Crowd Forecasting based on WiFi Sensors and LSTM Neural Networks

Utkarsh Singh, Jean-François Determe, François Horlin, and Philippe De Doncker

Abstract—**To ensure effective management and security in large scale public events, it is imperative for the event organizers to be aware of potentially critical crowd densities. This paper, therefore, presents a solution to the above problem in terms of WiFi based crowd counting and LSTM neural network based forecasting. Monitoring of an actual event organized in Brussels has been described, wherein crowd counts are obtained using WiFi sensors in a privacy-preserved manner. The time-stamped crowd counts are used to develop univariate time-series, which are in-turn utilized for forecasting. Five different LSTM models are utilized for crowd time-series forecasting and analyzed for their suitability. A random walk model is used as reference for performance assessment. Among different LSTM models, Convolutional LSTM delivered the best performance. Overall results and analysis show that the developed system is suitable for crowd monitoring.**

Index Terms—**Crowd monitoring, deep learning, forecasting, long-short term memory, neural networks, time series, wireless sensor networks.**

I. Introduction

OVERCROWDING is a common phenomenon observed during major events such as concerts, festivals, sports, games, and entertainment. There is a potential risk to public safety during such large scale public events. The love parade disaster in Germany [1] and the Turin stampede [2] are significant examples of crowd disasters. In view of such mishaps, the event organizers must be able to monitor and predict the crowd behaviour. Apart from safety concerns, crowd monitoring can also help the organizers to effectively plan the venue layout (e.g. food points, shopping points, entries and exits), and prepare crowd management strategy (e.g. restricting access, evacuation, and additional installations).

A. Overview of the Motivation

Conventional research in crowd monitoring mainly focuses on systems with computer vision approach [3]. Factors such as occlusion, huge data handling, cost of implementation and privacy concerns inhibit the possibility of using such systems for real-time situational awareness [4]. These shortcomings led to the development of alternative crowd monitoring strategies using cellphone, global positioning system (GPS), Bluetooth and WiFi [5]-[8]. An insight into these technologies reveals that, the WiFi sensors are not only free from occlusion, but are also non-intrusive and cost effective [8], [9]. Successful applications of wireless sensor networks in sensing and monitoring have already been shown for: pedestrian counting in urban traffic [10], industrial process automation [11], landslide monitoring [12], and distributed environmental monitoring [13]. Motivated from such applications, this paper contributes to the existing knowledge by utilizing WiFi sensors for not only estimating but also forecasting the crowd count in large scale public events. More specifically, this paper discusses, how to prepare time series of crowd counts derived from WiFi measurements and utilize it for achieving multi-step ahead crowd count predictions via long short-term memory (LSTM) neural network. These predictions can ultimately help in better management of large scale public events.

B. Related Works

Probably the most popular WiFi based crowd counting approach is based on channel state information (CSI) [8], [14], [15]; which quantifies the impact of crowd density on CSI. Though this approach has the merit of being non-intrusive, it does not assert the technical aptness for large scale public events, where the crowd density surges during peak hours. Another crowd counting method relies on direct measurement of WiFi received signal strength indicator (RSSI). Authors in [16] have used this approach for occupancy estimation in indoor and outdoor environments. Only 9 people were considered to validate the effectiveness of this technique. An interesting development is presented in [17], where WiFi linkage blockage patterns are classified using LSTM for crowd counting. It remains to be seen whether the system can perform equally well in outdoor environments with large and dense crowd. A more robust way of crowd count estimation is to utilize probe requests (PRs). PRs help a smartphone (or any WiFi enabled device) in searching nearby access points (APs). Authors in [9] and [10] have used this method to estimate pedestrian density /count. These works show that a PR contains device address and AP address. In PR based crowd counting, it is important to be aware of variable PR frequency in devices. As mentioned in [18] and [19], it can be tackled by having two or more sensors separated by up to 100 meters for efficient crowd counting.

This work is a part of the MUFINS project – funded by INNOVIRIS. The authors thank Brussels Major Events (BME) for the collaboration.

All authors are with the OPERA Wireless Communications Group, Université libre de Bruxelles, 1050 Brussels, Belgium. Corresponding author e-mail: utkarsh.singh@ulb.ac.be

Though PRs can provide more accurate crowd count estimation, it could also lead to privacy concerns. Authors in [19], give a solution for preserving privacy in terms of SHA256 encrypted MAC addresses. Weppner et al. [20] used 31 scanners to monitor a motor show for 13 days based on sensing of WiFi /Bluetooth enabled devices. The collected data was utilized only to create heat maps, which does not give a quantitative representation of the crowd. Besides, there is no discussion on the possibility of devices anonymizing their media access control (MAC) addresses, which could lead to duplicate counts [21]. With respect to WiFi based crowd forecasting, there is very limited literature available. Authors in [22] use WiFi signal detectors for crowd count estimation and an autoregressive integrated moving average (ARIMA) model for forecasting. Another work on indoor crowd queuing time estimation is presented in [23]. The authors used WiFi positioning data and nonstandard autoregressive (NAR) model to predict the queuing time. Possibly the only work, which nearly matches the idea of this paper is highlighted in [24]. It vaguely presents the idea of crowd count forecasting based on Bluetooth data for a public event.

Although the purview of this paper is limited to crowd forecasting, it is important to highlight the key works in localizing crowd using WiFi probes. Authors in [25], present a large-scale pedestrian monitoring system, where the importance of scanning and hopping time is discussed to avoid packet loss and increase localization accuracy. Potorti et al. [26], present a fingerprint interpolation based approach and conclude that WiFi probes are viable and economic for crowd localization. A discussion on range-based and range-free methods and the challenges in indoor localization is also presented. Another work on ranging and positioning based on WiFi sensing, is presented in [27]. The authors manage to achieve an accuracy within 2.5m in 80% cases. An experimental study on the usage of WiFi probes for user localization, profiling, and device classification is reported in [28]. The authors have performed tests in both indoor and outdoor environment, leading to three crowd databases. The inter-probing times for Apple, Motorola, and Samsung devices have also been analyzed. A very well elaborated case study of WiFi based crowd localization in TU Delft Campus in presented in [29]. The work presents useful insights on sensor setup/ calibration, occupancy estimation, positioning accuracy/ latency, movement patterns in indoor environments.

C. *Proposed Approach*

This work proposes the crowd count estimation based on PRs collected by WiFi sensors. Note that, crowd density and crowd counts can be used interchangeably because crowd density can be easily converted into crowd count for a fixed area. Although WiFi sensors face attenuation problem, it is not really difficult to estimate their operational range and allocate requisite number of sensors for proper monitoring. Aggregating the counts from all sensors gives an approximation of the crowd count in a monitored area. To account for the people without WiFi enabled devices, the aggregated crowd count is multiplied by a pre-assessed extrapolation factor to give the net crowd count. Received signal strength indicator (RSSI) values is used to ensure that each device is counted only once. To ensure privacy-preserved crowd counting, the MAC addresses in the received PRs are anonymized by default. For each area, a univariate time-series is prepared using time-stamped crowd counts. It is true that a multivariate case (i.e. forecasts with additional variables such as social media response, weather, traffic, etc.) might help in giving better crowd forecasts. But getting additional data can require further permissions and multi-party dependency. The objective of this work is to present a self-sufficient crowd monitoring system and therefore only WiFi based univariate case is presented.

In large-scale public events lasting several days, generally the granularity of forecast horizon is taken in minutes. Under such conditions, short-term fluctuations and unpredictable behaviour can make a time series hard to predict via conventional methods. Given the abundance of data in such events, LSTM becomes a suitable choice for forecasting [30]. Five variants of LSTM have been used to make 30 minutes ahead crowd count forecasts. The purpose is to make event organizers aware of critical crowd density well in advance. The proposed methods neither use nor identify trend and seasonality in the time series. This ensures that the proposed methods are generalized and apply to the widest class of events. Since the proposed forecasting techniques rely on time series, they can be easily employed in any other crowd monitoring technique which is capable of producing time-stamped crowd counts. The novelty and importance of this work is marked by the following key contributions: (i) the proposed crowd forecasting is based on real measurements taken from an actual event; (ii) a large scale public event has been monitored instead of transportation systems or indoor crowd; and (iii) a first-hand application of LSTM is presented for crowd forecasting, with five different variants.

D. *Paper Layout*

This paper is organized as follows: Section II describes the crowd counting mechanism. Section III presents the Winter Wonders festival, which was used for data collection. Section IV offers an overview of LSTM based crowd forecasting. Section V presents the results and discussion for crowd forecasting. The paper concludes in Section VI, giving some important future directions.

II. Crowd Counting via WiFi Sensors

A. *Probe Requests (PRs) Detection using WiFi Sensors*

This paper is written from application perspective and therefore only the measurement methodology is described hereafter. The crowd counts required for forecasting are derived on the basis of PR, which allows a WiFi enabled device to seek and connect to a WiFi access point (AP) in its vicinity [9]. Each PR frame, contains the source address of the sending device, which is basically its MAC address (Refer [10, Fig. 2]). Therefore, counting the PRs indirectly accounts for the number of people present at an event. To avoid overestimation, it must be ensured that the PRs with same MAC address are counted

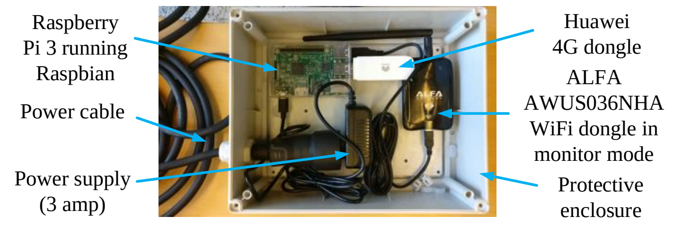

Raspberry Pi 3 running Raspbian
Power cable
Power supply (3 amp)
Huawei 4G dongle
ALFA AWUS036NHA WiFi dongle in monitor mode
Protective enclosure

Fig. 1. Setup of the WiFi sensor: All components are glued inside a robust protective enclosure. During deployment, the sensors are mounted on poles using zip-ties such that the antenna of WiFi dongle is vertically positioned. 4G dongle provides the internet connection for transferring the anonymized probe requests from Raspberry Pi to a central server. Power supply is accessed via sufficiently long cables from a nearby source.

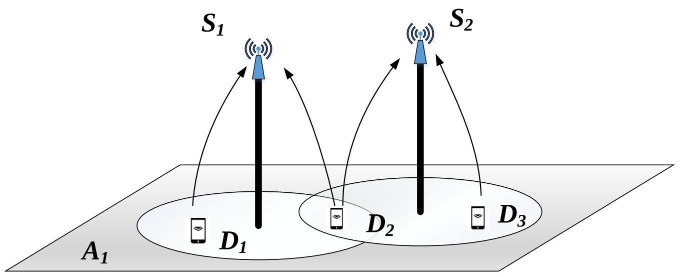


Fig. 2. Deployment and detection: Pole mounted sensors S_1 and S_2 detect the PRBs emanating from users/ devices D_1, D_2, and D_3 in area A_1. The detection range of each sensor is depicted by an ellipse and arrows represent the detection of PRBs. Devices D_1 and D_3 are respectively detected by their nearest sensors, i.e. S_1 and S_2. Being in a common zone, D_2 is detected by both S_1 and S_2.

just once. PRs are generally sent as several bursts within a small time frame [31], which can be referred to as probe request bursts (PRBs). In this work, WiFi sensors were developed, which are capable of collecting and sending PRBs to a central server. Setup of the WiFi sensor with description of the components is shown in Fig. 1. Often the operational range of two or more sensors overlap. This is a desirable situation, because it is hard to precisely determine the range of a sensor. The operational range of a WiFi sensor can vary based on crowd density, because the signals are attenuated by human body. A high density of sensors ensures that the area of interest is fully covered. A typical deployment and detection scenario is explained in Fig. 2. Overlapping range implies that several sensors may detect identical PRBs. To avoid duplicate counting, data from all sensors are jointly processed on a central server.

B. *Processing, Counting, and Time Series Preparation*

Consider n_S sensors deployed in n_A areas at an event, where each sensor and area is represented by their respective IDs: S_i and A_j, where $i \in [1, n_S]$; $j \in [1, n_A]$; $n_S, n_A \in \mathbb{N}$; and $n_A \leq n_S$. The objective of the processing is to generate a time series with crowd counts every $T_i = 5$ minutes, corresponding to each area A_j. The first step consists of computing counts for disjoint and contiguous elementary time frames of duration $T_e = 30$ seconds. For each time frame, all the corresponding anonymized MAC addresses are extracted from the database. This results in an array of 3-tuples. The kth 3-tuple is represented as $\left(S^{(k)}, \text{aMAC}^{(k)}, \text{RSSI}^{(k)}\right)$. Here, $S^{(k)}$ is a sensor ID S_i, $\text{aMAC}^{(k)}$ is the corresponding anonymized MAC

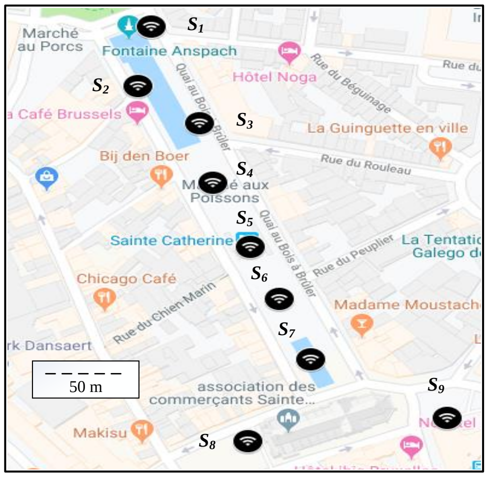


Fig. 3. Map of *Sainte-Catherine* area showing approximate sensor locations. This figure has been generated using 'My Maps' tool from Google.

address, and $\text{RSSI}^{(k)}$ is the corresponding RSSI (a number quantifying the received power by sensor $S^{(k)}$ in dBm). It must be noted that, $\text{aMAC}^{(k)}$ does not reveal the actual MAC address of the device corresponding to detected PRB, because it is anonymized for preserving privacy. The number of distinct PRBs are counted within each elementary time frame per sensor. If a PRB is detected by more than one sensor, it is counted only for the nearest sensor (i.e., the one which reports highest RSSI). With this type of processing, the total crowd count per area can be simply obtained by adding the counts from all sensors within that area. To prepare a time series, counts of ten corresponding elementary time frames are averaged to generate counts at every $T_i = 5$ minutes. Averaging counts over five minutes also alleviates the measurement noise. The entire process relies on PRB transmission within a short time frame ($T_e = 30$ seconds). It does not intend long term tracking of devices /users by deanonymizing the received PRs [32]. In this way, the proposed crowd counting methodology is *immune to MAC address randomization*. To account for the people who may not have a WiFi enabled device, the counts are multiplied by an *extrapolation factor* to obtain the net crowd count. The extrapolation factor is obviously influenced by the number of people who have their WiFi disabled or those who might not have smartphones. Therefore, the extrapolation factor was obtained by comparing crowd counts generated by WiFi sensors, against those provided by a cellular network, in a previous event. An extrapolation factor of 3 was found suitable for large scale public events.

III. Event Description

A large scale public event – Winter Wonders, was organized in the city of Brussels (Belgium) between November 30, 2018 and January 6, 2019. This work focuses on the two most crowded areas of this event, namely *Sainte-Catherine* and *Bourse*. Maps of these areas are respectively shown in Fig. 3 and Fig. 4. Both the areas had arrangements for food and leisure, with almost similar setup. These areas are in the center of Brussels city, with nearby metro and bus stations serving as the major points of crowd influx. The time-series for *Sainte-*

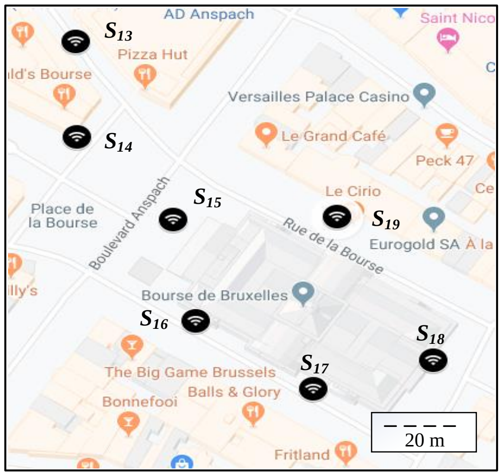


Fig. 4. Map of *Bourse* area showing approximate sensor locations. This figure has been generated using 'My Maps' tool from Google.

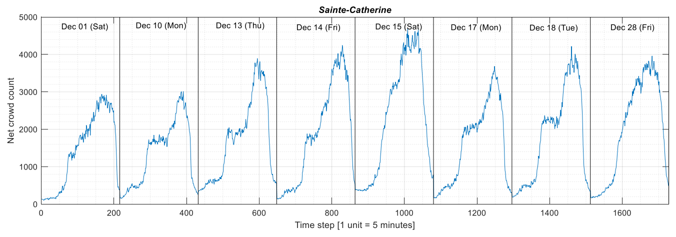


Fig. 5. Time series (ground truth) for *Sainte-Catherine* area.

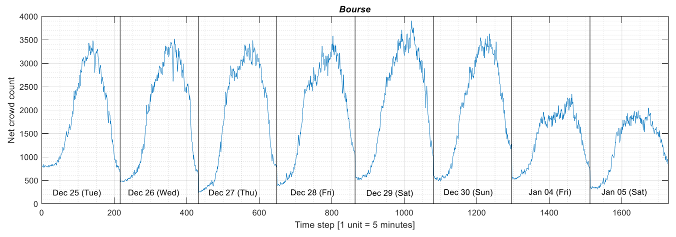


Fig. 6. Time series (ground truth) for *Bourse* area.

Catherine area, obtained by aggregating the crowd counts from sensors S_1 to S_9, is shown in Fig. 5. For *Bourse*, the time series is obtained by aggregating crowd counts from sensors S_{13} to S_{19}, and is shown in Fig. 6. The readers are requested to zoom in the figures for clarity. Only those days are shown, when all the sensors were continuously online in the respective areas. Each day has 216 time steps between the time range of interest (6 AM to 12 AM). The variations in crowd counts can be perfectly captured in the mentioned time range, and therefore, forecasting results will also be shown for the same.

IV. Crowd Forecasting with LSTM

A. Description of LSTM

LSTM is a type of recurrent neural network which propagates or forgets information over a long and recurrent training period, so as to improve the prediction performance. The capability to correlate between the previous and current information makes it a suitable candidate for time series forecasting. The basic unit in LSTM modeling is known as a cell. Fig. 7 shows the architecture of an LSTM cell, which is the most basic unit of an

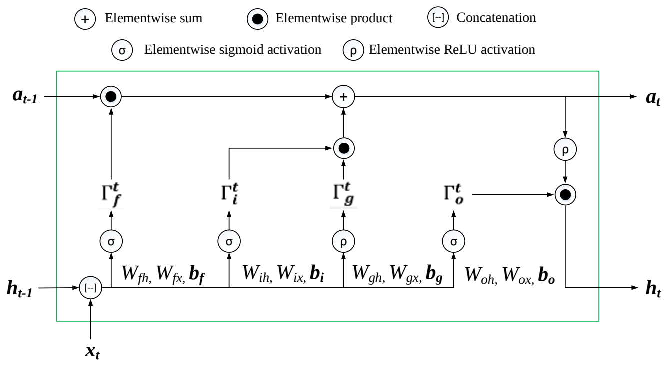


Fig. 7. LSTM Cell structure. All vectors are represented in bold. Weight matrices are symbolized by $\boldsymbol{W}$.

LSTM neural network. All bold notations henceforth represent vector quantities. Let $\boldsymbol{x_t}$ be a sequence vector, where sample index $t = 1, 2, \ldots T$ and T is the total time samples in the sequence. At each index $\boldsymbol{t}$, LSTM takes an input sample from $\boldsymbol{x_t}$, past cell state $\boldsymbol{a_{t-1}}$, and past hidden state $\boldsymbol{h_{t-1}}$. The learning of temporal relations in LSTM is defined by the following equations [33]:

$$\Gamma_{\boldsymbol{f}}^{\boldsymbol{t}} = \sigma\left(W_{fh}\boldsymbol{h_{t-1}} + W_{fx}\boldsymbol{x_t} + \boldsymbol{b_f}\right) \tag{1}$$

$$\Gamma_{\boldsymbol{i}}^{\boldsymbol{t}} = \sigma\left(W_{ih}\boldsymbol{h_{t-1}} + W_{ix}\boldsymbol{x_t} + \boldsymbol{b_i}\right) \tag{2}$$

$$\Gamma_{\boldsymbol{g}}^{\boldsymbol{t}} = \rho\left(W_{gh}\boldsymbol{h_{t-1}} + W_{gx}\boldsymbol{x_t} + \boldsymbol{b_g}\right) \tag{3}$$

$$\Gamma_{\boldsymbol{o}}^{\boldsymbol{t}} = \sigma\left(W_{oh}\boldsymbol{h_{t-1}} + W_{ox}\boldsymbol{x_t} + \boldsymbol{b_o}\right) \tag{4}$$

$$\boldsymbol{a_t} = \Gamma_{\boldsymbol{f}}^{\boldsymbol{t}} \odot \boldsymbol{a_{t-1}} + \Gamma_{\boldsymbol{i}}^{\boldsymbol{t}} \odot \Gamma_{\boldsymbol{g}}^{\boldsymbol{t}} \tag{5}$$

$$\boldsymbol{h_t} = \Gamma_{\boldsymbol{o}}^{\boldsymbol{t}} \odot \rho(\boldsymbol{a_t}) \tag{6}$$

Here, W_{fh}, $W_{fx}, W_{ih}, W_{ih}, W_{gh}, W_{gh}, W_{oh}, W_{oh}$, are the weight matrices and $\boldsymbol{b_f}$, $\boldsymbol{b_i}$, $\boldsymbol{b_g}$, $\boldsymbol{b_o}$ are the bias vectors which lead to the respective resultant vectors of $\Gamma_{\boldsymbol{f}}^{\boldsymbol{t}}$, $\Gamma_{\boldsymbol{i}}^{\boldsymbol{t}}$, $\Gamma_{\boldsymbol{g}}^{\boldsymbol{t}}$, $\Gamma_{\boldsymbol{o}}^{\boldsymbol{t}}$. The subscripts $\boldsymbol{f}$, $\boldsymbol{i}$, $\boldsymbol{g}$, and $\boldsymbol{o}$ respectively represent the forget gate, input gate, input node, and output gate. Symbol '$\odot$' represents elementwise product. In (1)-(6), the dimension of all vector quantities is $T \times 1$ and the weight matrices is $T \times T$, where T represents total time samples in input sequence. Cell state is the memory of an LSTM cell, whereas hidden state is virtually its output. Before the operations inside an LSTM cell can be deciphered, it is important to understand the role of activation functions. In this work, sigmoid and rectified linear unit (ReLU) activation functions are used. The expression for sigmoid activation is given as $\sigma(z) = \frac{1}{1+e^{-z}}$ which gives an output in the range $(0, 1)$ for any input z. The expression for the ReLU is given as $\rho(z) = \max(z, 0)$. The sigmoid activation is used in the input, output, and forget gate; where its output decides whether an information should be propagated (values closer to 1) or rejected (values closer to 0). Training of LSTM involves gradient computation, which may lead to vanishing gradient problem if the gradients shrink to zero (see [34] for further details). This problem is solved by the choice of ReLU activation, whose gradients are faster to compute and do not vanish easily [35]. From Fig. 7, it can be seen that the forget gate decides what information to omit from $\boldsymbol{h_{t-1}}$ and $\boldsymbol{x_t}$. This resulting vector $\Gamma_{\boldsymbol{f}}^{\boldsymbol{t}}$ (see (1)) contains values in (0,1) which helps in omitting irrelevant values from cell state $\boldsymbol{a_{t-1}}$ via elementwise product (see (5)). Using a sigmoid activation, the input gate decides the indices at which new information should

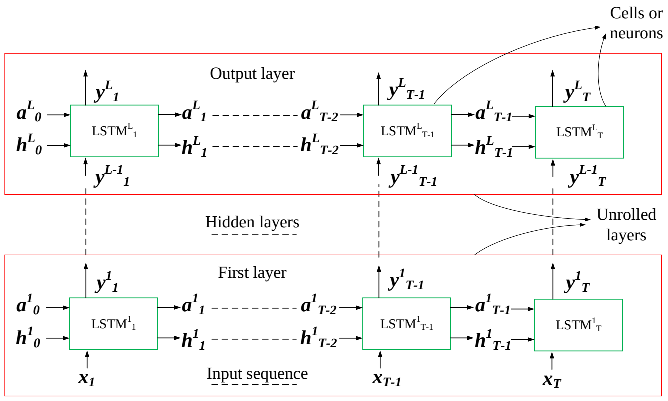


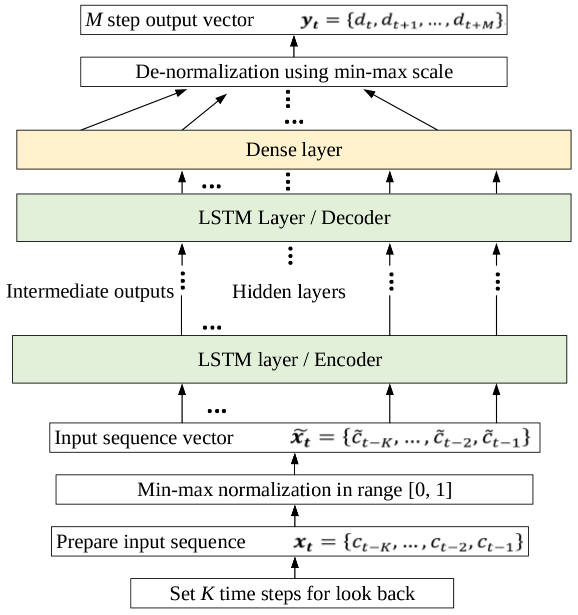
Fig. 8. Unrolled and layered representation of LSTM. Rolled over $\boldsymbol{T}$ time samples of time sequence $\boldsymbol{x_t}$, depicting $\boldsymbol{L}$ layers of LSTM with hidden layers in between. For example, $\mathbf{LSTM}_{\boldsymbol{T}}^{\mathbf{1}}$ represents $\boldsymbol{T}$th cell in layer 1, which has past cell state $\boldsymbol{a}_{\boldsymbol{T-1}}^{\mathbf{1}}$ and past hidden state $\boldsymbol{h}_{\boldsymbol{T-1}}^{\mathbf{1}}$ with present input $\boldsymbol{x_T}$, giving the output $\boldsymbol{y}_{\boldsymbol{T}}^{\mathbf{1}}$ to subsequent cell in layer 2. An LSTM model can have any number of cell (neurons) or layers, as required by the problem in hand.

be added, which leads to representative vector Γ_i^t (see (2)). The ReLU activation in input node, gives the new values to be added in terms of vector Γ_g^t (see (3)). The resultant of elementwise product of Γ_i^t and Γ_g^t contains new values which are added to $\Gamma_f^t \odot \boldsymbol{a_{t-1}}$, resulting in an updated cell state $\boldsymbol{a_t}$ (see (5)). Finally, the output gate passes a filtered or relevant values form the updated cell state $\boldsymbol{a_t}$, as the new hidden state $\boldsymbol{h_t}$. The values to be passed in the new hidden state $\boldsymbol{h_t}$ are obtained by passing updated cell state $\boldsymbol{a_t}$ through ReLU activation, which gives $\rho(\boldsymbol{a_t})$. The locations of the updated cell state vector which will hold the filtered values are decided by sigmoid activation (see (4)), resulting in vector Γ_o^t. The new hidden state $\boldsymbol{h_t}$ is finally obtained by elementwise product of Γ_o^t and $\rho(\boldsymbol{a_t})$ (see (6)).

The handling of a time series can be understood by the unrolled representation of an LSTM at subsequent time samples as shown in Fig. 8. From the application perspective the cells are referred to as neurons. Vertical stacking of time-unrolled cells represents the LSTM layers. Fig. 8 shows the T samples of the unrolled sequence and also L different layers which can be stacked on top of each other for deep learning. Deep learning symbolizes automatic feature learning capability of the LSTM which enables it to enhance its predictive performance. For example, it can read a crowd density time series and learn the time dependent variations in crowd density using several LSTM layers to predict the crowd density for a desired time horizon. An LSTM can have several intermediate layers which are known as the hidden layers. As seen in Fig. 8, each cell passes an output $\boldsymbol{y_t}$ to the subsequent layer. This output is essentially the hidden state $\boldsymbol{h_t}$, as mentioned earlier. If LSTM inputs are encoded, then $\boldsymbol{h_t}$ is passed through another activation function to obtain the output $\boldsymbol{y_t}$. If inputs are not encoded, then the hidden state is simply passed as the output, i.e. $\boldsymbol{h_t} = \boldsymbol{y_t}$.

B. Forecasting Methodology

LSTM is considered suitable for time series forecasting because of sequential input support, mapping of input data to vector output for multistep ahead predictions, and effective modeling of non-stationary and non-linear processes. It is a common practice to remove non-stationarity in time series through various transformations, which removes trends and seasonality [36]. However, in this paper, such transformations shall not be used, because LSTM has been shown to perform well even without accounting for non-stationarity [30], [33], [37]. A general model for LSTM based multistep ahead forecasting is shown in Fig. 9. The objective of forecasting in this paper, is to obtain M future crowd counts given K past crowd counts. A sequence vector of crowd counts is created with past K values, represented as $\boldsymbol{x_t} = \{c_{t-K}, \dots, c_{t-2}, c_{t-1}\}$. LSTM computations are sensitive to data scale, therefore the sequence vector must be normalized in the range [0, 1]. Min-max normalization is used for this purpose and the resulting vector is represented as $\boldsymbol{\tilde{x}_t} = \{\tilde{c}_{t-K}, \dots, \tilde{c}_{t-2}, \tilde{c}_{t-1}\}$. This normalized vector is fed to the first LSTM layer. Inside the LSTM, the sequence vectors are utilized to train weights and biases, which will in-turn help in making predictions. During training, the weights are initialized randomly and the biases are initialized as one to prevent information loss. The initialization of cell state and hidden state vectors is generally done with zero. After the training is finished, the output of LSTM is passed through one or more dense layers. An LSTM dense layer is a fully connected layer which links each input neuron to the output neuron, just like in a conventional neural network. In simpler terms, it helps in mapping an input of different size to an output of desired size. The output received from dense layer is normalized and needs to be inverted to its min-max scale. The scale inverted output vector gives the M step ahead predictions, which can be represented as $\boldsymbol{y_t} = \{d_t, d_{t+1}, \dots, d_{t+M}\}$. This paper presents 6-step (30-minutes) ahead crowd forecasts, therefore the output sequence length is chosen as $M = 6$ samples. The size of input sequence can be flexible. An input sequence of size $K = 12$ samples (twice that of output size) is chosen in this work, based on trial and error method.



M step output vector $\boldsymbol{y_t} = \{d_t, d_{t+1}, \dots, d_{t+M}\}$
De-normalization using min-max scale
Dense layer
LSTM Layer / Decoder
Intermediate outputs
Hidden layers
LSTM layer / Encoder
Input sequence vector $\boldsymbol{\tilde{x}_t} = \{\tilde{c}_{t-K}, \dots, \tilde{c}_{t-2}, \tilde{c}_{t-1}\}$
Min-max normalization in range [0, 1]
Prepare input sequence $\boldsymbol{x_t} = \{c_{t-K}, \dots, c_{t-2}, c_{t-1}\}$
Set K time steps for look back

Fig. 9. General model for LSTM based forecasting. $\boldsymbol{M}$-step ahead predictions are obtained using preceding $\boldsymbol{K}$ samples.

Five different variants of LSTM are used in this paper: (i) Vanilla LSTM (VLSTM), (ii) Bidirectional LSTM (BiLSTM),

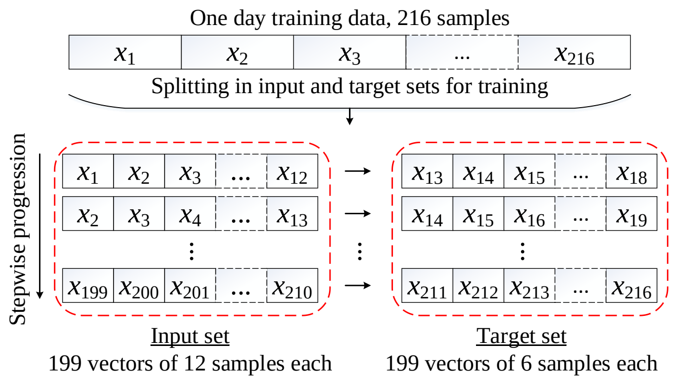

One day training data, 216 samples
x_1 x_2 x_3 ... x_{216}
Splitting in input and target sets for training
Stepwise progression
x_1 x_2 x_3 ... x_{12}
x_2 x_3 x_4 ... x_{13}
x_{199} x_{200} x_{201} ... x_{210}
x_{13} x_{14} x_{15} ... x_{18}
x_{14} x_{15} x_{16} ... x_{19}
x_{211} x_{212} x_{213} ... x_{216}
Input set
199 vectors of 12 samples each
Target set
199 vectors of 6 samples each

Fig. 10. An insight into LSTM Training: Training data of 216 samples is restructured such that an input window of 12 samples targets an output window of next 6 samples. LSTM learns to predict for the chosen horizon of (6 samples×5minutes) 30 minutes. This results in 199 training sets [input + target] without hindering temporal dependency in the time series data. A collection of 18 training sets forms one batch.

(iii) Encoder-Decoder LSTM (EDLSTM), (iv) Convolutional Neural Network LSTM (CNNLSTM), (v) Convolutional LSTM (ConvLSTM). VLSTM is the simplest variant of LSTM which has only one LSTM layer which is immediately followed by a dense output layer to make predictions. BiLSTM consists of 2 layers in opposite directions connected to same output. EDLSTM is a variant which is specially used for sequence to sequence prediction problems. The length of input and output sequences in EDLSTM can be same or different, as per the requirement of forecasting problem. Mapping between input and output vectors of different length is done using coded vector representations. The CNNLSTM and ConvLSTM variants involve convolutional neural networks for input (time series) feature extraction, followed by LSTM for sequence prediction. The only difference between CNN LSTM and ConvLSTM is that, the latter has convolutional reading of input is built directly into LSTM layer. The objective of this paper is only to apply and compare the performance of mentioned LSTM variants for crowd forecasting. Due to length restrictions, it is not possible to describe each variant in detail. Therefore, the readers are kindly requested to follow references [38]-[41], which elucidate the architecture and modeling of the LSTM variants.

C. *Hyperparameter Selection*

Hyperparameters are those parameters whose values are chosen before training a machine learning model. Obtaining the optimal hyperparameters is a computation intensive process. Moreover, it is contextual and may not be optimal for different problems. Therefore, this paper will only follow the best practices in LSTM hyperparameter selection. Similar approach has been used in [33], [37]. The aim is to obtain forecasting models which can perform well across different test cases. The choice of parameters common to all models is given as follows:
Batch Size: A batch is the number of samples of training set needed to update the coefficients, i.e., weights and biases. Based on trial and error approach, a batch size of 18 is chosen.
Epochs: It is number of times an overall training set is passed through the network. Based on trial and error approach, 70 epochs were found suitable in this work.
Optimizer/ learning rate: Due to its proven performance [33], ADAM optimizer has been used in this work with its default parameters (learning rate = 0.001).
Loss function: Being the most common choice, mean squared error (MSE) is chosen as the loss function in this work.
Neurons or cells: 100 neurons per hidden layer were chosen based on performance analysis in the search space of 20 to 200, as recommended in [42]. Using more neurons may lead to slightly better performance, but the training time is too high.
Hidden layers: Based on the discussion given in [42] and tests performed, single LSTM hidden layer has been chosen in this work apart from the model specific layers. Adding further layers complicates the training without giving better results.

D. *Training*

As seen in Fig. 5 and Fig. 6, *Sainte-Catherine* and *Bourse* time series have 8 days of available data. These days are divided in training and test sets, as shown in Table I. The training and test sets are selected sequentially from the two time series. However, it should not lead to any bias in LSTM forecasts, because the days in time series are not consecutive. An insight in the training procedure is given in Fig. 10. It can be seen how the 216 samples (crowd counts) available from each day of training data are restructured to create 199 training sets, each having an input window of 12 samples and target or output window of 6 samples. An *overfitting check* has been implemented via walk-forward validation, also known as ‘rolling forecast’. In walk-forward validation, LSTM models will be required to make 6-step ahead crowd count forecasts. Thereafter, the actual data will be made available to predict the next 6 steps. Since this method does not provide complete data to the model at once, therefore overfitting is prevented by default. The training process is dynamic in nature because LSTM preserves memory across the training samples. An *underfitting check* is also implemented by comparing the performance metrics of LSTM model against a random walk (RW) baseline model [Section V-A]. If the LSTM models show worse metrics than the baseline model, then it would be considered as underfitting.

V. Crowd Forecasting Results

A. *Metrics and Baseline Model*

Two metrics are chosen in this work for performance assessment, root mean square error (RMSE) and mean absolute percentage error (MAPE). Let $\boldsymbol{x_t}$ be actual crowd count at any time instance $\boldsymbol{t}$ and $\boldsymbol{y_t}$ be the corresponding predicted count, where $\boldsymbol{t}$ varies from 1 to T. Then, the RMSE can be expressed as:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x_t} - \boldsymbol{y_t})^2} \tag{7}$$

From (7), it can be seen that RMSE penalizes large errors because the errors are squared before averaging. Though RMSE is suitable for the situation where large errors are undesirable, its sensitivity to outliers can also lead to misjudgment of a forecasting model. This is where MAPE can be useful. The expression for MAPE is given as:

$$\text{MAPE} = \frac{100\%}{T}\sum_{t=1}^{T}\frac{|\boldsymbol{x_t} - \boldsymbol{y_t}|}{\boldsymbol{x_t}} \tag{8}$$

TABLE I
TRAINING AND TEST SETS

Sainte-Catherine		
Case	Training days	Test Days
1	Dec 01	Dec 10, Dec 13, Dec 14, Dec 15, Dec 17, Dec 18, Dec 28
2	Dec 01, Dec 10	Dec 13, Dec 14, Dec 15, Dec 17, Dec 18, Dec 28
3	Dec 01, Dec 10, Dec 13	Dec 14, Dec 15, Dec 17, Dec 18, Dec 28
Bourse		
Case	Training days	Test Days
1	Dec 25	Dec 26, Dec 27, Dec 28, Dec 29, Dec 30, Jan 04, Jan 05
2	Dec 25, Dec 26	Dec 27, Dec 28, Dec 29, Dec 30, Jan 04, Jan 05
3	Dec 25, Dec 26, Dec 27	Dec 28, Dec 29, Dec 30, Jan 04, Jan 05

Since MAPE focuses on absolute errors ($|\boldsymbol{x_t} - \boldsymbol{y_t}|$), it is less sensitive to outliers as compared to RMSE. Owing to the normalization of the absolute error, it offers a way to minimize relative error in the predictions. Clearly, RMSE and MAPE have different ways to account for errors. But, this would also help in having a balanced performance assessment of the forecasting models. The metrics RMSE and MAPE, do not take into account the sign of the error. Due to the peculiarity of the application it would be interesting to see if the forecasting techniques underestimate or overestimate the crowd count. For this, the average error (AE) will also be reported for each case as follows:

$$\text{AE} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x_t} - \boldsymbol{y_t}) \tag{9}$$

To prove that the LSTM models perform well, their RMSE and MAPE values must be compared against a baseline model and must perform better than it. A random walk (RW), alternatively known as naive persistence model can be used as a baseline for comparison [43], [44]. It works on an assumption that a time series is constant over the required forecast horizon. For an actual value $\boldsymbol{x_t}$ at time sample t, the RW model gives a prediction at time $t + \mathcal{H}$ as $\boldsymbol{y_{t+\mathcal{H}}} = \boldsymbol{x_t}$, where $\mathcal{H}$ is the required time horizon.

B. Results and Discussion

All the simulations are performed in Python (version 3.6) and the figures are plotted using MATLAB (version 9.4). LSTM models have been prepared using Keras library (version 2.2.4) in Python environment. The test cases mentioned in Table I have been considered to check whether the prediction performance of models improve with more training data or not, which generally does happen in deep learning models. Improvement in performance with increase in training days would thus validate the correctness of developed models. Also, consideration of three different cases would help in averaging the performance (metrics) of each model for the purpose of comparison. Best model would be determined on the basis of percentage (%) improvement in average RMSE and MAPE values as compared to RW model. Fig. 5 and Fig. 6 represent the ground truth for respective areas. It should be noted that the reported metrics are computed only in those parts of time series, where the crowd counts have an increasing trend. This is because during early and late hours, the changes in crowd

TABLE II
CROWD COUNT FORECAST ERRORS FOR *SAINTE-CATHERINE* AREA

RMSE values					
Model	Case 1	Case 2	Case 3	Avg.	(+ %) *
RW	272.60	282.16	274.46	276.41	-
VLSTM	265.08	251.92	214.30	243.77	13.39
BiLSTM	201.71	206.95	223.17	210.61	31.24
EDLSTM	212.01	178.94	197.44	196.13	40.93
CNNLSTM	180.91	194.80	206.39	194.03	42.46
ConvLSTM	184.10	200.73	190.18	191.67	**44.21**
MAPE values					
Model	Case 1	Case 2	Case 3	Avg.	(+ %) *
RW	7.16	7.15	6.89	7.07	-
VLSTM	7.30	6.52	5.24	6.35	11.34
BiLSTM	5.26	5.17	5.59	5.34	32.40
EDLSTM	5.52	4.35	4.72	4.86	45.47
CNNLSTM	4.67	4.58	5.20	4.82	46.68
ConvLSTM	4.72	5.00	4.52	4.75	**48.84**

* (+%) represents percentage improvement in average metrics attained by LSTM variants as compared to average metrics of RW model. The average is taken over the three different test cases having different number of training days.

TABLE III
CROWD COUNT FORECAST ERRORS FOR *BOURSE* AREA

RMSE values					
Model	Case 1	Case 2	Case 3	Avg.	(+ %) *
RW	240.16	236.37	229.40	235.31	-
VLSTM	176.48	166.74	158.49	167.24	40.70
BiLSTM	193.85	155.55	146.59	165.33	42.33
EDLSTM	183.01	151.24	149.64	161.30	45.88
CNNLSTM	163.83	148.60	156.06	156.16	50.69
ConvLSTM	162.37	152.99	142.68	152.68	**54.12**
MAPE values					
Model	Case 1	Case 2	Case 3	Avg.	(+ %) *
RW	10.33	10.22	9.97	10.17	-
VLSTM	7.15	6.72	6.28	6.72	51.34
BiLSTM	7.95	6.31	5.84	6.70	51.79
EDLSTM	7.29	5.97	5.95	6.40	58.91
CNNLSTM	6.77	5.73	6.18	6.23	63.24
ConvLSTM	6.49	5.98	5.52	6.00	**69.50**

* (+%) represents percentage improvement in average metrics attained by LSTM variants as compared to average metrics of RW model. The average is taken over the three different test cases having different number of training days.

counts are not critical. The aim is to forecast in range where the crowd builds up until its peak count. The time range of interest for metrics computation in *Sainte-Catherine* area is 4 PM to 8 PM, whereas in *Bourse* area it is 10.30 AM to 4.30 PM.

Table II and Table III respectively show the results for 6-step (30 minutes) ahead crowd count forecasts for *Sainte-Catherine* and *Bourse* area. All reported values are rounded to two decimal places. Average RMSE and MAPE values corresponding to the three test cases are shown. From the RMSE and MAPE values, it is evident that the LSTM models outperform RW model in all cases. Except some deviations in Table II, the metrics generally improve with increase in training data, which confirms that the models tend to perform better with more training data. It can also be seen that the metrics reported for *Sainte-Catherine* are bigger as compared to *Bourse* area. This is obvious, given the fact that crowd counts are much higher and random in *Sainte-Catherine*. High counts in *Sainte-Catherine* area can be

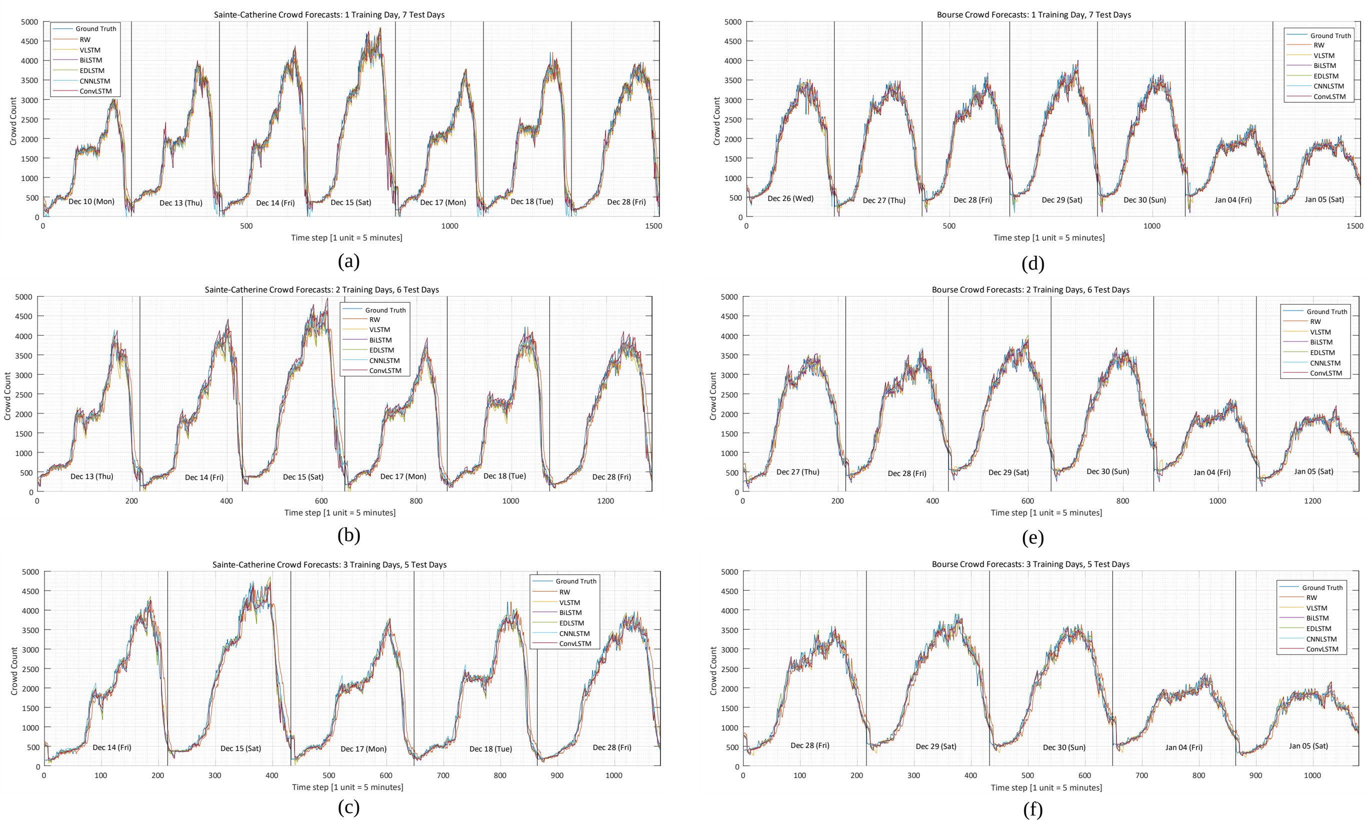


Fig. 11. Comparison of predictions made by different models against ground truth: (a) *Sainte-Catherine* with 1 training day, (b) *Sainte-Catherine* with 2 training days, (c) *Sainte-Catherine* with 3 training days, (d) *Bourse* with 1 training day, (e) *Bourse* with 2 training days, and (f) *Bourse* with 3 training days.

accredited to its bigger size than *Bourse* (compare scales in Fig. 3 and Fig. 4). For both areas, it is observed that the net improvement achieved is more for MAPE values as compared to RMSE values. This signifies that the forecasting models better track the small variations in crowd count as compared to drastic variations. In general, the large and sudden variations in a crowd time series are less predictable. Comparing the LSTM forecasts in *Sainte-Catherine* and *Bourse*, it is seen that the metrics are improved more in *Bourse* area. This is because *Bourse* time series being smoother and thus more predictable (compare Fig. 5 and Fig. 6). This in-turn implies that the LSTM models would not achieve significant improvements as compared to RW model. To comment on individual performances of LSTM models, it can be said that the encoding decoding of time series helps in achieving better predictions. See in Table II and Table III that, EDLSTM, CNNLSTM, and ConvLSTM models report more improvements as compared to VLSTM and BiLSTM. Based on the overall improvement in metrics, ConvLSTM is deemed as the best crowd forecasting model. This could be accredited to its convolutional reading capability, where it creates two or more subsequences from an input sequence to learn temporal dependencies. Overall, all the observations are found to be in close conformity with the expectations. For a quick qualitative analysis, 6-step ahead (30 minutes) predictions achieved by different models are shown in Fig. 11. For further assessment of the forecasts, a summary of average errors (AE) is shown in Table IV for the three training cases of both areas. As expected, the LSTM models report lesser error in general as compared to RW model. Except in two cases, the proposed models underestimate the crowd counts as shown by the positive average error. The overestimations (negative error) in EDLSTM and CNNLSTM may be subjective. The obtained results are favourable in two ways: (i) The event managers prefer underestimations so as not to raise frequent false alarms; and (ii) The average errors seem reasonable, with ConvLSTM giving the most consistent and reliable performance. The total runtime taken in training, testing and computing the metrics for different models are reported in Table V. All computations were performed on Intel i7 1.80 GHz CPU, with 16 GB RAM and Intel 620 Integrated GPU. Since RW model simply reiterates past values, it has least runtime in all cases. In LSTM variants, the runtime is obviously higher and it increases with the increases with increase in size of training data. ConvLSTM shows high runtime owing to the convolutions performed during training.

In general, the runtime can vary from system to system based on hardware specifications. Therefore, time complexity notations are often used to specify the computational requirements of an algorithm. Time complexity of the preferred ConvLSTM model can be obtained by adding the complexities of convolutional layer and LSTM hidden layer. Based on the derivations given in [45], the overall time complexity of ConvLSTM can be interpreted as: $\mathcal{O}\left(\left(\sum_{j=1}^{C}\left(\mathrm{p}_j . \mathrm{q}_j^2 . \mathrm{r}_j . \mathrm{s}_j^2\right)+\boldsymbol{\omega}\right). X. E\right)$. Here, j is convolutional layer number, C is total number of convolutional layer, p is number of input channels, q is spatial size of filter, r is number of filters, s is spatial size of output feature map, $\boldsymbol{\omega}$ is number of weights in hidden LSTM layer, X is input length, and E is number of epochs. In a

TABLE IV
AVERAGE ERRORS (AE) FOR TRAINING CASES IN RESPECTIVE AREAS

Model	*Sainte-Catherine*			*Bourse*		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
RW	168.86	173.27	166.15	166.08	160.14	153.51
VLSTM	204.00	171.52	117.34	77.67	77.74	56.10
BiLSTM	103.41	117.81	136.05	134.76	39.77	29.98
EDLSTM	124.33	-2.84	62.70	116.37	2.34	39.53
CNNLSTM	43.37	-53.45	122.73	75.04	50.17	75.47
ConvLSTM	66.63	78.50	46.46	59.08	43.39	30.41

TABLE V
RUNTIME IN SECONDS FOR TRAINING CASES IN RESPECTIVE AREAS

Model	*Sainte-Catherine*			*Bourse*		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
RW	0.01	0.01	0.01	0.01	0.01	0.01
VLSTM	7.90	15.75	20.62	7.21	13.55	20.56
BiLSTM	8.51	14.92	24.37	9.46	17.83	24.20
EDLSTM	10.29	21.52	30.60	11.92	22.79	32.71
CNNLSTM	7.49	14.67	22.24	8.45	15.86	23.59
ConvLSTM	11.01	23.05	34.23	13.23	23.07	33.88

univariate forecasting scenario, ConvLSTM can be used by reshaping the input into subsequences within a single convolutional layer (kindly refer to [36] for modeling insights). Considering single convolutional layer, the complexity of ConvLSTM can be redefined as: $\mathcal{O}\big((\mathrm{p}.\,\mathrm{q}^2.\,\mathrm{r}.\,\mathrm{s}^2+\boldsymbol{\omega}).X.E\big)$. The obtained expression explains the high runtime of ConvLSTM as compared to the other models. But given its impressive performance, it can be readily given preference over other variants. As reported in [46], the complexity of RSSI based localization process can be given as $\mathcal{O}(\mathcal{N})$, where $\mathcal{N}$ is the number of measurements. Combining the complexities of sensing and forecasting, the overall complexity of the proposed crowd monitoring approach may be given as $\mathcal{O}\big((\mathrm{p}.\,\mathrm{q}^2.\,\mathrm{r}.\,\mathrm{s}^2+\boldsymbol{\omega}).X.E\big)+\mathcal{O}(\mathcal{N})$. This will be effectively equal to $\mathcal{O}\big((\mathrm{p}.\,\mathrm{q}^2.\,\mathrm{r}.\,\mathrm{s}^2+\boldsymbol{\omega}).X.E\big)$ because the complexity of forecasting algorithm is more dominant.

VI. CONCLUSION

This paper presented a first-hand application of WiFi sensors and LSTM for crowd forecasting. A procedure for privacy-preserved crowd counting based on probe request detection was described. A large scale public event (Winter Wonders 2018-2019, Brussels) was monitored to obtain crowd counts and corresponding time series. Five different LSTM based univariate forecasting models were used to make 6-step (30 minutes) ahead predictions on the resulting time series. All LSTM models outperformed the random walk model, which was used as a baseline. ConvLSTM was found to be most suitable model for crowd forecasting. As compared to random walk model, respective improvements of 44.21% and 48.84% were observed in average RMSE and MAPE values for *St. Catherine* area. Corresponding improvements in *Bourse* area were 54.12% and 69.50%. The average errors (AE) reported by ConvLSTM were also found to be most reliable and consistent. It provides most significant improvements in metrics by slightly compromising on the time. Clearly, the forecast performance is different for different areas, which possibly depends on a time series being more or less predictable. Overall, the proposed crowd forecasting system can be deemed suitable for monitoring the large scale public events. In the future work, multivariate crowd forecasting models can be developed by assuming interdependency between crowd counts of two or more areas in proximity or by considering additional features such as social media response, weather, and traffic updates. It can lead to more efficient forecasts and offer more insight into the crowd behaviour.

ACKNOWLEDGMENTS

The authors thank Innoviris for funding this research through the MUFINS project. The authors are also thankful to Brussels Major Events for their active collaboration.

REFERENCES

[1] G. Cardone, A. Cirri, A. Corradi, L. Foschini, R. Ianniello, and R. Montanari, “Crowdsensing in urban areas for city-scale mass gathering management: Geofencing and activity recognition,” *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4185-4195, 2014.

[2] N. Shiwakoti, S. Xiaomeng, and Y. Zhirui, “A review on the performance of an obstacle near an exit on pedestrian crowd evacuation,” *Safety science*, vol. 113, pp. 54-67, 2019.

[3] J. C. S. Jacques, Jr., S. R. Musse, and C. R. Jung, “Crowd analysis using computer vision techniques,” *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 66–77, Sep. 2010.

[4] S. Gong, C. C. Loy, and T. Xiang. “Security and surveillance.” In *Visual analysis of humans*, pp. 455-472. Springer, London, 2011.

[5] F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti, “Real-time urban monitoring using cell phones: A case study in Rome,” *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 1, pp. 141–151, Mar. 2011.

[6] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Troster, “Inferring crowd conditions from pedestrians’ location traces for real-time crowd monitoring during city-scale mass gatherings,” in *Proc. IEEE WETICE*, Jun. 2012, pp. 367–372.

[7] M. Versichele, T. Neutens, M. Delafontaine, and N. Van de Weghe, “The use of bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the Ghent festivities,” *Appl. Geograph.*, vol. 32, no. 2, pp. 208–220, 2012.

[8] W. Xi, J. Zhao, X.-Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang, “Electronic frog eye: Counting crowd using wifi,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 361–369.

[9] A. Kurkcu and K. Ozbay, “Estimating pedestrian densities, wait times, and flows with WiFi and bluetooth sensors,” *Transportation Research Record*, vol. 2644, no. 1, pp. 72–82, 2017.

[10] A. Guillén-Pérez and M. D. C. Baños, “A wifi-based method to count and locate pedestrians in urban traffic scenarios,” in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2018, pp. 123–130.

[11] L. Ascorti, S. Savazzi, G. Soatti, M. Nicoli, E. Sisinni, and S. Galimberti, “A wireless cloud network platform for industrial process automation: Critical data publishing and distributed sensing,” *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 4, pp. 592-603, 2017.

[12] P. Giri, K. Ng, and W. Phillips, “Wireless Sensor Network System for Landslide Monitoring and Warning,” *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 4, pp.1210-1220, 2018.

[13] L. Lombardo, S. Corbellini, M. Parvis, A. Elsayed, E. Angelini, and S. Grassini, “Wireless sensor network for distributed environmental

monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 5, pp.1214-1222, 2017.

[14] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: device-free location-oriented activity identification using finegrained wifi signatures," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 617–628.

[15] S. Liu, Y. Zhao, F. Xue, B. Chen, and X. Chen, "Deepcount: Crowd counting with wifi via deep learning," *arXiv preprint arXiv:1903.05316*, 2019.

[16] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy estimation using only wifi power measurements," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, 2015.

[17] O.T. Ibrahim, W. Gomaa, and M. Youssef, "CrossCount: A Deep Learning System for Device-Free Human Counting Using WiFi," *IEEE Sensors Journal*, vol. 19, no. 21, pp. 9921-9928, 2019.

[18] C. Groba, "Demonstrations and people-counting based on Wifi probe requests," *In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 596-600.

[19] M. Uras, R. Cossu, and L, Atzori, "PmA: a solution for people mobility monitoring and analysis based on WiFi probes," *In 2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2019, pp. 1-6.

[20] J. Weppner, B. Bischke, and P. Lukowicz, "Monitoring crowd condition in public spaces by tracking mobile consumer devices with wifi interface," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. ACM*, 2016, pp. 1363-1371.

[21] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 413–424.

[22] I. K. Tan, O. B. Yaik, and O. B. Sheng, "Predicting shopper volume using ARIMA on public Wi-Fi signals," *International Information Institute (Tokyo). Information*, vol. 19, no. 8A, p. 3295, 2016.

[23] H. Shu, C. Song, T. Pei, L. Xu, Y. Ou, L. Zhang, and T. Li, "Queuing time prediction using WiFi positioning data in an indoor scenario," *Sensors*, vol. 16, no. 11, p. 1958, 2016.

[24] M. Claeys Bouuaert, "Modeling crowds at mass-events: learning large-scale crowd dynamics from Bluetooth tracking data," http://cartogis.ugent.be/mobileghent/sites/default/files/slides/Modeling%20crowds%20at%20mass-events%20-%20MG2013%20-%20MCB.pdf, 2013.

[25] Z. Xu, K. Sandrasegaran, X. Kong, X. Zhu, B. Hu, J. Zhao, and C. Lin, "Pedestrain monitoring system using Wi-Fi technology and RSSI based localization," *International Journal of Wireless & Mobile Networks*, vol. 5, no. 4, pp. 17-34, 2013.

[26] F. Potortì, A. Crivello, M. Girolami, P. Barsocchi, and E. Traficante, "Localising crowds through Wi-Fi probes," *Ad Hoc Networks*, vol. 75, pp.87-97, 2018.

[27] S. Yan, H. Luo, F. Zhao, W. Shao, Z. Li, and A. Crivello, "Wi-Fi RTT based indoor positioning with dynamic weighted multidimensional scaling," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2019, pp. 1-8.

[28] A.E. Redondi and M. Cesana, "Building up knowledge through passive WiFi probes," *Computer Communications*, vol. 117, pp.1-12, 2018.

[29] R. Braggaar, "Wi-Fi network-based indoor localisation: The case of the TU Delft campus," M.S. thesis, TU Delft, Delft, NL, 2018.

[30] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series". In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394-1401.

[31] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating MAC address randomization through timing attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 15–20.

[32] M. Gast, *802.11 wireless networks: the definitive guide*. "O'Reilly Media, Inc.", 2005.

[33] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841-851, 2017.

[34] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp.107-116, 1998.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," In *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[36] V. M. Guerrero, "Time-series analysis supported by power transformations," *Journal of Forecasting*, vol. 12, no. 1, pp. 37–48, 1993.

[37] 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.

[38] M. Schuster, and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[39] S. H., Park, B., Kim, C.M., Kang, C.C., Chung, and J.W., Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," *In IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1672-1678.

[40] C. J., Huang, and P.H., Kuo, "A deep cnn-lstm model for particulate matter (PM2. 5) forecasting in smart cities," *Sensors*, vol. 18, no. 7, p. 2220, 2018.

[41] S. H. I. Xingjian, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," In *Advances in neural information processing systems*, 2015, pp. 802-810.

[42] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232, 2016.

[43] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[44] P. Torres, P. Marques, H. Marques, R. Dionísio, T. Alves, L. Pereira, and J. Ribeiro, "Data analytics for forecasting cell congestion on LTE networks," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2017, pp. 1–6.

[45] E. Tsironi, P. Barros, C. Weber, and S. Wermter, "An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition," *Neurocomputing*, vol. 268, pp.76-86, 2017.

[46] T. Xie, H. Jiang, X. Zhao, and C. Zhang, "A Wi-Fi-Based Wireless Indoor Position Sensing System with Multipath Interference Mitigation," *Sensors*, vol. 19, no. 18, p.3983, 2019.

Utkarsh Singh (S'17, M'18) graduated in electrical and electronics engineering from Uttar Pradesh Technical University, India, in 2012. He completed his master's degree with AICTE fellowship in Power Systems from Thapar University, India, in 2014. He completed his PhD with MHRD fellowship in Power Quality from the Indian Institute of Technology Roorkee in February 2018. In June 2018, he joined OPERA-Wireless Communications Group at Université libre de Bruxelles, Belgium, where he is working on project 'MUFINS' funded by INNOVIRIS. His research interests include artificial intelligence, data analysis, optimization, signal processing and power systems.

Jean-François Determe received the electrical engineering degree (Master en ingénieur civil électricien) from Université libre de Bruxelles (ULB) in 2013.He also jointly received the PhD in Engineering from ULB and Université catholique de Louvain in2018. From 2013 to 2017, he was an FNRS research fellow, funded by the Belgian FRS-FNRS. Since 2018, he has been a postdoctoral researcher with the OPERA-WCG department at ULB and is currently funded by Innoviris. His research interests focus on applied time series analysis and on sparse recovery algorithms.

François Horlin (S'01–M'02) received the Ph.D. degree from the Université Catholique de Louvain (UCL) in 2002. He specialized in the field of signal processing for digital communications. After his Ph.D., he joined the Inter-university Micro-Electronics Center (IMEC). He led the project aiming at developing a 4G cellular communication system in collaboration with Samsung Korea. In 2007, François Horlin became professor at the Université libre de Bruxelles (ULB). He is currently supervising a research team working on next generation communication systems. Localization based on 5G signals, filter bank-based modulations, massive MIMO and dynamic spectrum access are examples of currently investigated research topics.

Philippe De Doncker received the M.Sc. degree in physics engineering and the Ph.D. degree in science engineering from the Université Libre de Bruxelles (ULB), Brussels, Belgium, in 1996 and 2001, respectively. He is currently a Professor with the ULB, where he leads the research activities on wireless channel modeling and electromagnetics.