Applied Intelligence (2020) 50:3252–3265
https://doi.org/10.1007/s10489-020-01716-1

A recurrent neural network for urban long-term traffic flow forecasting

Asma Belhadi[1] · Youcef Djenouri[2] · Djamel Djenouri[3] · Jerry Chun-Wei Lin[4]

Published online: 16 May 2020

Abstract
This paper investigates the use of recurrent neural network to predict urban long-term traffic flows. A representation of the long-term flows with related weather and contextual information is first introduced. A recurrent neural network approach, named *RNN-LF*, is then proposed to predict the long-term of flows from multiple data sources. Moreover, a parallel implementation on GPU of the proposed solution is developed (*GRNN-LF*), which allows to boost the performance of *RNN-LF*. Several experiments have been carried out on real traffic flow including a small city (Odense, Denmark) and a very big city (Beijing). The results reveal that the sequential version (RNN-LF) is capable of dealing effectively with traffic of small cities. They also confirm the scalability of *GRNN-LF* compared to the most competitive GPU-based software tools when dealing with big traffic flow such as Beijing urban data.

Keywords Learning long-term flows · Recurrent neural network · Weather information · Contextual information

1 Introduction

Recent advances in technologies and infrastructures, such as high support GPS, mobile communications, wireless sensing and internet of things, make our cities more and more connected, digitalized, and thus smart. One of the most attractive smart city applications is urban traffic flow analysis. The traffic flow is computed by counting the number of objects (cars, passengers, cabs, buses, etc.) that cross a given location during a time interval. In the last few years, several learning algorithms have been proposed for traffic flows forecasting [1–5]. However, these algorithms are only able to predict short-term flow, i.e flows represented by a single flow observation. That is, they are only able to provide short-term flow forecasting, but not long-term. Long-term flows, is defined by the set of flow sequences captured during a specific time period [6, 7]. In the last decade, several works have been proposed for sequence data forecasting. Zhao et al. [8] proposed the global weighting algorithm for forecasting sequence data retrieved from real image datasets. The sequence data is constructed from the set of the training data, which is viewed as basis function. Each sequence data is estimated by weighting all the basis functions using the average distance of all the training data. The results of the global weighting approach are satisfactory when the training and test data have high similarity values. However, the accuracy of this approach tends to decrease when there is a high gap between the training and the test images. Chen et al. [9] developed the softmax regression approach by considering the posterior emotion as a softmax transformation of linear functions of the features of each sequence data. The learning parameters are obtained by minimizing the errors between the predicted results and their ground truth

✉ Asma Belhadi
abelhadi@usthb.dz

Youcef Djenouri
youcef.djenouri@sintef.no

Djamel Djenouri
djamel.djenouri@uwe.ac.uk

Jerry Chun-Wei Lin
jerrylin@ieee.org

1 RIMA, USTHB University, Algiers, Algeria

2 Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway

3 Computer Science Research Centre, Department of Computer Science and Creative Technologies, University of the West of England, Bristol, UK

4 Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway

using the gradient descent method [10]. Chang et al. [11] developed the shared sparse learning approach, with a main assumption that the shared sparse coefficient is used to match between each data sample and the remaining training sequence data using the $L-1$ norm. Lv et al. [12] developed a hybrid supervised and unsupervised model to automatically predict driver's braking intensity in a smart vehicle system environment. The cylinder brake pressure data is fitted to the gaussian distribution and labeled to three classes (*low*, *moderate*, and *intensive*) using the gaussian mixture model. Random forests and the artificial neural networks are then incorporated to predict braking intensity using the generated labeled gaussian distribution and the vehicle state information retrieved from the controller area network bus signals. Most sequence data forecasting apply optimization techniques. Obtaining the best fitting parameters for these techniques is challenging, which reduces the accuracy of the prediction process (noabely in scenarios of heterogeneous data). Therefore, these methods could not be applied for long-term traffic flow forecasting, and new methods are needed.

1.1 Motivation

For illustration, let us consider a daily scheduler of a person presented in Fig. 1. This person has some flexibility to arrange his daily activities from 11:00 onward. His lunch may be scheduled between 11:30 and 13:00, some personnel business between 15:00 and 16:00, and he may return home anytime from 17:00 to 20:00. Having an accurate long-term traffic flow forecast will enable this person to make optimal schedule of these activities (early morning or overnight) while minimizing the wasted time on the traffic jam. The existing models only allow to predict a short-term traffic flow and not the flow over long interval, which is required in this scenario. This motivates the need of models for long-term flows in several time intervals and arises the following questions:

1. How can we efficiently represent the long-term flow values for different time intervals?
2. How can we predict a new long-term flows from historical data?
3. How can we improve the accuracy of the existing sequence data forecasting process?
4. How can we predict real long-term flows from the real city data?

1.2 Contributions

In this work we consider the aforementioned questions and propose a new framework for predicting long-term traffic flows. To the best of our knowledge, we are the first to predict the long-term traffic of flows. The main contributions of the paper are summarized as follows:

1. We define an adequate representation of long-term flow values based on sequence data. A new strategy of constructing the historical traffic flow database is then developed by taking into account the flow information, the temporal information, and the contextual information.
2. We propose a new framework called RNN-LF that uses multiple data sources for predicting long-term traffic flow data. The framework is based on a recurrent neural network (RNNs) with the considered information plugged as input (information on the weather, and the context), and the historical long-term traffic flows as output.
3. Motivated by the success of GPU (Graphical Processing Units) in solving real world complex problems, we propose a GPU-based implementation called GRNN-LF, which deals with big urban traffic flow data in reasonable time. In this implementation, the initialization of the weights is performed on CPU, where the two most intensive-time (computing and updating weights) benefits from the massively threaded GPU.
4. To demonstrate the usefulness of the proposed framework, two use cases have been studied. The first case study is a real Odense traffic flow, and the second one is real big Beijing traffic flow data. The results show the

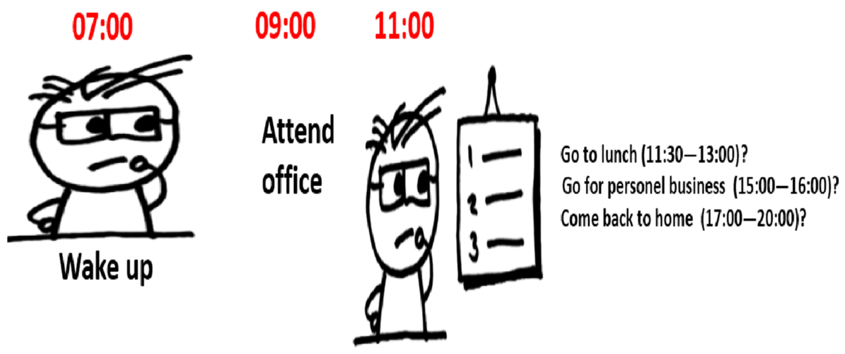

07:00
09:00
11:00
Wake up
Attend office
1
2
3
Go to lunch (11:30–13:00)?
Go for personel business (15:00–16:00)?
Come back to home (17:00–20:00)?

Fig. 1 Motivating example

superiority of our solutions compared to the baseline traffic-flow forecasting algorithms.

1.3 Outline

The paper is organized as follows: Section 2 relates the existing traffic flow forecasting algorithms. Section 3 presents the proposed framework for long-term traffic flow forecasting. Section 4 presents GPU implementation of the framwork. Experimental analysis of the two real case studies is shown in Section 5. Section 6 presents the main findings of applying the proposed framework on urban traffic data. Section 7 concludes the paper and previews future work.

2 Literature review

In the last few years, several studies have been investigating traffic flow prediction [1, 3, 13]. Yang et al. [2] used a state-of-the-art deep learning model (auto-encoder Levenberg-Marquardt) for improving the accuracy of traffic flow forecasting. It was designed using the Taguchi method to develop an optimized structure and learn traffic flow features. A layer-by-layer feature granulation was used with a greedy layer wise unsupervised learning algorithm. Huang et al. [4] proposed a hybrid model, which incorporates the online seasonal adjustment factors and adaptive Kalman filter to model the high traffic flow rate in a seasonal period. Four seasonal adjustment factors (daily, weekly, long daily, and long weekly) are first determined using online seasonal adjustment factors. The adaptive Kalman filter is then established to estimate high traffic flow rate upon a normality assumption. Zhang et al. [14] proposed the use of machine learning and evolutionary computation to predict flow rate. The most relevant features are first selected using both random forest and the genetic algorithm. The best features are then plugged into a support vector machine for learning and predicting new flow rates. Daraghmi et al. [15] used negative binomial model to smooth both spatial and temporal variables in the traffic flow forecasting. Chen et al. [5] proposed an ensemble learning approach represented by multiple non-linear least squares support vector regression models for predicting traffic flow. To adjust the parameters of such models, a heuristic harmony search approach has been applied. Chan et al. [16] treated the problem of time-variation between the historical flows and the new flow. They integrated the particle swarm optimization on the fuzzy neural network for predicting short-term traffic flow. We conclude from this review that solutions of traffic flow prediction (including those based on deep leaning) are restricted to a short-term flows, in which only a single value of flow is observed. To the best of our knowledge, there is no work related to long-term flows in the context of urban traffic forecasting. The remaining of this paper addresses this by proposing a novel framework that integrates an RNN for long-term traffic flow forecasting based on the weather information, the historical traffic flow data, and the contextual information.

3 Proposed approach

Figure 2 presents the general predictive long-term traffic flow framework. It includes three steps:

1. **Data Collection:** Urban sensing and data acquisition technologies are used to collect urban traffic data. Each row data represents one observation defining the date, the time, and the type of objects (vehicles or bikes) that pass through the given location. Table 1 gives an example of data collection obtained in Gronlandsgade, Odense (Denmark).
2. **Extraction and Merging:** First, the long-term traffic flows are extracted from the urban traffic data obtained in the previous step, and then the flows are merged with

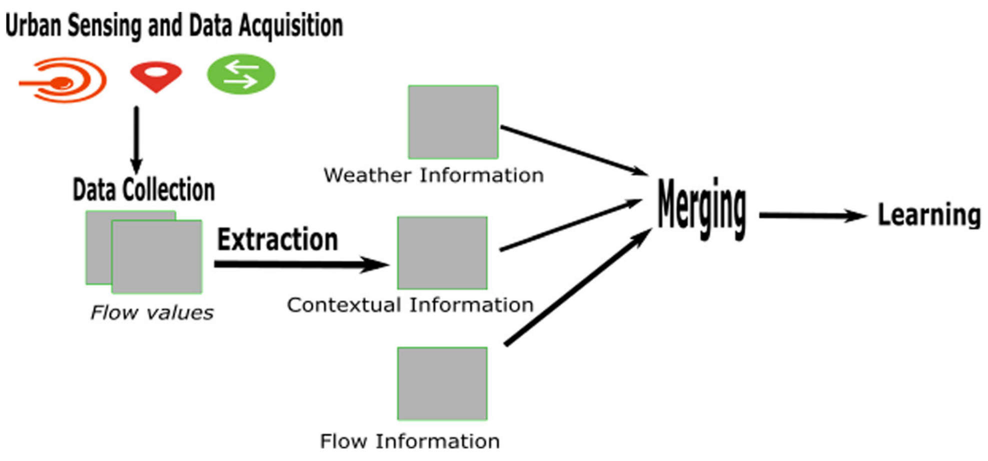


Fig. 2 General framework

Table 1 Data collection: Example of gronlandsgade location in Odense (Denmark)

Date	Time	Type
2017-01-23	14:17:20	Vehicle
2017-01-23	14:17:21	Vehicle
2017-01-23	14:17:25	Vehicle
2017-01-23	14:17:27	Vehicle
2017-01-23	14:17:29	Vehicle
2017-01-23	14:17:31	Vehicle
2017-01-23	14:17:34	Vehicle
2017-01-23	14:17:42	Vehicle

the weather information and the contextual information to build the long-term traffic flow database.

3. **Learning:** Machine learning and high performance computing (HPC) tools are used to process the input data, as well as the historical long-term traffic flow database designed in the previous step. The output data is the long-term traffic flows of the new observation.

The detail explanation of the main steps of this framework is given in the following.

3.1 Extraction and merging step

A long-term traffic flow database is created in this step. The long-term traffic flows are first extracted from the urban traffic data obtained in the data collection step. The daily long-term traffic flows are then merged with the daily weather information, and the daily contextual information.

Extraction A *traffic flow* is defined as the number of vehicles passing through a location in the road network during a given time interval. A *long-term traffic flow* (D_F) links flow values to their likelihood of occurrence during a given period of time. We estimate the probability of each D_F on the basis of their empirical counterparts based on real-life measurements. Let I be the set of time instants at regular time intervals at which flow measurements are collected in a specific location, and let $X = \{x_1, ..., x_{|I|}\}$ be the set of corresponding flow values. Let λ be the duration of the time interval between two succesive measurement instants, and μ the duration to be considered by each flow measurement $x \in X$. For example, X can be the set of number of vehicles or bikes passed through a location in the previous hour measured every 5 minutes, collected during a period of one month (30 days). In this case, $\lambda = 5$ minutes, $\mu = 60$ minutes and the number of measurements is $|I| = 43200/5 = 8640$. From I we can extract a collection $\mathcal{T} = \{T_1, \ldots, T_r\}$ of non-intersecting subsets of τ consecutive time instants. For a subset T_j, $j = 1, \ldots, r$, we identify the time instant when the subset starts with $\iota(T_j)$. That is, $T_j = \{\iota(T_j), \iota(T_j) + 1, \ldots, \iota(T_j) + \tau\}$. To each T_j, $j = 1, \ldots, r$, it corresponds a set of flow measurements X_{T_j}. For example, each X_{T_j}, $j = 1..7$ can contain the flow measurements between 7:00 and 9:00 for each day of a week. The flow measurements in each set X_{T_j}, $j = 1, \ldots, r$, can be represented as continuous random variables $Y_j \in \mathbb{N}_0$ to capture the uncertainty related to those measurements. Consequently, each Y_j can be described by its probability mass function $f_{Y_j}: \mathbb{N}_0 \rightarrow [0, 1]$ defined as $f_{Y_j}(y) = Pr(Y_j = [a, b])$, $a \in \mathbb{N}_0$, $b \in \mathbb{N}_0$ and $b \geq a$. To estimate $f_{Y_j} = f_j$, we use the empirical density function $\hat{f}_j$ of the flow given by the relative frequency of the measurements in X_{T_j}, i.e.,

$$\hat{f}_j([a, b]) = |\{x \in [a, b] \mid x \in X_{T_j}\}|/|X_{T_j}|, \tag{1}$$

where, $a \in \mathbb{N}_0, b \in \mathbb{N}_0, b \geq a, j = 1, \ldots, r$.

Example 1 Table 2 illustrats how to extract the long-term traffic flows from real traffic data of the week days retrieved from *Gronlandsgade*, located at Odense city in Denmark. We an interval size of 20, which allows to generate the following intervals: {[0-20], [21-40], [41-60], [61-80], [81-100], [101-120], [121-140]}. For instance, to build long-term traffic flows of Monday, we have to compute the number of possible flow values of all intervals of this day. In this example we have the following results:
$\hat{f}_1([0, 20]) = 8/12$ (i.e., there are 8 flow values between 0 to 20 that equal to 0 for Mondays from 9:00 to 12:00.

Similarly, we have

$$\hat{f}_1([21, 40]) = 1/12,\ \hat{f}_1([41, 60]) = 0,$$
$$\hat{f}_1([61, 80]) = 2/12,\ \hat{f}_1([81, 100]) = 1/12.$$
$$\hat{f}_2([101, 120]) = \hat{f}_2([121, 140]) = 0.$$

The final long-term traffic flows is given as follows:

$$\hat{f}_1 = \{8/12, 1/12, 0, 2/12, 1/12, 0, 0\},\ \hat{f}_2 = \{3/12, 0, 2/12, 5/12, 2/12, 0, 0\},$$
$$\hat{f}_3 = \{5/12, 1/12, 4/12, 2/12, 0, 0, 0\},\ \hat{f}_4 = \{4/12, 1/12, 3/12, 4/12, 0, 0, 0\},$$
$$\hat{f}_5 = \{3/12, 3/12, 2/12, 1/12, 1/12, 1/12, 1/12\}.$$

Merging The long-term traffic flows extracted in the previous step are merged with the weather information/contextual information. The weather information is composed with several features. In this paper, we are limited to the following three features:

- Conditioning: It includes three possible values (overcast, scatter, and part).
- The average wind temperature: It represents the average temperature of the wind given in degree celcius.
- The wind speed: It determines the wind speed in kilometre per hour.

Table 2 Odense (Gronlandsgade Location) Traffic data on Week days captured from 9:00 to 12:00

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:01-9:15	79	71	44	34	0
9:16-9:30	0	93	10	0	13
9:31-9:45	0	16	7	8	22
9:46-10:00	0	72	68	70	87
10:01-10:16	0	81	49	59	76
10:16-10:30	0	73	5	65	29
10:31-10:45	0	11	15	8	43
10:46-11:00	0	52	62	66	126
11:01-11:15	0	58	32	52	0
11:16-11:30	75	76	40	75	39
11:31-11:45	87	3	3	5	47
11:46-12:00	37	63	43	57	116

The contextual information is also composed with several features, which represents the profiling of the day. In this paper, we are limited to the following two features.

– (Weekend end vs regular) day: It includes the type of the day in the observation, 0 for weekend day, 1 for the regular day.
– Event day: It indicates if the day includes specific events such as new year day, national celebration day, or others. We set 0 for event day, 1 for non-event day.

In the long-term traffic flows database $DB = \{DB_1, DB_2...DB_r\}$, each row DB_i is composed by a tuple $< F_i, W_i, C_i >$. $F_i = \{F_{i1}, F_{i2}...F_{ik}\}$ is its long-term traffic flows, while k is the number of all possible flow intervals. $W_i = \{W_{i1}, W_{i2}...W_{ip}\}$ is its related weather information, while p is all possible weather features. $C_i = \{C_{i1}, C_{i2}...C_{in}\}$ is the set of contextual features, n is all possible contextual features. Table 3 presents an example of the long-term traffic flows database by considering 5 long-term traffic flows, 4 intervals, 3 weather features including conditioning: CD, average wind temperature degree: AWT, and windy speed: WS, and 2 pieces of contextual information, weekend vs. regular days: WR, and the whether it is an event day: E.

3.2 Learning step

The long-term traffic flows of the new observation is predicted in this step using recurrent neural network (many-to-many) architecture. As sketched in Fig. 3, the input data consists of the flow information, the weather information, and the contextual information of the current day, while the output data is a long-term traffic flow for the next day. We applied a multilayer feedforward neural network. Each neuron in layer l is linked with all neurons of the layer $(l-1)$ using different weight values. The neurons of the input layer is associated to each input data F_{i-1} (k possible intervals), W_{i-1} (p features of the weather information), and C_{i-1} (n features of the contextual information). The number of neurons in the input layer is $k + p + n$. The neurons of the output layer are connected to the output of the network $(\hat{F_{i1}}, \hat{F_{i2}}...\hat{F_{ik}})$. The aim is to minimize the error between the output data of the network and the long-term traffic flows F_i such that,

$$err = \sum_{i=1}^{r} \sqrt{\sum_{j=1}^{k} (F_{ij} - \hat{F_{ij}})^2 + (\bar{F_i} - \bar{\hat{F_i}}) + (sd(F-i) - sd(\hat{(F_i)}))}. \tag{2}$$

Table 3 Long-term traffic flows database example

Long-term traffic floas	Weather information	Contextual information
	(CWD, CD, AWT)	(WR, E)
(0.1:[1-10], 0.4:[11-20], 0.5:[21-30])	(Scatter, 10.0, 5.0)	(0, 0)
(0.3:[1-10], 0.5:[11-20], 0.2:[21-30])	(Overcast, 8.0, 12.0)	(0, 1)
(0.0:[1-10], 0.5:[11-20], 0.5:[21-30])	(Part, 12.0, 9.0)	(0, 0)
(0.5:[1-10], 0.1:[11-20], 0.4:[21-30])	(Overcast, 8.2, 8.5)	(1, 1)
(0.6:[1-10], 0.4:[11-20], 0.0:[21-30])	(Overcast, 12.0, 12.0)	(0, 1)

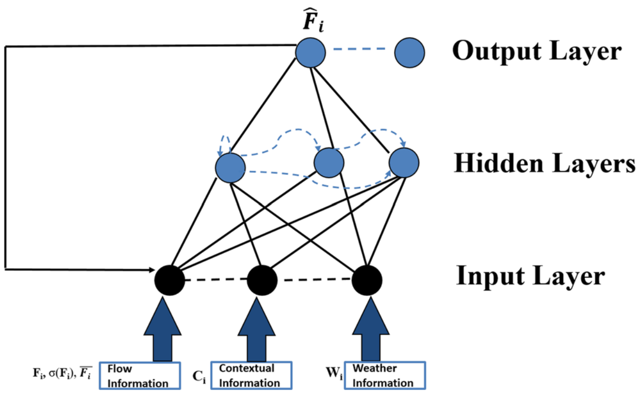


Fig. 3 RNN-LF framework

The output of the m^{th} neuron in the layer l, noted s_l^m, is given by (3). Note that the sum of the outputs of all neurons in the given layer should be equal to 1. This simulates the output of the long-term traffic flows.

$$s_l^m = \sigma\left(\sum_{j=1}^{|l-1|} s_{l-1}^j \omega_{l-1}^{mj} + b_l^m\right), \qquad (3)$$

with

$$\sum_{m=1}^{|l|} s_l^m = 1$$

, where

$\sigma(.)$	is the activation function.
$\|l\|$	is the number of neurons in the layer l.
s_{l-1}^j	is the output of the j^{th} neuron in the l-1 layer.
ω_{l-1}^{mj}	is the weight value that connects the neurons s_l^m and s_{l-1}^j.
b_l^m	is the bias value associated to the neuron s_l^m.

At each iteration i, the updating weight rule is given as follows,

$$\omega_{l-1}^{mj}(i) = \omega_{l-1}^{mj}(i-1) - \mu \times F_i \times 2 \times E_i \qquad (4)$$

Where
μ is the learning parameter rate, and

$$E_i = \sum_{j=1}^{k} (F_{ij} - \hat{F_{ij}})^2 \qquad (5)$$

Algorithm 1 RNN-LF algorithm.

```
1: Input: $DB = \{DB_1, DB_2 ... DB_r\}$: The long-term traffic flows database.
   $\sigma$: Activation function.
   $L$: The number of layers excluding the input and the output layers.
2: Output:
   $\Omega = \{\omega_{l-1}^{mj} | l \in [1...L], m \in [1..|l|], j \in [1..|l-1|]\}$.
3: for l=1 to L do
4:     for j=1 to $|l-1|$ do
5:         for m=1 to $|l|$ do
6:             $\omega_{l-1}^{mj} \leftarrow GenerateRandomValue(0, 1)$.
7:         end for
8:     end for
9: end for
10: for i=1 to r do
11:     for m=1 to p do
12:         $s_1^m \leftarrow W_{im}$.
13:     end for
14:     for l=2 to L do
15:         for j=1 to $|l-1|$ do
16:             for m=1 to $|l|$ do
17:                 $Computing(s_l^m, \sigma)$. {Eq. 3}
18:             end for
19:         end for
20:     end for
21:     $UpdatingWeight(\Omega)$. {Eq. 4}
22: end for
23: return $\Omega$
```

Algorithm 1 presents the pseudo-code of the RNN-LF algorithm. It starts by initializing weight values. The function *GenerateRandomValue* generates random values between 0 and 1. At each iteration, i, the neurons of the input layer receives the input data consisting of the flow information, the weather information, and the contextual information. This input is explored by all neurons of the network using the *Computing* function, which is calculated by (3). The output of the network is compared to the output data, and the error is determined using (5). This error value is propagated across the network to update the weight value as illustrated in (4). This process is repeated until all the input data is processed. This allows to minimize the error function shown in (2).

3.3 Complexity analysis

The theoretical complexity cost of the proposed framework is divided into the following costs:

1. Long-term traffic flow database construction cost: The long-term traffic flow database is built from the traffic flow. This operation requires $|DB_i|$ scans of the i^{th}

long-term traffic flow. Therefore, this operation needs $\sum_{i=1}^{t} |DB_i|$ scans.

2. Learning cost: The complexity cost of the recurrent neural network is O(L), where L is the number of layers.

The complexity cost of the proposed framework is $O(\sum_{i=1}^{r}((|DB_i|) + (L))$.

4 GRNN-LF

Generally speaking, the recurrent neural network models need massive training data in the learning process. This requires high computational time in a single CPU machine. The emergence of HPC tools such as CPU multi cores, GPU, and cluster computing allows to boost the performance of such models. We are interested in this paper by GPU computing. Several GPU-based software (including machine learning algorithms) have been developed for real world applications, e.g., [17–20]. However, these tools are limited, i.e, they come with predefined parameters and do not provide flexibility for users. For instance, a user cannot specify which tasks should be performed in parallel in GPU, how to distribute tasks to the GPU blocks, event to the threads-block, how to manage the shared memories of GPU blocks. To address such limitation, we need to deeply understand GPU computing and goes with more low level tools such as CUDA libraries,[1] which directly communicates with the hardware GPU components. In this section, we follow this approach and propose a new algorithm (GRNN-LF) that deals with efficient implementation of RNN-LF on GPU.

4.1 RNN-LF asnalysis

To design an efficient GPU-based approach, the most time consuming tasks for RNN-LF version should be determined. Algorithm 1, RNN-LF is divided into three steps: i) Initialization of weights (from line 3 to 9), ii) Computing outputs (from line 10 to 20), and iii) Updating weights (line 21). Flags are available after and before each of the three steps of RNN-LF mentioned above. Table 4 presents the experimental results for every step of the RNN-LF algorithm separately using different number of training data. Notice that by increasing the training data size from 1,000 to 10,000, the computing outputs and updating weights are clearly the most consuming tasks; The computing outputs task exceeds 77%, whereas the updating weights does not exceed 26% for the overall RNN-LF process. This explains

Table 4 Ratio of CPU time (%) of the main RNN-LF tasks on different training data size

$\|DB\|$ (*1,000)	Init.	Computing Outputs	Updating Weights
1	15	60	25
2	14	60	26
3	14	62	24
4	14	66	20
5	12	66	22
6	11	72	17
7	8	73	19
8	5	74	11
9	4	76	10
10	2	77	11

our choice for parallelizing this type of tasks. Motivated by the success of GPU in speeding up several real complex problems [21–23], we propose in the next section GRNN-LF, a GPU-version of RNN-LF.

4.2 Principle

GPUs (Graphic Processing Units) are graphical cards initially developed for video games, but their use as a powerful computing tool is now gaining field at many software application domains. The GPU hardware is composed of two hosts, i) the CPU and, ii) the GPU hosts. The former contains one processor and one main memory. The latter is a multi-threading system that consists of multiple computing *cores*, each core executes a block of threads. Threads of a block in the same core communicate with one another using a shared memory, whereas the communication between blocks relies on a global memory. The CPU/GPU communication is made possible by some hardware buses.

In the following, GRNN-LF is introduced as an adaptation of RNN-LF on GPU architectures for boosting the runtime performance. The most time-intensive operations are sent for GPU processing, whereas the less time consuming tasks are kept in CPU. The initialization of weights is first performed on CPU, and then the training data is then transmitted to the GPU that is used to update wrights and compute the outputs. Two main operations are distinguished:

1. Computing outputs: The process starts by handling the second layer using the output of the first layer, and so forth with the next layers until all layers have been processed. The process takes L iterations, where L is the number of layers in the neural network. At each iteration i, each block b_j is mapped onto the neuron s_i^j,

[1]https://developer.nvidia.com/

and every thread th^t_j is charged to compute the output of the connection between the neurons s^j_i and s^t_{i-1}. With, n neurons of each layer, n blocks, and n threads per block are required to compute the outputs of the given neural network. GNN-LF defines a *local table*, say $table_j$, for computing the weights of the neuron s^j_i.

2. Updating weights: Each block of threads is mapped onto one neuron, and every thread is charged to update the connection between the neuron of its block and one neuron situated in the previous layer. With, L, layers and, n neurons of each layer, $L \times n$ blocks, and n threads per block are required to compute and update all weights of the given neural network. GRNN-LF also defines a *local table* ($table_i$), for updating the weights of the i^{th} neuron of the neural network. This table is allocated in the shared memory of each block.

Algorithm 2 presents the pseudo-code of GRNN-LF using standard CUDA operations.

Algorithm 2 GRNN-LF.

```
1:  for each entry in the training data do
2:      {***********Computing Outputs***********}
3:      for $l = 2$ to $L$ do
4:          for $j = 1$ to $|l|$ do
5:              $v_{threadIdx.x} \leftarrow s^{threadIdx.x}_{l-1} \times \omega^{threadIdx.x,j}_{l-1}$
6:              $syncthreads()$
7:              $s^{blockIdx.x}_l \leftarrow \sigma(\sum^{|l-1|}_{threadIdx.x=1} v_{threadIdx.x} + b^j_l$
8:          end for
9:          $syncthreads()$
10:     end for
11:     {***********Updating Weights***********}
12:     $idx \leftarrow blockIdx.x \times blockDim.x + threadIdx.x$
13:     $\omega^{idx}_{blockIdx.x}(i) \leftarrow \omega^{idx}_{blockIdx.x}(i-1) - \mu \times F_i \times 2 \times E_i$
14:     $\Omega \leftarrow \bigcup \omega^{idx}_{blockIdx.x}$
15:     $syncthreads()$
16: end for
17: cudaMemcpy($\Omega$, cudaMemcpyDeviceToHost).
```

From a theoretical standpoint, GRNN-LF improves the sequential version of RNN-LF by exploiting the massively threaded computing of GPUs while computing outputs and updating weights. GRNN-LF also minimizes the CPU/GPU communication, by defining only two points of CPU/GPU communication. The first one takes place when the training database is loaded into the GPU unit, and the second when the final weight values are returned to the CPU. Moreover, GRNN-LF does not suffer from any threads divergence, because each thread deals with one multiplication for both computing outputs and updating weights steps. It also provides an efficient memory management, which explores the different shared memories of the blocks. However, synchronization between threads is needed at each step. Three synchronization points are then required at each iteration. The first point is when computing the output of each neuron, the second is observed when switching between layers for computing outputs. The last point of synchronization is needed when passing to another entry in the training data.

5 Performance evaluation

A number of experiments have been carried out using real traffic flow datasets from two different cities, Odense and Beijing. In the following, we first test *RNN-LF* approach on Odense urban traffic data using several configuration (location, number of hidden layers, and the learning rate) and compare it with the baseline data sequence forecasting algorithms. The ability of predicting real long-term traffic flows is also demonstrated. *GRNN-LF* is then evaluated using both Odense and Beijing traffic datasets and compared with the baseline GPU machine learning approaches. In the following, the interval time used for determining each flow value is fixed to 5 minutes, and each long-term traffic flows is measured within 4 hours (from 8 to 12) of each day. To ensure fair comparison between the proposed solution and the baseline algorithms, the experimentation has been carried out in the same environment and the same operating system. The results obtained are the average of 100 different tests. To prevent overfitting, we used the well-known dropout function available in Keras. We also used a high number of epochs values to prevent the underfitting (100 for sequential version, 1000, and 2000 for GPU-based version).

5.1 Data description

Two kinds of data have been used:

1. The first is a real urban traffic data from Odense Kommune (Denmark).[2] The data is a set of rows, where each row contains information related to the cars detected at specific locations such as the gap, length, location, date time, speed, class, as well as the weather data. The location is represented by lattitude and longtitude dimensions, the speed is calculated in km/h, and the date time (in the format YYYY-MM-DD hh:mm:ss) represents the year, the month, the day, the hour, the minute and the second that the car is passed by the given location. The most important information of each car is given as follows: The class is an integer that defines the type of the vehicle or the bike, e.g., 2 represents a passenger car. Tempm computes

[2] https://www.odense.dk/

Table 5 Odense location description

ID	Type	Name	(Latitude, Longtitude)	# (Cars or Bikes)
L_1	Cars	Anderupvej	(55.4383, 10.3896)	16.932
L_2	Cars	Falen	(55.3868, 10.3569)	25.310
L_3	Cars	Aløkke Alle	(55.4036, 10.3682)	238.775
L_4	Bike	Thriges	(55.4011, 10.3908)	46.978
L_5	Bike	N. Bohrs Alle	(55.3753, 10.4570)	445.883
L_6	Bike	Rodegadesvej	(55.3854, 10.4168)	575.089
L_7	Cars	Rugardsvej	(55.3990, 10.3634)	2.318.852
L_8	Cars	Nyborgvej	(55.3940, 10.4079)	2.352.930
L_9	Cars	Gronlandsgade	(55.4009, 10.4020)	2.955.464
L_{10}	Cars	Odins Bro	(55.4222, 10.3803)	3.921.746

the average daily temp in C, Wspdm determines the wind speed in km/h, and Conds defines a description of conditions weather. In this study, we focus on ten locations described in Table 5. The traffic data input is obtained from Odense flow that are observed between 1^{st} January 2017 and 30^{th} April 2018.

2. The second is a real urban traffic data from Beijing traffic flow [3]. It consists of more than 900 million traffic flow entries during a two-months time period in one location. The most important information of each car is given as follows: The Class in this dataset defines the type of vehicle or bus.

5.2 RNN-LF vs. sequence data prediction algorithms

The Odense dataset is used in this part of the experiments. The predictive rate is defined as the number of long-term traffic flows that are correctely predicted over the tested all long-term traffic flow. The first part of this experiment is to tune the RNN-LF parameters. Figure 4 presents the predictive rate of the *RNN-LF* algorithm using different locations in the city, different number of hidden levels (3, 5, 10), and different learning rate values (0.2, 0.5, 0.8, and 1.0). By varying the number of learning rate values from 0.2 to 0.8, and the number of hidden levels from 3 to 5, the predictive rate increases for locations. When the learning rate set is to 1.0 and the number of hidden to 10, the predictive rate decreases. The Gaussian function is used as it simulates a sequence much better than the other activity functions (e.g., Sinc function). In conclusion of this experiment, we set i) the number of hidden levels to 5, and ii) the learning rate to 0.8, as the best fitting parameters of *RNN-LF*. The next experiment investigate how RNN-LF can predict new long-term traffic flows. Figure 5 shows the long-term traffic flows predicted by RNN-LF compared to the real long-term traffic flows that uses the global weighting algorithm on Anderupvej location. This figure shows correlation between the result returned by RNN-LF and those for real long-term traffic flows. This result confirms the superiority of the proposed framework as compared to other existing algorithms. The following features contributed into this performance:

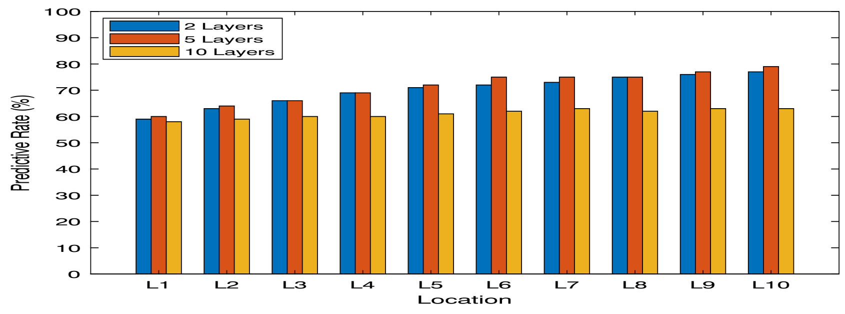
- The integration of several sources of data; flow information, the weather, and the contextual information.
- The recurrent neural network approach that allows to learn multiple output form multiple input of different sizes.
- The configuration used in the recurrent neural network that uses the best fitting values for the parameters (the number of hidden layers, and the learning rate).

5.3 GRNN-LF performance

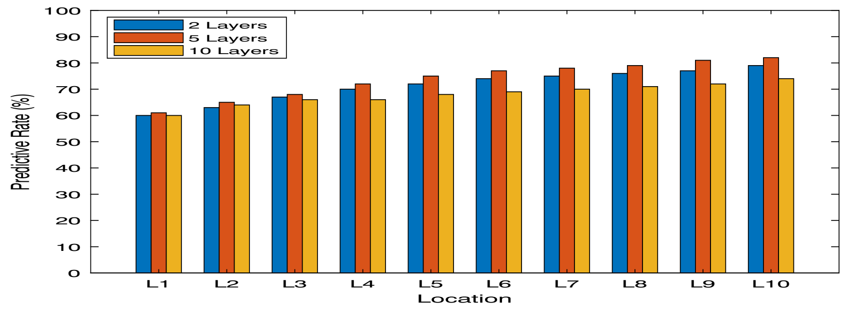
The performance of GRNN-LF using different dataset sizes is tested in this part. Figure 6 shows the speed up of GRNN-LF compared to the sequential version using Odense traffic data. The speed-up metric is calculated as $\frac{\lambda_1}{\lambda_b}$, where λ_b is the runtime using b GPU blocks. By varying the location used, the speed up of GRNN-LF is more than 50 for non-dense location (low traffic flows) and reaches 160 for dense location (high traffic flows). The last experiment considers big databases (Beijing traffic flow data) and compares GRNN-LF to some state-of-the-art GPU-based recurrent neural network software tools such as Agib's work [17], and Du's work [18]. Table 6 presents the runtime by varying the number of flows in million from 100 to 900. The results show that GRNN-LF outperforms all the other GPU-based algorithms. GRNN-LF deals with Beijing traffic flow in less than 3300 seconds, while the best performing GPU-based recurrent neural network software takes about 4000 seconds. These results confirm the effectiveness of GRNN-

[3]https://www.beijingcitylab.com/

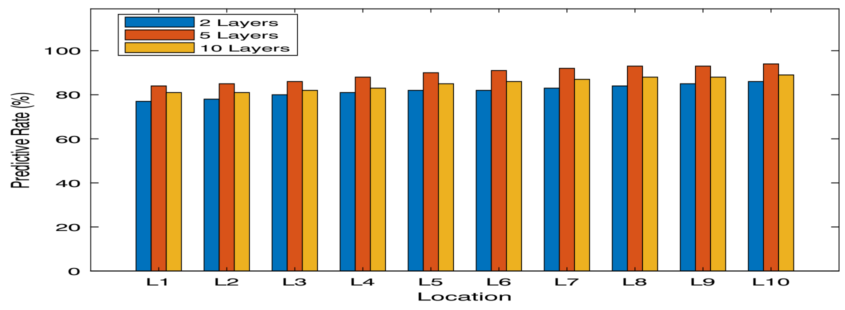
Fig. 4 Predictive rate (%) of the RNN-LF algorithm using different number of hidden layers (3, 5, 10), Gaussian function as activity function, and different learning rate values (0.2, 0.5, 0.8, 1.0)



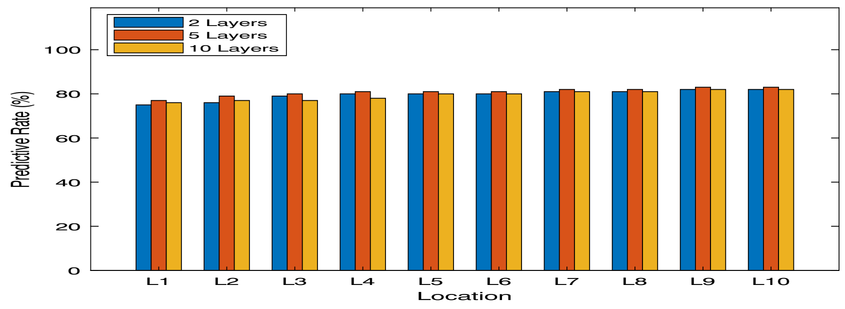
a- Learning rate=0.2



b- Learning rate=0.5



c- Learning rate=0.8



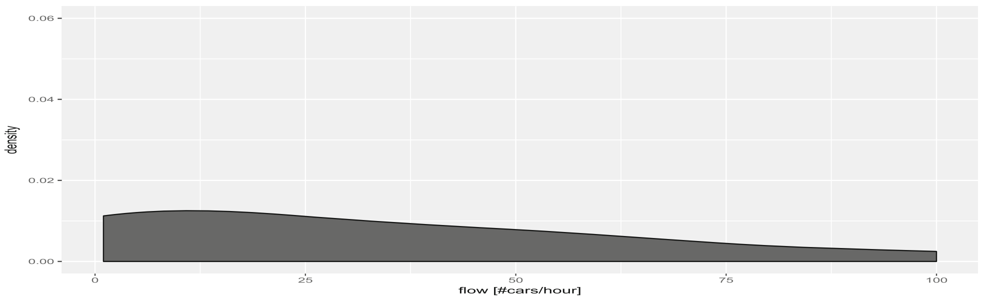
d- Learning rate=1.0

LF that performs intelligent mapping between the training data and the GPU massively threaded. However, the other GPU-based software tools are not flexible and proposed a general mapping without any deep analysis of the problem to be solved in GPU.

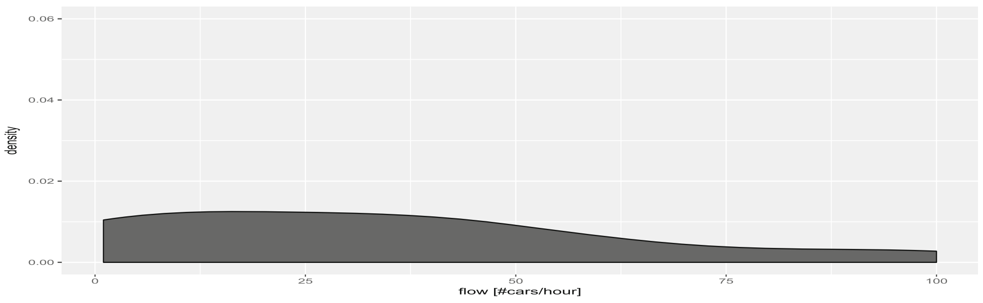
6 Discussions

This section discusses the main findings and limits from the application of our approach to both Odense and Beijing real traffic data.

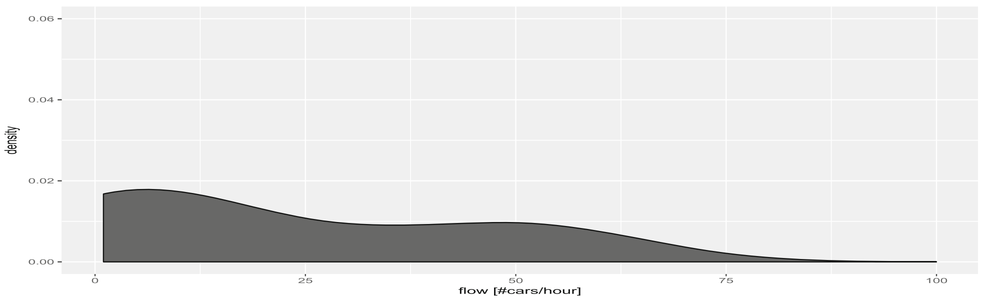
Fig. 5 The Original and the predicted long-term traffic flow by RNN-LF and the global weighting algorithm



Original long-term traffic flow



Predicted long-term traffic flow by RNN-LF



Predicted long-term traffic flow by the global weighting algorithm

- The first finding is that the proposed framework has the ability to integrate different data sources including urban data, weather information, and contextual information in the learning process. It is also flexible for integrating other data sources that can improve the accuracy of the learning model.
- The second finding is that the proposed framework outperforms the state-of-the-art traffic flow forecasting and has the ability to predict new patterns represented by long-term traffic flows. Moreover, the parallel implementation is able to deal with big traffic flow such as the Beijing dataset contains more than 900 million of flow entires.
- This work is an example of the application of a generic machine learning technique (a recurrent neural network) to a specific context. The literature calls for this type of research, particularly with the emergence of massive spatio-temporal data that becomes data are available in different locations and at different times. As in many other cases, porting a pure machine

Fig. 6 Speedup of GRNN-LF algorithm for different Odense locations

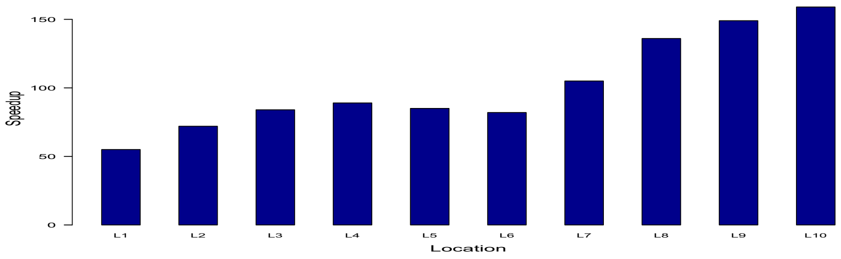


Table 6 Runtime (seconds) of GRNN-LF and baseline GPU algorithms for Beijing traffic flow data

# Flows (in million)	GRNN-LF	Agib‘s work [17]	Du‘s work [18]
100	1574	1748	1847
200	2001	2341	2594
300	2214	2745	2847
400	2470	2945	3067
500	2684	3147	3349
600	2814	3341	3674
700	3004	3541	3978
800	3145	3748	4179
900	3247	4017	4426

learning technique into a specific application domain requires methodological refinement and adaptation. In this context, we argue that our approach benefits from the knowledge extracted in the refinement step that shifts the intelligence required for predicting the long-term traffic flows.

However, there are some limitations of the application of our approach such as,

- The prediction model is based on three dimension (flow information, weather information, and contextual information) of one location. In contrast, the flow could be influenced by other information of other locations. Therefore, spatial information could also be used for the long-term traffic flow forecasting. Further, studying the different correlations among different long-term traffic flows allows to better understand the urban traffic data, and consequently increas the long-term traffic flow forecasting.
- In this research study, the considered contextual information could be coarse-grained for some applications. For instance, regular urban traffic data might be similar both in weekdays or weekends in some areas in a big smart city, where any difference of traffic could be identified for both weekdays or weekends. Thinking about other contextual information for some specified scenarios may improve the long-term traffic flow forecasting.
- The prediction model is based on the database of entire long-term traffic flows. This gives satisfactory results for datasets of small and medium size. However, the accuracy decreases for large and big datasets. e.g., Beijing traffic data. One direction to solve this is to study the correlation between long-term traffic flows. For instance, by grouping the long-term traffic flows into similar clusters and applying the learning model on these clusters separately. Another way to address this issue is to preprocess the data, by ignoring noise (detecting outliers), and/or extract relevant features (feature selection and extraction).

7 Conclusions and perspectives

A novel traffic flow prediction framework has been proposed in this paper. It aims to learn long-term traffic flows from multiple data sources. In this framework, the set of long-term traffic flows with weather and contextual information is first generated. The *RNN-LF* algorithm is then used to predict new long-term traffic flows. The scalability of the proposed framework has been investigated, and the results show that *RNN-LF* outperforms the state-of-the-art learning models for predicting sequence data. In addition, *RNN-LF* could predict long-term traffic flows from real case of Odense traffic flow data. To deal with big traffic flow data in real time, HPC-based version of *RNN-LF* has been developed. The approach called GRNN-LF has been implemented on GPU by developing an efficient mapping between the threads-block and the training data, and memory management optimization between different level of GPU memories. The less time-intensive task (initialization of weights) is performed on CPU, while the two most intensive-time tasks (computing outputs and the updating weights) benefit from the massively GPU threaded. In the computing outputs task, each block is mapped onto one neurone, and every thread is charged to compute the output of the connection between this neurone and one neurone of the previous layer. For the update of weights task, each block of threads is mapped onto one neurone, and every thread is charged to update the connection between the neurone of its block and one neurone situated in the previous layer. The results reveal that the GPU-based approach reach more than 160 speed up compared to the sequential RNN-LF when dealing with Odense traffic data. The results also show the superiority of the parallel version of GRNN-LF compared to the existing

GPU-based solutions for neural network learning when dealing with big Beijing traffic data.

Motivated by the promising results reported in this paper, we plan to investigate the following:

1. Extend *RNN-LF* for predicting urban trajectories data[24–26]. This aims to predict trajectory flows using multiple data sources.
2. Integrate deep learning and computational intelligence approaches [27–30] in the *RNN-LF* to improve again the accuracy of such approach.
3. Predicting other sequence data in real world applications such as Data Driven [31, 32], Image Processing [8, 9] and others.
4. Apply other hybrid GPU hardwares such as MultiCore GPU [33], and Clusters of GPUs [34].

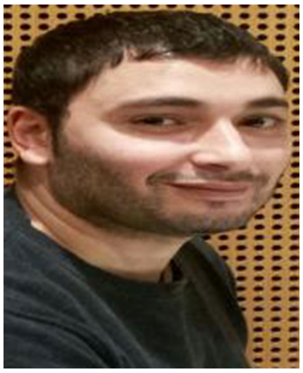
Funding Information Open Access funding provided by SINTEF AS.

References

1. Dombalyan A, Kocherga V, Semchugova E, Negrov N (2017) Traffic forecasting model for a road section. Transportation Research Procedia 20:159–165
2. Yang HF, Dillon TS, Chen YPP (2017) Optimized structure of the traffic flow forecasting model with a deep learning approach. IEEE Trans Neural Netw Learning Sys 28(10):2371–2381
3. Abadi A, Rajabioun T, Ioannou PA et al (2015) Traffic flow prediction for road transportation networks with limited traffic data. IEEE Trans Intell Transpo Sys 16(2):653–662
4. Huang W, Jia W, Guo J, Williams BM, Shi G, Wei Y et al (2017) Real-time prediction of seasonal heteroscedasticity in vehicular traffic flow series. IEEE Trans Intell Transpo Sys (99)1–11
5. Chen X, Cai X, Liang J, Liu Q (2018) Ensemble learning multiple LSSVR with improved harmony search algorithm for short-term traffic flow forecasting. IEEE Access 6:9347–9357
6. Djenouri Y, Zimek A, Chiarandini M (2018) Outlier detection in urban traffic flow distributions. In: 2018 IEEE international conference on data mining (ICDM). IEEE, pp 935–940
7. Djenouri Y, Belhadi A, Lin JCW, Cano A (2019) Adapted K-Nearest neighbors for detecting anomalies on spatio–temporal traffic flow. IEEE Access 7:10015–10027
8. Zhao S, Yao H, Gao Y, Ji R, Ding G (2017) Continuous probability distribution prediction of image emotions via multitask shared sparse regression. IEEE Trans Multimed 19(3):632–645
9. Chen L, Zhou M, Su W, Wu M, She J, Hirota K (2018) Softmax regression based deep sparse autoencoder network for facial emotion recognition in human-robot interaction. Inf Sci 428:49–61
10. Boyd N, Schiebinger G, Recht B (2017) The alternating descent conditional gradient method for sparse inverse problems. SIAM J Optim 27(2):616–639
11. Chang X, Ma Z, Lin M, Yang Y, Hauptmann AG (2017) Feature interaction augmented sparse learning for fast kinect motion detection. IEEE Trans Image Process 26(8):3911–3920
12. Lv C, Xing Y, Lu C, Liu Y, Guo H, Gao H, et al. (2018) Hybrid-learning-based classification and quantitative inference of driver braking intensity of an electrified vehicle. IEEE Transactions on Vehicular Technology
13. Lv C, Hu X, Sangiovanni-Vincentelli A, Marina C, Li Y, Cao D (2018) Driving-style-based co-design optimization of an automated electric vehicle: A cyber-physical system approach. IEEE Transactions on Industrial Electronics
14. Zhang L, Alharbe NR, Luo G, Yao Z, Li Y (2018) A hybrid forecasting framework based on support vector regression with a modified genetic algorithm and a random forest for traffic flow prediction. Tsinghua Science and Technology 23(4):479–492
15. Daraghmi YA, Yi CW, Chiang TC (2014) Negative binomial additive models for short-term traffic flow forecasting in urban areas. IEEE Trans Intell Transpo Syst 15(2):784–793
16. Chan KY, Dillon TS, Chang E et al (2013) An intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems. IEEE Transactions on Industrial Electronics 60(10):4714–4725
17. Aqib M, Mehmood R, Alzahrani A, Katib I, Albeshri A, Altowaijri SM (2019) Smarter traffic prediction using big data, in-memory computing, deep learning and GPUs. Sensors. 19(9): 2206
18. Du B, Peng H, Wang S, Bhuiyan MZA, Wang L, Gong Q et al (2019) Deep irregular convolutional residual LSTM for urban traffic passenger flows prediction. IEEE Transactions on Intelligent Transportation Systems
19. Matthews DG, Alexander G, Van Der Wilk M, Nickson T, Fujii K, Boukouvalas A et al (2017) GPflow: A Gaussian process library using TensorFlow. J Mach Learning Res 18(1):1299–1304
20. Wongsuphasawat K, Smilkov D, Wexler J, Wilson J, Mané D, Fritz D et al (2018) Visualizing dataflow graphs of deep learning models in TensorFlow. IEEE Trans Vis Comput Graph 24(1):1–12
21. Boyer V, El Baz D, Elkihel M (2012) Solving knapsack problems on GPU. Comput Oper Res 39(1):42–47
22. Fioretto F, Pontelli E, Yeoh W, Dechter R (2018) Accelerating exact and approximate inference for (distributed) discrete optimization with GPUs. Constraints. 23(1):1–43
23. Van Luong T, Melab N, Talbi EG (2013) GPU computing for parallel local search metaheuristic algorithms. IEEE Trans Comput 62(1):173–185
24. Zheng Y (2015) Trajectory data mining: an overview. ACM Trans Intell Sys Technol (TIST) 6(3):29
25. Liu D, Weng D, Li Y, Bao J, Zheng Y, Qu H et al (2017) Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations. IEEE Trans Vis Comput Graph 23(1):1–10
26. Zhan X, Zheng Y, Yi X, Ukkusuri SV (2017) Citywide traffic volume estimation using trajectory data. IEEE Trans Knowl Data Eng 29(2):272–285
27. Ren R, Hung T, Tan KC (2018) A generic deep-learning-based approach for automated surface inspection. IEEE Trans Cybernetics 48(3):929–940

28. Sun M, Zhou Z, Hu Q, Wang Z, Jiang J (2018) SG-FCN: A motion and memory-based deep learning model for video saliency detection. IEEE Trans Cybernetics
29. Ding W, Lin CT, Cao Z (2018) Deep neuro-cognitive co-evolution for fuzzy attribute reduction by quantum leaping PSO with nearest-neighbor memeplexes. IEEE Trans Cybernetics (99):1–14
30. Gong YJ, Li JJ, Zhou Y, Li Y, Chung HSH, Shi YH et al (2016) Genetic learning particle swarm optimization. IEEE Trans Cybernetics 46(10):2277–2290
31. Luo J, Gupta A, Ong YS, Wang Z (2018) Evolutionary optimization of expensive multiobjective problems with co-sub-pareto front gaussian process surrogates. IEEE Trans Cybernetics
32. Wang H, Jin Y, Sun C, Doherty J (2018) Offline data-driven evolutionary optimization using selective surrogate ensembles. IEEE Transactions on Evolutionary Computation
33. Djenouri Y, Belhadi A, Fournier-Viger P, Lin JCW (2017) An hybrid multi-core/gpu-based mimetic algorithm for big association rule mining. In: International Conference on Genetic and Evolutionary Computing. Springer, pp 59–65
34. Djenouri Y, Belhadi A, Fournier-Viger P, Fujita H (2018) Mining diversified association rules in big datasets: A cluster/GPU/genetic approach. Information Sciences 459:117–134

Asma Belhadi obtained the PhD in Computer Engineering from the University of Science and Technology USTHB Algiers, Algeria, in 2016. She is working on topics related to artificial intelligence and data mining, with focus on logic programming. Dr Asma Belhadi participated in many international conferences worldwide, and she has been granted short-term research visitor internships to many renowned universities including IRIT in Toulouse. She has published more than 30 refereed research articles in the areas of artificial intelligence.



Youcef Djenouri obtained the PhD in Computer Engineering from the University of Science and Technology USTHB, Algiers, Algeria, in 2014. He is currently research scientist at SINTEF, Oslo, Norway. He is working on topics related to artificial intelligence and data mining, with focus on association rules mining, frequent itemsets mining, parallel computing, swarm and evolutionary algorithms and pruning association rules. Dr Youcef Djenouri participated in many international conferences worldwide, He has published more than 70 papers in the areas of data mining, parallel computing and artificial intelligence.

Djamel Djenouri obtained the PhD in Computer Science from the University of Science and Technology (USTHB), Algiers, Algeria, in 2007. From 2008 to 2009, he was granted a post-doctoral fellowship from the European Research Consortium on Informatics and Mathematics (ERCIM), and he worked at the Norwegian University of Science and Technology (NTNU), Norway. He is currently an Associate Professor at the University of the West England in Bristol. He is working on topics related Internet of things, wireless and mobile networks, machine learning and application for smart cities and green applications. He has been conducting several research projects with international collaborations as the principal investor for many of them. He participated in many international conferences worldwide and gave many keynotes and plenary-session talks. He published more than 100 papers in international peer-reviewed journals and conference proceedings, two books, and he is holding two national patents.

Jerry Chun-Wei Lin received his Ph.D. in Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2010. He is now working as an associate professor at Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway. His research interests include data mining, privacy-preserving and security, Big Data analytics, and social networks. He has published more than 200 research papers in peer-reviewed international conferences and journals, which have received more than 4000 citations. He is the co-leader of the popular SPMF open-source data-mining library and the Editor-in-Chief of Data Mining and Pattern Recognition (DSPR), Associate Editor of Journal of Internet Technology and IEEE Access, and a member of the Editorial Board of Intelligent Data Analysis.