through a sequence of matrix operations involving upsampling, padding, and shifting. This approach results in a representation where it is feasible for a convolution to respect the hexagonal arrangement through a kernel constraint of a ConvLSTM.

3.2.1. UpSampling

The first step in data preprocessing is *UpSampling*. The goal of this operation is to increase the vertical resolution of the matrix by duplicating each row, while keeping the horizontal content unchanged. Assuming that the original matrix $C \times C$ and that the result of upsampling is A', the relationship between the elements of these matrices can be expressed as:

$$A'_{i,j} = A_{\lfloor \frac{i}{2} \rfloor, j} \quad , \forall i \in [1, 2C], \forall j \in [1, C].$$

Visually, if we consider A as the original matrix, then, after applying the *UpSampling* process, A' results as follows:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,C} \\ a_{2,1} & \cdots & a_{2,C} \\ \vdots & \ddots & \vdots \\ a_{C,1} & \cdots & a_{C,C} \end{bmatrix} \quad A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,C} \\ a_{1,1} & \cdots & a_{1,C} \\ a_{2,1} & \cdots & a_{2,C} \\ a_{2,1} & \cdots & a_{2,C} \\ \vdots & \ddots & \vdots \\ a_{C,1} & \cdots & a_{C,C} \\ a_{C,1} & \cdots & a_{C,C} \end{bmatrix}.$$

In this example, each row of A has been duplicated in A' resulting in a matrix of $2C \times C$.

3.2.2. Padding

The second step, *Padding*, adds additional rows to the matrix to prepare the data for the *Shifting* process, which requires a specific number of rows to operate correctly. Specifically, this step adds C rows of zeros at the bottom of A', resulting in a new matrix A'' with size $(2C+C) \times C$. The transformation from A' to A'' can be described as follows:

$$A''_{i,j} = \begin{cases} A'_{i,j}, & \text{if } 1 \leq i \leq 2C \\ 0, & \text{if } 2C < i \leq 2C + C \end{cases}, \quad \forall j \in [1, C],$$

This equation specifies how C rows of zeros are added at the bottom of A'.

Visually, we can see that while A' is a $2C \times C$ matrix resulting from the *UpSampling* process, the result of the *Padding*, A'', will be visualized with the last C rows composed of zeros,

$$A'' = \begin{bmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,C} \\ a'_{1,1} & a'_{1,3} & \cdots & a'_{1,C} \\ \vdots & \vdots & \ddots & \vdots \\ a'_{C,1} & a'_{C,2} & \cdots & a'_{C,C} \\ a'_{C,1} & a'_{C,2} & \cdots & a'_{C,C} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

In this matrix A'', the elements $a'_{i,j}$ represent the values of A', and the last C rows are zeros, creating a final matrix of $(2C + C) \times C$. This adjustment in the padding process ensures that the extended matrix has the appropriate size for the Shifting operation.

3.2.3. Shifting

The final step in the preprocessing is the *Shifting*, which shifts each column of the matrix upwards by a number of positions equal to the column index. This procedure introduces a shift that depends on the column position, achieving the necessary configuration to apply the hexagonal constraint kernel. For the matrix A'', the resulting matrix A''' is obtained as follows:

$$A'''_{i,j} = A''_{(i+j) \mod 2C, j}, \quad \forall i \in [1, 2C], \forall j \in [1, C],$$

where i is the row index and j is the column index in A''. The *mod* operation ensures that the shifting is cyclic, allowing elements that exceed the bottom of the matrix to re-enter from the top.

Considering that A'' is a $(2C + C) \times C$ matrix, the shifting will result in a matrix where each column has been shifted downwards based on its column index. Here is a simplified representation to illustrate the effect on some columns:

$$A'' = \begin{bmatrix} a''_{1,1} & a''_{1,2} & \cdots \\ \vdots & \vdots & \ddots \\ a''_{C,1} & a''_{C,2} & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \\ 0 & 0 & \cdots \end{bmatrix} \quad A''' = \begin{bmatrix} 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \\ a''_{1,1} & a''_{2,2} & \cdots \\ a''_{2,1} & a''_{3,2} & \cdots \\ \vdots & \vdots & \ddots \\ a''_{C,1} & 0 & \cdots \end{bmatrix}.$$

This pattern demonstrates how the elements of each column shift upwards, and those exceeding the upper limit of the matrix reappear at the bottom. The outcome of this final step enables the use of a kernel constraint in *any* ConvLSTM neural network implementation, ensuring strict adherence to the original hexagonal grid neighborhood.

3.3. HexConvLSTM Architecture

Assuming that the data were preprocessed into a square grid according to 3.2, we propose using a ConvLSTM neural network with a kernel constraint. Next, we will describe the kernel constraint mask that enables adherence to the hexagonal arrangement in the grid, followed by the neural network used.

3.3.1. Kernel constraint

The kernel constraint is defined by a binary mask given by:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} & \text{Ne(P)} & \\ \text{Ne(P)} & & \text{Ne(P)} \\ & \text{P} & \\ \text{Ne(P)} & & \text{Ne(P)} \\ & \text{Ne(P)} & \end{bmatrix}.$$

In this matrix, the positions where there is a 1 indicate the cells that will be active, allowing convolution at those specific positions; otherwise, the cells are not processed. The positions are represented in the left matrix, *P* is the target cell and *Ne(P)* is the neighbor of target *P*. In this way, the 6-neighborhood of a hexagonal cell is recovered in the square grid when using standard convolution operations.

3.3.2. The HexConvLSTM network

By introducing the kernel constraint, mentioned in the previous subsection 3.3.1, into a standard *ConvLSTM-2D* layer, we can recover the hexagonal neighborhood in a matrix tensor. We refer to this network as **HexConvLSTM**, where a diagram of it can be seen in Fig. 3. The variables and parameters of the ConvLSTM network are typically well-known and are detailed in [29]. The difference from a standard ConvLSTM lies in the application of the kernel constraint, which allows the network to consider only the neighbors provided by the original hexagonal configuration.

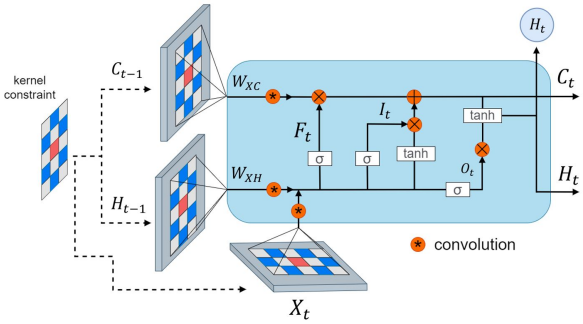


Figure 3: HexConvLSTM block at time step t. The variables are detailed in [29].

In a nutshell, our proposal entails representing a hexagonal grid in a Cartesian representation (see Section 3.1), preprocessing to preserve hexagonal neighborhood (see Section, 3.2), and ultimately applying a ConvLSTM neural network (see Section 3.3). Subsequent experiments aim to evaluate the efficacy of our approach on a real dataset.

4. Experiments

4.1. Data

The database used in this work corresponds to data extracted from the Transportation and Logistics Center of Andrés Bello University, a center dedicated to researching routing problems, last-mile, logistics optimization, among others. The raw data includes 22 million GPS measurements of last-mile cargo vehicle speeds in Santiago, Chile, Metropolitan Region. This data contains the following information:

Table 1
Data description

Term	Description
Id	Measurement identifier.
Date_time	Date and time of the measurement.
Lat	Latitude.
Lon	Longitude.
Speed	Measured speed.
Dir	Vehicle direction (in degrees).
Vehicle_code	Vehicle identifier.

For this work, we have limited our data to a specific subregion (see Fig. 4), considering a particular area with the highest data density in the city of Santiago de Chile,

the capital of Chile. A high data density is considered to minimize missing data, since cargo vehicles tend to prefer certain streets. The boundaries of the chosen area are between latitudes -33.4331 and -33.4524, and longitudes -70.6253 and -70.6655, forming a rectangle that includes the Santiago Centro commune and parts of its neighboring communes.

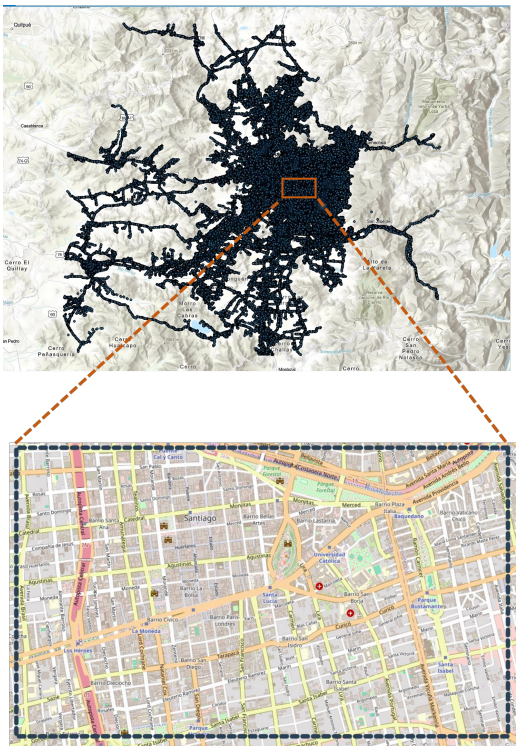


Figure 4: The upper image corresponds to the city of Santiago, while the lower image corresponds to the study area.

The measurements for this subregion span from January 4th to July 25th, 2020. All measurements recording a speed of zero were removed, indicating that the vehicle was stopped or out of operation. Additionally, records outside the time range of 8:00 a.m. to 7:00 p.m. were excluded, as this interval has the highest concentration of measurements. Measurements outside this range were excluded due to their low frequency. Similarly, measurements from Sundays were discarded as they also showed similarly low frequency. It should be noted that there were no measurements during the month of April during the measurement period.

Regarding temporality, the measurements will be treated as hourly time series, which can be divided into 157 days, with each day having 12 hours of measurement (from 8:00 a.m. to 7:00 p.m.), resulting in a total of 1,884 time series. Each of these intervals will be treated as a grid with values imputed according to Section 4.2.

In terms of experimental design, the HoldOut method for time series [30] was followed, where data were sequentially divided into training (70%), validation (15%), and testing (15%) sets, with MinMax scaling applied to each set. All methods were evaluated considering mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and coefficient of determination (R2). Furthermore, to ensure replicability, the demo source code for this work is available at: https://github.com/Francisco0178/HexConvLSTM. At this point, we state that our method is generic, and in future work we will test it on public datasets [31].

4.2. Data Imputation

Our dataset consists of a time series with 1,884 temporal steps, each representing one hour between 8:00 a.m. and 7:00 p.m. over 157 days. Using a fixed grid of 110 hexagonal cells, each time step contains average traffic speed information for each hexagonal cell. These 110 cells are derived from the hexagonal preprocessing described in Section 3.1 using the H3 library at a resolution of 9.

However, it is worth noting that the data originates from geolocated sensor data of cargo vehicles. Upon analyzing this data, it becomes apparent that these vehicles tend to favor certain routes and schedules, resulting in some regions being underrepresented in the data. For instance, at 8 am, the few vehicles that do transit may predominantly utilize main roads, leaving certain areas unmeasured. Consequently, in the utilized representation, there are hexagonal cells with missing measurement information, with the percentage of missing data depending on the H3 resolution parameter.

In our implementation using the H3 library, we opted for an H3 resolution of 9, which generates 110 hexagons. When represented in a square format, it yields 15x15 matrices (225 cells) with 54% missing data. Although this is a high percentage of missing values, using the next H3 resolution, 10, results in grids of 5x6, which are too small for the use of convolutional models; however, using an H3 resolution of 8 results in 500 hexagons leading to 90% missing data, which complicates the training of neural models

In this study, various imputation methods were experimented with, and we found experimentally that the PPCA method performs better than Gaussian-based or MICE imputations. It is worth noting that in [32], PPCA also emerges as a competitive imputation model for traffic prediction tasks.

4.3. Experimental Results

Comparative experiments were conducted among an MLP network, GRU, LSTM, ConvLSTM, and our HexConvLSTM network. The MLP network comprises two

layers with 256 and 128 neurons, while the LSTM and GRU networks consider 128 and 50 recurrent units, respectively. For the ConvLSTM and HexConvLSTM networks, 128 ConvLSTM units are employed. In all neural networks, Mean Squared Error (MSE) was utilized as the loss function.

In the first experiment, we trained the networks using data imputed by the three methods described in Section 4.2. The second experiment involved training the models with data imputed using the method that yielded the best results, but with a reshaping of the time series. This reshaping involved grouping the averages of two consecutive hourly periods, which resulted in halving the total dimension of our time series.

4.3.1. One-Hour Granularity Experiment

Table 2 shows that the proposed HexConvLSTM network achieved the best values across all metrics, surpassing ConvLSTM with relative improvements of 1.3%, 1.3%, 0.7%, and 0.9% in MAE, MSE, RMSE, and R2 respectively. This indicates that the hexagonal constraint better captures the dynamics between the cells, leading to improved performance of a ConvLSTM network. However, when comparing all models, HexConvLSTM yielded the best results, outperforming its closest competitor, MLP. We believe this model performs well due to the low resolution of the 15x15 grid. The competitiveness of MLP on small images, such as on the MNIST dataset, is shown in [33]. However, in the context of transportation in large cities we need to increase the size of the grids to improve the spatial resolution of prediction.

Table 2
Results from one-hour granularity data on test set

Red	MAE	MSE	RMSE	R^2
GRU	7.071	119.696	10.941	0.456
MLP	6.988	118.788	10.899	0.460
LSTM	7.207	121.019	11.001	0.450
ConvLSTM	7.032	118.883	10.903	0.459
HexConvLSTM	**6.973**	**118.048**	**10.865**	**0.463**

4.3.2. Two-Hour Granularity Experiment

Another experiment involved aggregating our data into the average of 2 consecutive time steps, resulting in sequences that still contain 12 steps, but now each step represents aggregated information from 2 consecutive days (6 steps per day), instead of one day per step. This grouping approach effectively reduces the temporal resolution of our data but enriches each time step with a more integrated view of temporal features.

Table 3
Results from two-hour granularity data on test set

Red	MAE	MSE	RMSE	R^2
GRU	5.522	64.784	8.049	0.606
MLP	5.466	63.93	7.996	0.612
LSTM	5.713	67.852	8.237	0.588
ConvLSTM	5.573	65.564	8.097	0.602
HexConvLSTM	**5.419**	**62.674**	**7.917**	**0.619**

Table 3 presents the results of each tested method. The HexConvLSTM network has once again achieved the best values across all metrics, surpassing ConvLSTM with relative improvements of 2.7%, 1.3%, 0.7%, and 2.8% in MAE, MSE, RMSE, and R2, respectively. This reaffirms that the hexagonal constraint effectively captures the dynamics between the cells. Moreover, the results are globally better than those from the one-hour granularity due to less variability since two-hour averages are considered, which appear to be more predictable for all models in general. In this experiment, HexConvLSTM further increases its advantage over the other models.

5. Conclusions

This work demonstrates that the proposed HexConvLSTM model outperforms ConvLSTM across all metrics, indicating superior capture of transit dynamics. It consistently shows an advantage in all metrics, and this advantage is expected to increase as larger grids and longer temporal intervals are used in the sequence of input grids.

The temporal grouping experiment shed light on another critical aspect: efficiency in data representation can be as crucial as the quality of the data itself. In this context, HexConvLSTM not only handled the imputed data well but also benefited significantly from the grouping, enhancing its predictive capacity. This result underscores how HexConvLSTM can extract value from adjustments in data preparation, a considerable advantage for any practical application.

As future work, we plan to use databases with more records, include larger study regions, and incorporate self-attention layers to improve the model's performance.

Acknowledgments

B. Peralta and A. Soto appreciate the support of the National Center for Artificial Intelligence CENIA FB210017, Basal ANID.

References

[1] N. Kumar, M. Raubal, Applications of deep learning in congestion detection, prediction and alleviation: A survey, Transportation Research Part C: Emerging Technologies 133 (2021) 103432.

[2] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 12 (2011) 1624–1639.

[3] M. Shaygan, C. Meese, W. Li, X. G. Zhao, M. Nejad, Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities, Transportation Research Part C: Emerging Technologies 145 (2022) 103921. URL: https://www.sciencedirect.com/science/article/pii/S0968090X22003345. doi:https://doi.org/10.1016/j.trc.2022.103921.

[4] C. Bratsas, K. Koupidis, J.-M. Salanova, K. Giannakopoulos, A. Kaloudis, G. Aifadopoulou, A comparison of machine learning methods for the prediction of traffic speed in urban places, Sustainability 12 (2019) 142.

[5] H. Wang, L. Liu, S. Dong, Z. Qian, H. Wei, A novel work zone short-term vehicle-type specific traffic speed prediction model through the hybrid emd–arima framework, Transportmetrica B: Transport Dynamics 4 (2016) 159–186.

[6] F.-H. Tseng, J.-H. Hsueh, C.-W. Tseng, Y.-T. Yang, H.-C. Chao, L.-D. Chou, Congestion prediction with big data for real-time highway traffic, IEEE Access 6 (2018) 57311–57323. doi:10.1109/ACCESS.2018.2873569.

[7] X. Yang, S. Luo, K. Gao, T. Qiao, X. Chen, et al., Application of data science technologies in intelligent prediction of traffic congestion, Journal of Advanced Transportation 2019 (2019).

[8] R. Jiang, X. Song, D. Huang, X. Song, T. Xia, Z. Cai, Z. Wang, K.-S. Kim, R. Shibasaki, Deepurbanevent: A system for predicting citywide crowd dynamics at big events, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2114–2122.

[9] R. Jiang, Z. Cai, Z. Wang, C. Yang, Z. Fan, Q. Chen, K. Tsubouchi, X. Song, R. Shibasaki, Deepcrowd: A deep model for large-scale citywide crowd density and flow prediction, IEEE Transactions on Knowledge and Data Engineering 35 (2021) 276–290.

[10] R. Jiang, Z. Cai, Z. Wang, C. Yang, Z. Fan, Q. Chen, X. Song, R. Shibasaki, Predicting citywide crowd dynamics at big events: A deep learning system, ACM Transactions on Intelligent Systems and Technology (TIST) 13 (2022) 1–24.

[11] S. Sun, J. Chen, J. Sun, Traffic congestion prediction based on gps trajectory data, International Journal of Distributed Sensor Networks 15 (2019) 1550147719847440.

[12] R. Ke, W. Li, Z. Cui, Y. Wang, Two-stream multi-channel convolutional neural network (TM-CNN) for multi-lane traffic speed prediction considering traffic volume impact, CoRR abs/1903.01678 (2019). URL: http://arxiv.org/abs/1903.01678. arXiv:1903.01678.

[13] Q. Liu, B. Wang, Y. Zhu, Short-term traffic speed forecasting based on attention convolutional neural network for arterials, Computer-Aided Civil and Infrastructure Engineering 33 (2018) 999–1016.

[14] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, P. Hellinckx, A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data, Transportation Research Part C: Emerging Technologies 112 (2020) 62–77. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19309349. doi:https://doi.org/10.1016/j.trc.2020.01.010.

[15] T. Schlosser, M. Friedrich, D. Kowerko, Hexagonal image processing in the context of machine learning: Conception of a biologically inspired hexagonal deep learning framework, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1866–1873.

[16] L. Wu, Y. qing Wang, J. bei Liu, D. hui Shan, Developing a time-series speed prediction model using transformer networks for freeway interchange areas, Computers and Electrical Engineering 110 (2023) 108860. URL: https://www.sciencedirect.com/science/article/pii/S0045790623002847. doi:https://doi.org/10.1016/j.compeleceng.2023.108860.

[17] X. Wang, R. Zeng, F. Zou, L. Liao, F. Huang, Sttf: An efficient transformer model for traffic congestion prediction, International Journal of Computational Intelligence Systems 16 (2023). doi:10.1007/s44196-022-00177-3.

[18] J. Lai, Z. Chen, J. Zhu, W. Ma, L. Gan, S. Xie, G. Li, Deep learning based traffic prediction method for digital twin network, Cognitive Computation (2023) 1–19.

[19] D. Wang, Y. Yang, S. Ning, Deepstcl: A deep spatio-temporal convlstm for travel demand prediction, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[20] X. Zhang, Q. Zhang, Short-term traffic flow prediction based on lstm-xgboost combination model., CMES-Computer Modeling in Engineering & Sciences 125 (2020).

[21] Z. Duan, Y. Yang, K. Zhang, Y. Ni, S. Bajgain, Improved deep hybrid networks for urban traffic flow prediction using trajectory data, Ieee Access 6 (2018) 31820–31827.

[22] F. Dai, P. Huang, X. Xu, L. Qi, M. R. Khosravi, Spatio-temporal deep learning framework for traffic speed forecasting in iot, IEEE Internet of Things Magazine 3 (2020) 66–69.

[23] X. Hu, T. Liu, X. Hao, C. Lin, Attention-based conv-lstm and bi-lstm networks for large-scale traffic speed prediction, The Journal of Supercomputing 78 (2022) 12686–12709.

[24] C. Steppa, T. L. Holch, Hexagdly—processing hexagonally sampled data with cnns in pytorch, SoftwareX 9 (2019) 193–198.

[25] S. Fadaei, A. Rashno, A framework for hexagonal image processing using hexagonal pixel-perfect approximations in subpixel resolution, IEEE Transactions on image processing 30 (2021) 4555–4570.

[26] Y. Zhao, Q. Ke, F. Korn, J. Qi, R. Zhang, Hexcnn: A framework for native hexagonal convolutional neural networks, in: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1424–1429.

[27] E. Hoogeboom, J. W. Peters, T. S. Cohen, M. Welling, Hexaconv, arXiv preprint arXiv:1803.02108 (2018).

[28] I. Brodsky, H3: Uber's hexagonal hierarchical spatial index, https://eng.uber.com/h3/, 2018. Available from Uber Engineering website. Accessed: 22 June 2019.

[29] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, Advances in neural information processing systems 28 (2015).

[30] V. Cerqueira, L. Torgo, I. Mozetič, Evaluating time series forecasting models: An empirical study on performance estimation methods, Machine Learning 109 (2020) 1997–2028.

[31] R. Jiang, D. Yin, Z. Wang, Y. Wang, J. Deng, H. Liu, Z. Cai, J. Deng, X. Song, R. Shibasaki, Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction, in: Proceedings of the 30th ACM international conference on information & knowledge management, 2021, pp. 4515–4525.

[32] T. Sun, S. Zhu, R. Hao, B. Sun, J. Xie, Traffic missing data imputation: a selective overview of temporal theories and algorithms, Mathematics 10 (2022) 2544.

[33] A. Baldominos, Y. Saez, P. Isasi, A survey of handwritten character recognition with mnist and emnist, Applied Sciences 9 (2019) 3169.