Scaling Deep Learning Models for Large Spatial Time-Series Forecasting

Zainab Abbas*, Jon Reginbald Ivarsson*†, Ahmad Al-Shishtawy†, Vladimir Vlassov*
*KTH Royal Institute of Technology †RISE Research Institutes of Sweden
*{zainabab, vladv} @ kth.se *†mail@reginbald.com †ahmad.al-shishtawy@ri.se

***Abstract*—Neural networks are used for different machine learning tasks, such as spatial time-series forecasting. Accurate modelling of a large and complex system requires large datasets to train a deep neural network that causes a challenge of scale as training the network and serving the model are computationally and memory intensive. One example of a complex system that produces a large number of spatial time-series is a large road sensor infrastructure deployed for traffic monitoring. The goal of this work is twofold: 1) To model large amount of spatial time-series from road sensors; 2) To address the scalability problem in a real-life task of large-scale road traffic prediction which is an important part of an Intelligent Transportation System.**

We propose a partitioning technique to tackle the scalability problem that enables parallelism in both training and prediction: 1) We represent the sensor system as a directed weighted graph based on the road structure, which reflects dependencies between sensor readings, and weighted by sensor readings and inter-sensor distances; 2) We propose an algorithm to automatically partition the graph taking into account dependencies between spatial time-series from sensors; 3) We use the generated sensor graph partitions to train a prediction model per partition. Our experimental results on traffic density prediction using Long Short-Term Memory (LSTM) Neural Networks show that the partitioning-based models take 2x, if run sequentially, and 12x, if run in parallel, less training time, and 20x less prediction time compared to the unpartitioned model of the entire road infrastructure. The partitioning-based models take 100x less total sequential training time compared to single sensor models, i.e., one model per sensor. Furthermore, the partitioning-based models have 2x less prediction error (RMSE) compared to both the single sensor models and the entire road model.

***Index Terms*—spatial time-series, deep learning, LSTM**

I. Introduction

Deep neural networks (NN) have shown promising results for different machine learning and data mining tasks, such as classification and prediction, in various application domains. Modelling of a large complex system requires a large dataset to train a deep NN with many parameters [1]. At scale, training deep NNs is computationally and memory intensive. Partitioning and distribution is a general approach to the challenge of scale in NN-based modelling. A number of methods have been proposed to achieve scalability, such as distributed and collaborative machine learning [2], [3] that rely on dividing the problem into smaller tasks. These tasks comprise of smaller models working on subsets of data.

In this work, we address the scalability problem in large-scale spatial time-series forecasting. Spatial time-series are multiple time-series that correspond to different spatial locations [4]. There are dependencies between spatial time-series that need to be taken into account when building an NN-based prediction model. Our approach to tackle the scalability problem is to partition time-series data while preserving essential dependencies between them. Our application domain is the analysis of road traffic data collected by road infrastructure sensors and modelling of traffic flow behavior for the task of traffic prediction. Accurate traffic predictions can further help in route planning, traffic congestion reduction, air pollution reduction, infrastructure planning, and other tasks.

The number of sensors and the number of measurements from the sensors are increasing. For example, the number of infrastructure sensors in the Motorway Control System (MCS) deployed on highways in Stockholm and Gothenburg, Sweden, has increased from about 800 in 2005 to more than 2000 in 2016. The number of measurements has also increased from 400 million to about 1 billion per year, as shown in Fig. 1.

Sensor data are spatial time-series, and having a large number of sensors causes a scalability problem in traffic time-series forecasting. Moreover, often there are real-time requirements for traffic data analysis and forecasting that put constraints on the inference time. This requires a scalable solution for processing a large amount of data with low-latency. In some cases, to achieve scalability, the data is partitioned in order to perform mining or modelling tasks in parallel.

Careless partitioning causes degradation in the model accuracy. For example, too small partitions might not provide enough information to train the model; Placing unrelated data, such as traffic data from unconnected road segments, in the same partition might negatively affect the accuracy or complexity of the model. Careful partitioning is especially important for grouping time-series data of multiple sensors (spatial time-series), because of dependency between the sensor readings. One example of such a dependency is that a moving car counted by one sensor will be counted by the next sensor in the flow direction. Another example is that a traffic queue growing in the opposite direction of the traffic flow causes a slowdown of cars and, as a consequence, dependency between sensor readings. The dependency is strong between sensors which are closely placed and there is a path between them; the dependency is weak between sensors which are far apart or have no path between them. Taking these dependencies into account is important for the partitioning of sensors, meaning that a partition should include correlated sensors.

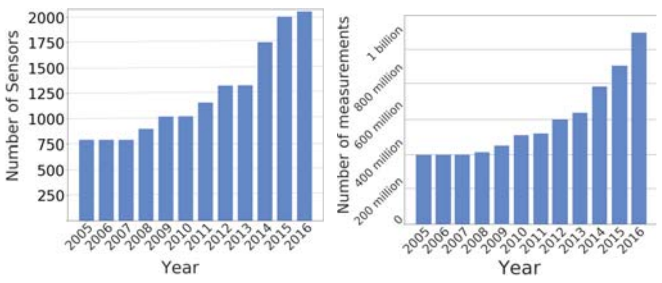


Fig. 1: Number of sensors' measurements over years.

In our previous work [5], we have found that partitioning the sensors by grouping them based on area results in low prediction accuracy because a partition might cover sensors on unconnected road segments or sensors with long distances between them, hence with uncorrelated or weakly correlated data. In order to group only relevant correlated sensors, to reduce the complexity and improve the accuracy of the prediction model, we performed manual partitioning by checking the locations of the sensors and grouping the related sensors located on the same road paths. Our focus was on accuracy within a few manually selected partitions, not the entire network, because it was not feasible to manually partition the entire traffic network. In our other work [6] on road congestion detection, we have found that representing the sensor infrastructure in the form of a weighted graph allows to capture inter-dependencies and spatial relations between sensors and to use graph processing in traffic analysis.

In this work, we propose a novel partitioning algorithm to automate partitioning of sensor networks that shows similar results, in terms of prediction accuracy, compared to manual partitioning used in our previous work. We show that the deep NN models using the partitioned network for the prediction can scale to cover large traffic networks with improved performance in terms of compute time and accuracy.

In this work, we address the following research questions: *How to partition spatial time-series to preserve dependence among them? Which and how many spatial time-series do we take into account for a fast and accurate forecast?*

Contributions: The contributions of our work are:

- **Representation:** We propose to represent a complex system as a directed weighted graph that captures the dependencies between spatial time-series generated by components of the system, and thus, enables graph partitioning that yields correlated component groups. We use our approach in representing the road infrastructure sensors as a directed weighted graph that reflects spatio-temporal dependencies among sensor readings in order to partition the sensor graph into correlated sensor groups.
- **Partitioning:** We propose a novel graph partitioning algorithm to find groups of correlated sensors for a given prediction time horizon. The size of a partition is determined in such a way that it covers all correlated sensors reachable within the time horizon by a car travelling at the average speed in each segment of the partition, i.e., the sensors affected by the same traffic stream are placed in the same partition if the travel time between them lies within the prediction horizon. This allows partitioning of a large sensor network while preserving dependencies between spatial time-series generated by the sensors.
- **Scalability:** We use our proposed partitioning technique to tackle the scalability problem by enabling parallelism in both training and prediction. After partitioning the sensor network graph, we train a prediction model for each partition. We show that our approach can scale to cover large traffic networks with improved performance in terms of the compute time and accuracy.
- **Comparison:** We evaluate our approach with various partition sizes and prediction horizons. We also compare our partitioning-based models with single sensor models, i.e., one model trained per sensor, and the entire sensor infrastructure model, i.e., one model trained for the entire sensor network. The comparison is done in order to find a better model in terms of performance (speed of training and prediction) and accuracy (prediction error).

Main Findings: Our experimental results show that using time-series prediction models on a related group of sensors yields better performance in terms of training and prediction time and also gives better accuracy compared to the models of a single sensor or the entire sensor network. With the case study of modelling of a large sensor network considered in this work, we have found that the partitioning technique that takes into account the dependencies between spatial time-series generated by the sensors is able to handle large scale data more efficiently compared to training single-sensor models or training a single model for the network of all sensors.

II. PRELIMINARIES

A. *Traffic Flow Theory*

Traffic flow theory [7] focuses on understanding the behavior of traffic on the road by analyzing the interactions between the transportation infrastructure and its users. Traffic flow theory is primarily used by transportation engineers to analyze and optimize the flow of traffic for designing, constructing and maintaining the road infrastructure.

Traffic flow theory uses three main measures, namely, traffic flow, speed, and density to characterize the traffic performance. These measures are collected from sensors placed on roads or vehicles monitoring the traffic flow and drivers' behavior. The aforementioned traffic measures can be defined as follows: 1) Traffic flow, f, is the number of vehicles passing through a particular point on the road per time interval. It is commonly denoted as vehicles per hour (veh/hr). 2) Speed, v, is the distance covered by a vehicle per time interval. It is denoted as kilometers per hour (km/hr). 3) Density, d, is the number of vehicles per distance unit. It is useful in analyzing the congestion on a road. Density is represented as vehicles per kilometer (veh/km). Empirical data shows the relation between these variables as $d = f/v$. In our experiments, we predict and use traffic density for training the prediction model because of its importance for road congestion detection.

B. Long Short-Term Memory NN-Based Prediction Model

Traditional time-series forecasting techniques for traffic prediction, including flow theory based models and statistical techniques, such as Bayesian analysis [8], Markov chains [9] and ARIMA [10], have been replaced by neural networks. NNs perform better for time-series forecasting than the traditional techniques because the latter are incapable of handling missing and multidimensional data. NNs have been employed in forecasting time-series data [11], [12]. In particular, Long Short Term Memory (LSTM) networks [13] have shown promising results for traffic prediction due to their deep hierarchical structure which enables extraction of non-linear and stochastic characteristics of the traffic data [14], [15].

In this work, we use LSTM-based prediction models. LSTM is a type of Recurrent Neural Network (RNN) architecture that is capable of learning long term trends in the data. In our previous work [5], we compared a 2 layered stacked LSTM based architecture with various statistical and neural networks based models. The stacked architecture containing 2 layers gave better accuracy compared to other models.

Complexity: The basic LSTM architecture consists of three layers: the input layer, LSTM (hidden) layer, and output layer. Data from the input layer is fed to the LSTM layer which contains memory blocks comprising of memory cells with self-connections and gates. These self-connections are from cell's output unit to the input unit and gates. These gates, namely the input, output and forget gate control the flow of data across these cells which represent the state of LSTM. The output units of cells are connected to the output layer.

The computational complexity of an LSTM network per time stamp and weight is $O(1)$ [13]. Therefore, the complete learning complexity of LSTM with a total of W number of parameters is $O(W)$. W is computed by the equation [16]:

$$W = n_c^2 \times 4 + n_i \times n_c \times 4 + n_c \times n_o + n_c \times 3 \qquad (1)$$

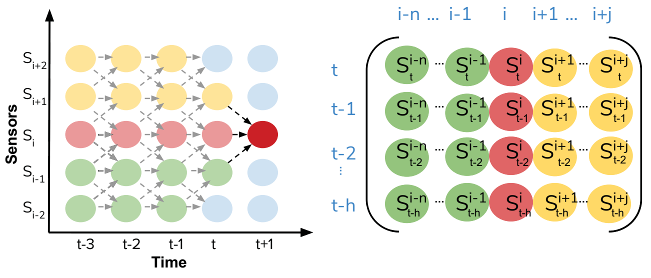
Here, n_c is the number of memory cells, n_i is the number of input units and n_o is the number of output units in the LSTM layer. A large number of input, output and memory units can increase the computational complexity of LSTM.

C. Data Representation for Spatial Time-Series Forecasting

Time-series data collected from road traffic sensors contain both spatial and temporal dependencies. To capture these dependencies, we need to take into account the sensor's previous readings, the readings of neighbouring sensors and neighbouring sensors' previous readings. Therefore, we model the sensor data using Equation 2, which represents the time-series dependencies in space and time. Using this equation, we consider that the predicted reading of a sensor S_i at time $t+1$ is given by the following equation:

$$\begin{aligned} S_{i,t+1} = f(&S_{i-n,t}, S_{i-(n+1),t}, ..., S_{i,t}, S_{i+1,t}, ..., S_{i+j,t},\\ &S_{i-n,t-1}, S_{i-(n+1),t-1}, ..., S_{i,t-1}, S_{i+1,t-1}, ..., S_{i+j,t-1},\\ &..., S_{i-n,t-h}, S_{i-(n+1),t-h}, ..., S_{i,t-h}, S_{i+1,t-h}, ..., S_{i+j,t-h}) \end{aligned} \qquad (2)$$

Here, $S_{i+1}, ..., S_{i+j}$ are readings of the sensors placed downstream, i.e., in the direction of traffic flow, $S_{i-1}, ..., S_{i-n}$ are readings of the sensors placed upstream, i.e., in the opposite direction of traffic flow, and $t, ..., t-h$ refers to h time units in the past from the current time t. Fig. 2a shows that the prediction of the sensor reading S_i at time $t+1$ depends on the readings from neighbouring sensors placed downstream (S_{i+1} and S_{i+2} shown in yellow), sensors placed upstream (S_{i-1} and S_{i-2} shown in green) and sensor S_i's previous readings in time intervals $t, ..., t-h$ (in red).



(a) Spatio-temporal dependencies of sensor data

(b) Space-Time window representation of sensor data

Fig. 2: Data representation of spatial time-series

We map Equation 2 to a space-time window which is fed as an input to train the prediction model. For a sensor S_i on a particular point in the highway, the space-time window contains sensor readings, i.e., the density of vehicles computed using the flow and average speed recorded by the sensor, from $i-n$ sensors placed upstream and $i+j$ sensors placed downstream. These readings are taken for a history of h time stamps from the current time t. As shown in Fig. 2b, the first row of the window contains sensor readings from $S_{i-n}...S_i...S_{i+j}$ at a current time interval t. Similarly, the next row is for readings at $t-1$, i.e., the previous minute. The rows following this contain previous readings till $t-h$ time interval. Once the input is fed, the prediction model is trained to predict for $t+n$ future time intervals traffic density.

III. Graph Representation and Partitioning

A. Graph Representation

We represent the traffic infrastructure sensors in the form of a directed weighted graph to capture the spatio-temporal dependencies between the time-series from the sensors. We construct the graph G based on the road paths and the traffic direction between the sensors. Next, we weight the graph using sensor readings and distance between the sensors. Sensors that are at the same location but on different lanes are represented as a vertex in V. The paths between these vertices are represented as edges in E. The direction of the traffic flow determines the direction of the edges. Fig. 3 shows the sensors on parallel lanes at the same location grouped together as sensor site vertices (in blue) and the road between them are represented as directed edges (in red). Fig. 4 depicts a high-level view of the sensor graph of Stockholm centre.

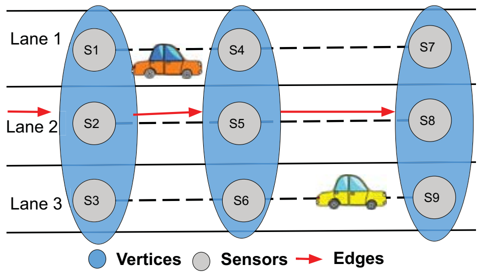


Fig. 3: Sensor graph creation

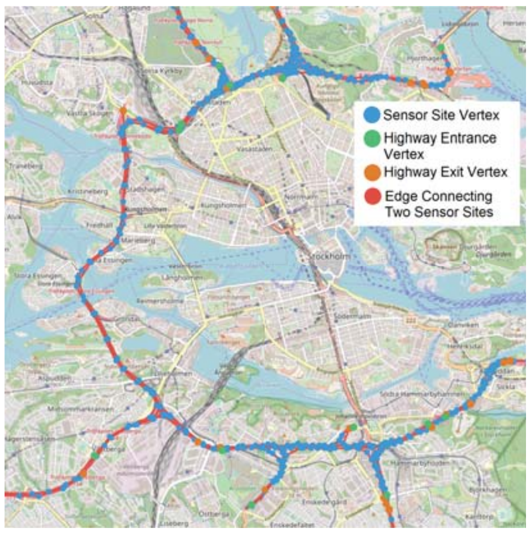


Fig. 4: Graph representation of road sensors in Stockholm

Once the graph is constructed, the edges are weighted based on the travel time of the cars. The edge weight is equal to the time it takes a vehicle to travel the road segment corresponding to that edge. The weight $w(i, j)$ represents the weight of an edge directed from the vertex v_i to v_j. $w(i, j)$ is calculated using the average speed v of vehicles recorded by the sensors corresponding to the vertex v_j during rush hours and distance d between the sensors corresponding to vertices v_i to v_j. Equations 3 [17] are used to compute the distance d and travel time t for weighting the edges.

$$
\begin{gathered}
d = 2r \arcsin(\sqrt{\sin^2(a) + \cos(\varphi_1)\cos(\varphi_2)\sin^2(b)}) \\
t = d/v \qquad (3) \\
\text{where } a = (\varphi_2 - \varphi_1)/2 \text{ and } b = (\lambda_2 - \lambda_1)/2
\end{gathered}
$$

φ_1 is start latitude in radians, λ_1 is start longitude in radians, φ_2 is end latitude in radians, λ_2 is end longitude in radians, r is Earth radius (6371 km), d is distance between sensor sites.

B. Graph Partitioning

We partition the directed weighted graph, that represents the traffic infrastructure sensors, in order to find groups of correlated sensors. The partitioning algorithm consists of three stages: 1) Creation of base partitions; 2) Creation of the base partitions graph; 3) Addition of partitions from front and behind to capture the dependencies of sensors located behind (in the direction of traffic flow towards the point of interest) and in front of the selected point of interest (in the direction of traffic queue growing or moving in the opposite direction of traffic flow towards the point of interest) in the graph.

Creation of Base Partitions. The base partitions are created using backward traversal of the graph from the starting point which is a vertex with no outgoing edge. The backward traversal is made to capture the dependencies between sensor readings caused by cars moving in the direction of traffic flow. For example, a moving car counted by one sensor will be counted by the next sensor in the flow direction.

The algorithm starts by taking an input parameter time t, and chooses a starting point, which is a node/vertex with no outgoing edge from it, in the graph. The traversal is made in the opposite direction of the incoming edge towards the node. The weights of edges along the traversal are added up until they reach the threshold t. Once the threshold is reached, the edges are added to a partition and the next partition starts from the last edge visited. The algorithm terminates when all edges are visited. This results in creating base partitions.

Algorithm 1 defines the steps to create base partitions. G is the input sensor graph, v is a vertex and $e(v, w)$ represents an edge. *getStartingVertices(G)* returns starting vertices in G. *getBackwardVertices(v)* is used to get vertices visited when traversed backwards from v and *getMaxWeight(v)* is used to get maximum weight of the partition containing vertex v.

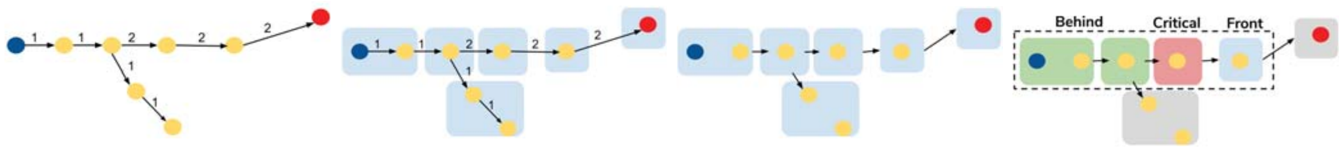
Fig. 5a shows an example of the directed weighted graph created to represent the road sensors, where the weights are based on travel times of vehicles. Fig. 5b shows the base partitions created over the weighted sensor graph with an input parameter $t = 2\ min$ and the starting vertex in red.

Creation of Base Partitions Graph. Once we have the base partitions, they are connected to form a base partitions graph, where the partitions are vertices and the flow directions between them are edges (Fig. 5c).

Additions of Partitions from Front and Behind. After we get the base partitions graph, first, we add some partitions from behind to capture the dependencies between sensor readings caused by the cars moving towards the tail of the base partition. Then, we add partitions from the front to capture dependencies between sensor readings caused by the traffic queue, created during a traffic jam, moving towards the head of the base partition. The number of partitions added from front and back affects the size of our final partitions.

A partition size is determined in such a way that it covers all correlated sensors reachable within the prediction time horizon by a car travelling at an average speed in each segment of the partition, i.e., the sensors affected by the same traffic stream are placed in the same partition if the travel time between them lies within the prediction horizon. This allows partitioning of large sensor networks while preserving dependencies between spatial time-series generated by the sensors.

In our experiments, we make predictions for up-to 30 min. Therefore, we add partitions from the back that cover at least



(a) Directed weighted graph (b) Base partitions (c) Base partitions graph (d) Critical partitions with forward and backward partitions

Fig. 5: Partitioning steps

Algorithm 1 Backward Graph Partitioning Algorithm for Creation of Base Partitions

```
P={}                                              ▷ Parititions initialized
function $main(G, s)$                    ▷ $G$ - graph, $t$ - input time
    starting_vertices=getStartingVertices($G$)
    for each ($v$ in starting_vertices)
        partition($G$,$v$,$t$,0)
    end for
    collect all partitions in $P$
    return $P$
function $partition(G, v, t, sum)$   ▷ $G$ - graph, $v$ - vertex, $t$ - input time, $sum$ - accumulated sum
    if $v$ is in $P$ then
        return
    if $t \leq sum$ then
        add $v$ to starting_vertices
        return
    add $v$ to a partition in $P$
    $next\_vertices$ = getBackwardVertices($v$)
    for $next\_vertex$ in $next\_vertices$ do
        $weight$ = weight of e($v$, $next\_vertex$)
        if $next\_vertex$ not in $P$ then
            $partition$($G$, $next\_vertex$, $t$, $sum$ + $weight$)
        else if $t$+$weight$ <=getMaxWeight($next\_vertex$) then
            merge the current partition with the partition containing $next\_vertex$.
        else if $sum$+$weight$ <= $t$ then
            add vertices of partition containing $next\_vertex$ to starting_vertices.
            $partition$($G$, $next\_vertex$, $t$, $sum$ + $weight$)
```

the distance travelled by the cars coming towards the base partition in 30 min time interval. Similarly, we add partitions at the front of the base partition by almost half the number of the ones added at the back, because the traffic queue builds up slowly and few sensors are covered by the traffic queue moving backwards, i.e., towards the head of the base partition. Fig. 5d shows the addition of partitions from the front (shown in blue) and behind (shown in green) of the current base partition or the critical partition (shown in red).

IV. Prediction Models

The prediction models we use for evaluation in our experiments consist of 1) The single sensor models, 2) the entire sensor infrastructure model and, 3) the partitioning-based traffic network models. In this section, we explain various features of these models, such as the number of input units, the memory units, and the output units. Table I contains the configuration parameters of the prediction models.

TABLE I: Configuration parameters of the prediction models

Model	Input units	Output units	Memory units
Single Sensor	1	1	50
Entire Sensor Infrastructure	2058	2058	1000
Partitioning-Based	partition size	critical partition size	1000

A. Single Sensor Models

The single sensor models (SSMs) are trained per sensor, where the input is only one sensor's readings and the prediction is also done for the same sensor. These models, when considered individually, have low complexity as the input and output is only one. However, these SSMs result in a very large number of models and they collectively require high computational power for training and serving. The prediction results of this model will be based only on the readings of a single sensor and no information from neighbouring sensors is taken into account. This might result in less accurate predictions because less information is fed to train the prediction model.

B. Entire Sensor Infrastructure Model

The entire sensor infrastructure model (ESIM) is a huge unpartitioned model trained on readings from all infrastructure sensors and makes predictions for all the sensors. Using all sensors' readings during model training can help the model learn the correlations between the sensor readings because of their spatio-temporal dependencies. The complexity of this model is directly dependant on the number of input and output units of this model. In our case, the number of input units and output units for this model are equal to 2058, which is the total number of sensors in the road sensor infrastructure.

C. Partitioning-Based Models

The partitioning-based models (Bt) are trained with sensors in the partitions created after partitioning the road infrastructure graph. The partitioning algorithm creates partitions to group the correlated sensors. Creating partitions helps in reducing the number of sensors used for training the prediction model, which results in a less complex model per partition compared to the entire sensor infrastructure model. The number of input and output units for the partitioning-based model depends on the size of the partition. In our experiments, we use various input parameters for creating partitions of different sizes. More details about the input parameters used in our work are given in the evaluation section V-C.

V. Evaluation

We evaluate the accuracy and performance of our proposed approach to predict road traffic density. We compare the proposed partitioning-based prediction models with the entire sensor infrastructure model and the single sensor models.

Accuracy: We measure the accuracy of models in terms of the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). RMSE and MAE between predicted density values $\hat{d}_{inm}$ and observed density values d_{inm} for i^{th} interval, N number of sensors over M minutes of the days are computed using the following equations:

$$MAE = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left|d_{inm} - \hat{d}_{inm}\right| \tag{4}$$

$$RMSE = \sqrt{\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(d_{inm} - \hat{d}_{inm}\right)^2} \tag{5}$$

Performance: We measure the performance of prediction models in terms of prediction and training time.

A. LSTM Network-Based Traffic Prediction Model

Our prediction model is based on the stacked LSTM network architecture with two LSTM layers. In our previous work [5], we compared the 2 layer LSTM network-based model with other prediction models including classical baseline statistical models, such as ARIMA, Support Vector Regression (SVR), and neural network-based models, such as RNNs with two layers, Feed Forward Neural Network (FFN) with two layers and LSTM-1 with a single LSTM layer. LSTMs with 2 layers proved to give better prediction accuracy than the aforementioned mentioned models. Therefore, in this work, we use the same stacked 2 layers-based LSTM prediction model and address the scalability problem.

B. Dataset

The traffic dataset used in this work was provided by the Swedish Transport Administration [18]. The dataset consists of readings from radar sensors on Stockholm and Gothenburg highways during the period 2005-2016. The sensors are placed a few hundred meters apart from each other on each lane. They collect data per minute, that results in a large and microscopic dataset compared to data aggregated per hour or over multiple lanes. The sensor readings include the flow and average speed of vehicles per minute. The dataset used for prediction consists of more than 88 million data points collected by 2058 sensors (Fig. 1) over a period of one month in 2016.

TABLE II: Accuracy of Prediction Models

Model	RMSE 10 min	RMSE 20 min	RMS 30 min	MAE 10 min	MAE 20 min	MAE 30 min
SSM	5.9	6.5	6.9	3.0	3.2	3.5
ESIM	6.1	6.2	6.5	3.1	3.1	3.2
B2	5.4	6.2	6.5	2.7	3.0	3.1
B5	5.4	5.7	6.2	2.7	2.8	2.9
B10	5.5	6.0	6.1	2.7	2.8	3.0
B15	**3.2**	**3.5**	**3.7**	**1.8**	**1.9**	**2.0**

C. Partitioning Parameters

We partition the road sensor network graph to create base partitions for various values of parameter t. Fig. 6 shows the results of partitioning for values of t = 3 min, 5 min, 10 min and 20 min. Partitions are shown in different colors, where colors of two partitions representing traffic in opposite directions may overlap. The higher the base partition creation input parameter, the bigger the partition size. Also, the total number of partitions becomes less with the increase in the input parameter. In Fig. 6d, we can see that almost all the highway is covered with the partition shown in black color. This partition further has one more similar partition for the traffic in the opposite direction.

D. Setup

We performed our experiments on-premises cluster. The on-premises nodes consist of `Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz`, 256GB of RAM and 16.6 TB disk. we used Apache Spark version 2.4.0. with Python 2.7.15 and Tensor Flow version 1.11.0.

VI. Results

Accuracy. We measured the accuracy of different prediction models by computing the RMSE and MAE. Table II shows the RMSE and MAE values for the Single Sensor Models (SSMs), the Entire Sensor Infrastructure Model (ESIM) and the Partitioning-Based models (Bt) using base partitions with the input parameter t = 2 min, 5 min, 10 min and 15 min, denoted as B2, B5, B10 and B15, respectively. SSMs have lower errors compared to ESIM for short prediction intervals, such as 10 min; whereas, ESIM has slightly less error for 20 min and 30 min intervals. Bt models show better accuracy compared to the base-line SSMs and ESIM. Partitioning models with 15 min (B15) base partition shows the best accuracy. Fig. 7 shows RMSE for the base-line SSMs and ESIM compared to B15. B15 has about 2x less RMSE compared to the base-line approaches. Furthermore, RMSE and MAE increase with the increase in the prediction horizon.

Performance. The performance of the prediction models is measured by computing the sequential training time, parallel training times, and the prediction time. Fig. 8 shows the training and prediction times of SSMs, ESIM and Bt models. One SSM has the prediction time < 1 sec, thus it is omitted from the plot in Fig 8c. SSMs have shorter parallel training and prediction times compared to other models. However, there are 2058 sensors and training/serving SSMs sequentially for

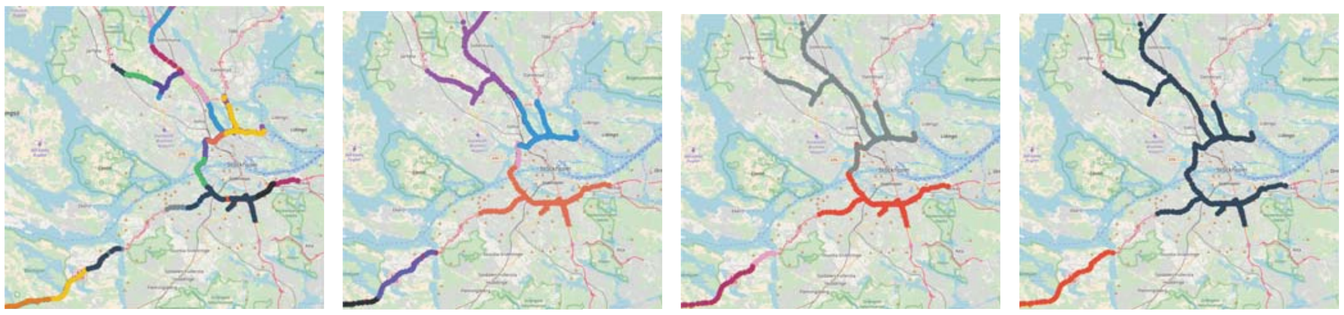

(a) 3 minutes partitions (b) 5 minutes partitions (c) 10 minutes partitions (d) 20 minutes partitions

Fig. 6: Partitioned road graph using different input parameters

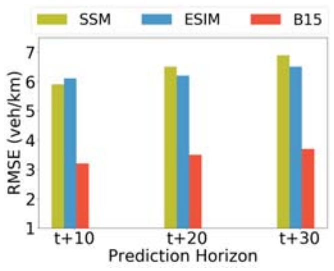


Fig. 7: RMSE (veh/km) of traffic density

TABLE III: Performance of Prediction Models

Model	Parallel training time (sec)	Sequential training time (sec)	Prediction time (sec)	No. of models
SSM	100.3	205600	<1	2058
ESIM	6300.2	6300	130.2	1
B2	4696.4	291152	55.3	62
B5	1776.1	33746	17.6	19
B10	838.2	10897	9.6	13
B15	522.1	4699	6.3	9

all these sensors take long time and makes SSMs inefficient in terms of performance. Therefore, the training and prediction time for SSMs, if run sequentially, is very high.

ESIM has the highest training time if run in parallel, and prediction time because of a large number of sensor data being processing to train this huge network. On the other hand, the Bt models have shorter training time, if run both in parallel and sequentially with B15, and less prediction time compared to ESIM. The training and prediction time for the partitioning-based models decreases with the increase in the input parameter for base partitions creation. Table III shows the performance comparison of these models. The number of partitions for B15 is only 9 and it has less training and prediction time compared to other models. Hence, it is suitable for large-scale traffic prediction. Overall, partitioning-based models Bt take 2x, if run sequentially, and 12x, if run in parallel, less training time, and 20x less prediction time compared to ESIM. The partitioning-based models take 100x less total sequential training time compared to SSMs.

Summary. Our results for comparing SSMs, ESIM and partitioning-based models show that we can achieve a scalable traffic prediction solution using partitioning-based models. Our experimental results show that: 1) SSMs are better for very short prediction horizons compared to ESIM and become less accurate with the increase in prediction horizon. At scale, the number of SSMs gets large which makes them inefficient for use in terms of overall high training and serving time. 2) ESIM albeit high training time yields better accuracy compared to SSMs for long prediction horizons. 3) Partitioning-based models with B15 shows overall best accuracy compared to SSMs and ESIM. The performance is also better compared to ESIM, in terms of the training and prediction time. Also, compared to SSMs the performance of partitioning-based models is better in terms of the training time if run sequentially.

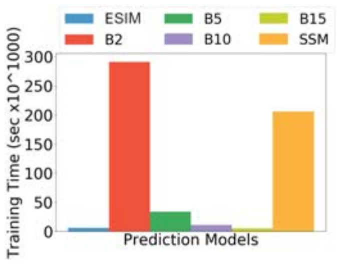
VII. Related Work

Partitioning. Our partitioning algorithm is based on backward traversal of the graph for finding base partitions. It is similar to finding sub-trees or sub-graphs in a graph. There are some prior works done on this problem. For example, [19] deals with finding siblings in the sub-tree and [20] deals with fusing child nodes to get smaller trees. The prior works are limited only to finding sub-trees in a tree and cannot be applied on general graphs that might contain cycles. In this work, we propose a partitioning algorithm for directed weighted graphs (with cycles). For the considered use-case of traffic flow, our algorithm takes into account the flow direction of cars to create sub-graphs or sub-trees containing road segments where the cars arrive from different directions towards a vertex in the graph selected as the starting point of the algorithm.

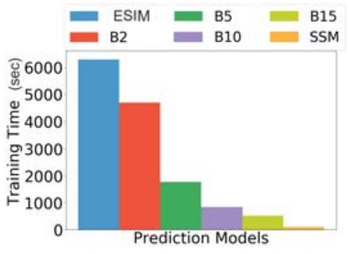
Scalability. Fouladgar et.al. [21] worked on scalable deep neural networks for urban traffic congestion prediction. However, they did not focus on addressing the scalability issue. Moreover, they use aggregated data over 5 min for only 58 locations. The issue of scale is still un-addressed. Other works [22], [23] done on large-scale traffic estimation and prediction do not fully address the issue of growing model complexities at scale.

VIII. Conclusion and Future Work

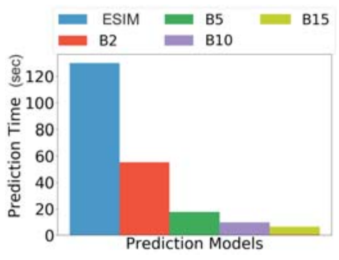
In this paper, we address the scalability problem of large-scale road traffic prediction using real-life large data-sets generated by traffic sensors deployed in Stockholm and Gothen-



(a) Training time (sec), if run sequentially



(b) Training time (sec), if run in-parallel



(c) Prediction time (sec)

Fig. 8: Training time (sec) for the sequential and parallel run, and prediction time (sec) of different prediction models

berg, Sweden. We propose a partitioning technique with corresponding algorithms to tackle the scalability problem that enables parallelism in both training and prediction, and hence reduces the training and model serving times while improving the accuracy of LSTM-based prediction models. We represent the road sensor infrastructure as a directed weighted graph to capture the spatio-temporal dependencies between the sensor readings; We propose a graph-based partitioning algorithm to group correlated sensors reachable within a given time horizon into partitions, to capture dependencies between spatial time-series of data collected by sensors in partitions, and to train and serve prediction models for partitions in parallel. The partitioning-based prediction models are fast and more accurate compared to single-sensor models and a single non-partitioned model for the entire road sensor infrastructure.

With this real-life case study, we have illustrated that partitioning is feasible and effective to address the scalability challenge in modelling of complex systems using deep learning, given that structural partitioning of data sources preserves the data dependencies, e.g., dependencies between sensor readings in the road infrastructure.

We believe that our proposed partitioning technique is general and can be applied to partition and model a complex system that can be represented as a directed weighted graph of dependencies between spatial time-series generated by components of the system.

Acknowledgment

This work was supported by the Erasmus Mundus Joint Doctorate in Distributed Computing funded by the European Commission (FPA 2012-0030) and by BIDAF project funded by the Swedish Knowledge Foundation (grant 20140221).

References

[1] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, vol. abs/1003.0358, 2010.
[2] J. Dean *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.
[3] J. Konečnỳ *et al.*, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.
[4] P. Zhang, Y. Huang, S. Shekhar, and V. Kumar, "Correlation analysis of spatial time series datasets: A filter-and-refine approach," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2003, pp. 532–544.
[5] Z. Abbas, A. Al-Shishtawy, S. Girdzijauskas, and V. Vlassov, "Short-term traffic prediction using long short-term memory neural networks," in *IEEE International Congress on Big Data*, July 2018, pp. 57–65.
[6] Z. Abbas, T. T. Sigurdsson, A. Al-Shishtawy, and V. Vlassov, "Evaluation of the use of streaming graph processing algorithms for road congestion detection," in *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications (ISPA)*, Dec 2018, pp. 1017–1025.
[7] B. S. Kerner, *The physics of traffic: empirical freeway pattern features, engineering applications, and theory*. Springer, 2012.
[8] S. Sun, C. Zhang, and G. Yu, "A bayesian network approach to traffic flow forecasting," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 124–132, 2006.
[9] H. Sun, H. X. Liu, H. Xiao, R. R. He, and B. Ran, "Short term traffic forecasting using the local linear regression model," in *82nd Annual Meeting of the Transportation Research Board, Washington, DC*, 2003.
[10] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," *Transportation Research Record Journal of the Transportation Research Board*, no. 722, 1979.
[11] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
[12] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.
[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[14] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.
[15] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.
[16] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *15th annual conference of the international speech communication association*, 2014.
[17] J. Inman, "Navigation and nautical astronomy, for the use of british seamen," 1849.
[18] Trafikverket, https://www.trafikverket.se/, 2010.
[19] C.-C. Kanne and G. Moerkotte, "A linear time algorithm for optimal tree sibling partitioning and approximation algorithms in natix," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 91–102.
[20] T. Ito, T. Nishizeki, M. Schröder, T. Uno, and X. Zhou, "Partitioning a weighted tree into subtrees with weights in a given range," *Algorithmica*, vol. 62, no. 3-4, pp. 823–841, 2012.
[21] M. Fouladgar, M. Parchami, R. Elmasri, and A. Ghaderi, "Scalable deep traffic flow neural networks for urban traffic congestion prediction," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2251–2258.
[22] X. Ma *et al.*, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.
[23] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLOS ONE*, vol. 10, no. 3, pp. 1–17, 03 2015.