IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Deep Learning Based Resource Availability Prediction for Local Mobile Crowd Computing

Pijush Kanti Dutta Pramanik[1], Nilanjan Sinhababu[2], Kyung-Sup Kwak[3], Member, IEEE, Prasenjit Choudhury[1]

[1]Dept. of Computer Science & Engineering, National Institute of Technology, Durgapur, India
[2]Reliability Engineering Centre, Indian Institute of Technology Kharagpur, India
[3]Dept. of Information and Communication Engineering, Inha University, South Korea

Corresponding author: Pijush Kanti Dutta Pramanik (pijushjld@yahoo.co.in) and Kyung-Sup Kwak (e-mail: kskwak@inha.ac.kr)

This work was supported by National Research Foundation of Korea-Grant funded by the Korean Government (Ministry of Science and ICT) under grant NRF-2020R1A2B5B02002478.

ABSTRACT Mobile crowd computing (MCC) that utilizes public-owned (crowd's) smart mobile devices (SMDs) collectively can give adequate computing power without any additional financial and ecological cost. However, the major challenge is to cope with the mobility (or availability) issue of SMDs. User's unpredicted mobility makes the SMDs really unstable resources. Selecting such erratic resources for job schedule would result in frequent job offloading and, in the worst case, job loss, which would affect the overall performance and the quality of service of MCC. In a Local MCC, generally, a set of users are available for a certain period regularly. Based on this information, the chances of a user being available for a certain duration from a given point of time can be predicted. In this paper, we provide an effective model to predict the availability of the users (i.e., their SMDs) in such an MCC environment. We argue that before submitting a job to an SMD, the stability of it is to be assessed for the duration of execution of the job to be assigned. If the predicted availability time is greater than the job size, then only the job should be assigned to the SMD. An accurate prediction will minimize the unnecessary job offloading or job loss due to the early departure of the designated SMD. We propose an advanced convolutional feature extraction mechanism that is applied to LSTM and GRU-based time-series prediction models for predicting SMD availability. To collect user mobility data, we considered a research lab scenario, where real mobility traces were recorded with respect to a Wi-Fi access point. We compared the prediction performances of convolutional LSTM and GRU with the basic LSTM and GRU and ARIMA in terms of MAE, RMSE, R^2, accuracy, and perplexity. In all the measurements, the proposed convolutional LSTM exhibits considerably better prediction performance.

INDEX TERMS Mobile grid, Mobile computing, Resource selection, Availability prediction, Deep learning, Convolutional feature extraction, CNN, RNN, LSTM, GRU, ARIMA

I. INTRODUCTION

A. MOBILE CROWD COMPUTING

Modern computationally powerful smart mobile devices (SMDs) are being considered equivalent to desktops and laptops [1]. This prospect has been alleviated not only due to significant improvement in SMD hardware but also the advancement of battery and quick charging technologies for SMDs, along with sophisticated energy management techniques that have emancipated the users from distressing about hasty energy hemorrhage off their SMDs [2]. Therefore, like desktop grid computing, a collection of SMDs can provide a satisfactory high-performance computing (HPC) facility comparable to the traditional HPC systems like supercomputers, grid computing, and cloud computing [3]. Instead of investing in their own infrastructure, organizations can utilize the SMDs available at their premises, which not only be economical but also environment-friendly [4] [5]. Since, in this computing paradigm, public's (or crowd's) devices are utilized, which can be deemed as a crowd of SMDs (crowdworkers), the system is called mobile crowd computing (MCC) [6]. MCC can be utilized not only to cater to the regular computing needs but also as an edge computing infrastructure for processing and analyzing the organizational IoT data in real-time. This would save the time and cost involved in cloud computing.

B. RESOURCE SELECTION IN MCC

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

As the SMDs are heterogeneous in terms of their resources and their owner's behavior and usage patterns, all of them bid varied computing capability and for different time periods and durations. Therefore, to make MCC successful and effective, it is crucial to select the most appropriate SMD for a computing job. While choosing an SMD as a crowdworker, three main criteria are considered:

- Hardware-wise, overall computing capability of the SMD: Whether the SMD has enough hardware specification to carry out the task.
- Usability of the computing resource of an SMD: Even if the SMD has sufficient resources to carry out the task, most of the time, it may be engaged in executing the owner's applications. Therefore, the crowd task would not get enough chances to get executed in a work-stealing fashion.
- Availability of the SMD: On satisfying the above two conditions, a task may be assigned to the SMDs. But most of them go off the network before completion of the job.

In this paper, we addressed the third criterion. Though the first two are the most important objectives, they are out of the scope of this paper. We assume that these two criteria are already met.

C. THE RESOURCE AVAILABILITY PROBLEM

In other HPC systems, the resources are more or less fixed. But since MCC is a dynamic environment, the resources cannot be considered dedicated and accessible anytime and anywhere. Due to the user's mobility, the resources (crowdworkers) in a particular network may not be available continuously over time. However, they might be available for several discrete periods. Due to this instability, there is always a high probability that a crowdworker leaves the network without finishing the assigned job. There are two possible solutions when a crowdworker departs before completing the assigned task:

a. The job is restarted from the beginning by another crowdworker. This delays the task execution as the whole process is to be started again, including resource (crowdworker) selection and job assignment.
b. Savepoints are maintained periodically. When a crowdworker departs, the task is rollbacked to the last savepoint and resumed from there by another crowdworker. This approach might be better than starting the job from the beginning, but keeping savepoints is overhead, and also determining the period length after which the savepoints are noted is a decision challenge.

In both cases, the quality of service of MCC is compromised, which ultimately affects the successful realization of MCC. That is why assessing the availability of the assignee crowdworker before assigning a task to it is so crucial in MCC.

D. SOLUTION APPROACHES AND THE PROPOSED SOLUTION

Before submitting a task to a crowdworker, it is to be assured that it would not leave until the job is finished. But, for mobile devices, guaranteeing availability is not straightforward. One approach, as assumed in [7], is that every crowdworker should announce their departure time immediately after entering the MCC network. Based on the declared departure time, suitable jobs would be assigned so that they could be finished timely. But it has two issues i) not very practical and ii) there is no guarantee that the crowdworker will keep its word that it would not leave before the declared time. Due to several reasons, a crowdworker may leave the network unscheduled even if it is priorly agreed to be there for a specified period.

Another possible option, as suggested in [8], is that if a crowdworker wants to leave the network, it will notify the coordinator. This would solve the first issue discussed in Section 1.3, but not the second completely. The overhead of job handover remains. Also, some dishonest devices may not follow the rule and leave abruptly without notifying.

Owing to these drawbacks, we propose an availability-aware smartphone selection scheme for a local MCC. We predict the availability of an SMD for a minimum duration of the task length (execution period), based on which the resource selection decision can be made. For this, we tracked the in-time and out-time of the SMDs on the previous occasions when they were connected to the considered Wi-Fi access point. Based on this historical mobility/availability information, the probable availability till a particular duration of an SMD at any given point of time is assessed. This problem can be represented as time-series analysis.

Time-series analysis is exercised on the set of observations where each record is observed at a specific time. Analysis of these types of data is not like other statistical modeling and inference due to its apparent correlation in adjoining records introduced by the sampling time. These features limit the applicability of many statistical models that assume the observations are independent and identically distributed. The Autoregressive Integrated Moving Average (ARMIA) model has been a widely used linear model for forecasting time-series data, and it has been a standard for a long time. ARIMA considers lag value determined by the correlation among the continuous values, the dependency between an observation and the residual, and seasonality (if it exists) for building the model and result in close prediction of the future [9]. ARIMA models have the advantages of being more flexible compared to other statistical models and have a better performance for a longer sequence of data with a stable correlation between past observations. But ARIMA model can only capture the linear pattern of the data, but not the hidden patterns that are stochastic and non-linear in nature [10] [11]. Furthermore, an ARIMA model assumes a constant standard deviation in errors, which may not be true in practice. Like most of the real data, the dataset considered in this work is also non-linear in nature.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

Another statistical model, the Markov chain process, has been popularly used for time-series prediction, especially where clarifying the interrelationships of the model is important [12] [13]. Though they provide significant accuracy for short-term prediction, they have a probability of presenting miss prediction for long-term sequences, which is necessary for crowdworker selection. For instance, it may happen that a particular user was not available for certain days in the previous months. Due to the lack of memory in the Markov model, it will not be able to hold the long-term historical information; hence it cannot incorporate such irregularities in the model. This leads to inaccurate predictions.

In summary, the classical time-series analysis models suffer from the following limitations:

- They are sensitive towards missing values.
- They require special transformations to convert the data into a linear form.
- Most of them support only univariate data; they do not support multiple independent variables to be taken as inputs.

Furthermore, the traditional prediction models work better when the data follow a statistical distribution. In our problem, we wanted to capture the real and consistent behavior of the user, which required considering a long-term data relationship. In the user mobility data, there is very little chance of finding a perfect fit of a distribution, which is a basic requirement for the traditional prediction tools. To tackle this, a time-series analysis model is needed that can make a prediction on the dataset without fitting a known distribution.

Considering the above-mentioned issues, machine learning techniques are intensely studied for use in time-series forecasting. Machine learning models are good to exercise the non-linear pattern. K-nearest neighbor, decision tree, support vector machine, etc., can be used to model time-series data when the observation consists of a non-linear pattern.

Capturing the data context changes in time-series data is an important criterion for a prediction model. However, traditional machine learning based prediction models cannot capture these changes appropriately. Hence, they are unable to provide satisfactory prediction accuracy in cases where the contexts of the considered data change frequently. In the user mobility data considered in this work, we are required to identify the users' behavior for a longer period. To provide expected prediction results, the prediction models need to capture these changes appropriately.

Deep learning based model like RNNs (recurrent neural networks) are capable of retaining long-term contextual information due to the presence of a specialized memory and use looping constraint that helps to capture the sequential information in the data. Deep learning models also have the inherent capability of searching relations in the dataset without prior knowledge of any distribution.

Generally, RNNs are used for time-series data prediction. But traditional RNNs suffer from vanishing gradient problems when the input length is too high [14]. So, to overcome the issue of the vanishing gradient problem, upgradations of RNNs like LSTM [15] [16] and GRU [17] [18] were developed. Both methods have been popularly used for sequence modeling and time-series predictions [19]. However, there are some issues with GRUs compared to LSTM, such as the GRUs cannot regulate the amount of memory content that is to be forwarded to the next unit, as they do not store cell state. Whereas LSTMs are capable of regulating the amount of information that is to be forwarded to the next unit. We tested our model with both LSTM and GRU to compare the effectiveness of these methods for this particular problem.

However, a prediction model is not sufficient to attain a handsome or the expected prediction accuracy since these models work on the available input features without any feature extraction. For that, it is required to apply a proper feature extraction mechanism to the dataset. The feature extraction is a crucial aspect of designing a prediction model because it captures the most relevant features from the data, which generally improves the model performance.

The existing and known feature extraction methods, in most cases, can extract the exact features required for a particular prediction problem. But they may be incompetent when we do not have a clear idea of the most dominant features. Therefore, we needed to frame a feature extraction methodology that dynamically extracts features for solving the proposed resource availability prediction problem.

Recently, CNNs (convolutional neural networks), the idea of which was first presented by Fukushima in 1980 [20] and later improved by LeCun *et al*. [21] [22], have become popular for extracting dynamic features for prediction-based models. A CNN is a special kind of neural network for processing 2-D image data [23] [24] [25]. CNNs are very effective in extracting and learning features not only from one-dimensional sequential data, such as univariate time-series data, but also from multivariate time-series data [26] [27].

Owing to their distinct architecture, the LSTM layers in the LSTM models can capture the sequence pattern information quite efficiently. As the LSTM networks are designed to deal with temporal correlations, they utilize only the features provided with the training set [28]. The convolutional layers of CNNs can extract more valuable features by filtering out the noise prevailing in the raw input data [29]. They are also capable of scooping the hidden features that otherwise could not be pulled out by using LSTM. This is the core motivation to exercise a convolutional feature extraction layer in addition to the basic LSTM for our presented availability prediction problem, so that we could exploit the benefits of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

both techniques to achieve a better prediction performance. The combination of CNN and LSTM is useful for learning features of not only short-term time variation but also long-term dependency periodicity [30].

Considering this, we proposed a specialized convolutional feature extraction method to enhance the performance of the LSTM and GRU models.

E. CONTRIBUTION OF THIS PAPER

To achieve our goal, we go along the following tasks:

a. We designed a logging model for recording the in-time and out-time of SMDs in consideration with respect to a Wi-Fi access point.
b. We proposed a novel dynamic feature extraction process suitable for the datasets where the features are unknown.
c. We designed a novel method for representing time-series data into a vector to perform the convolutional feature extraction.
d. The proposed convolutional feature extraction is combined with both LSTM and GRU for prediction.
e. The prediction performances are compared with each other as well as with the basic LSTM and GRU, and also with ARIMA.

F. IMPLICATION OF THIS WORK

The presented approach is most useful for a local MCC environment where the SMD owners visit and join the MCC network on a regular basis. In such a scenario, the main objective of this paper is to improve the QoS and reliability of the MCC by minimizing the handoff or job offloading and reassignment. The success of this approach will depend on the accuracy of the availability prediction of the considered SMDs.

G. ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows. Section 2 discusses the related research works. The MCC system model and the hypotheses considered for this work are presented in Section 3. A crowdworker selection method based on availability is also presented in this section. Section 4 covers the data collection, simulation, and result analysis for the availability prediction experiment. The paper is concluded in Section 5.

II. RELATED WORK

In this section, we report the closely related work to that of presented in this paper. The papers are grouped into similar problems and similar solution methods.

A. RESOURCE AVAILABILITY PREDICTION IN MOBILE GRID/CLOUD COMPUTING

Considering the hardware improvements of modern mobile devices, many have suggested utilizing their computing capabilities in different forms [31] [32] [33]. Lately, computation on mobile devices has attracted researchers' attention, especially to form mobile ad hoc cloud computing [34] [35] [36] [37]. But despite increasing research interest on mobile cloud computing, we could not find much work on predicting the stability of mobile devices in this particular scenario. Following are some initiatives that closely or remotely match that of our proposed work.

Brevik *et al*. [38] aimed at enabling the grid job scheduler to make on-the-fly decisions by providing live availability predictions. To predict the availability duration of a resource, they used the Weibull method (parametric model fitting technique) along with Resample and Binomial methods (non-parametric techniques). The authors attempted to estimate a specific quantile for the availability distribution and the confidence for each estimation. Andrzejak *et al*. [39] attempted to predict the availability of the grid resources within a time interval [T, T+p], where p is the prediction interval length with values [1,2]. For this, they used the Naive Bayes and Decision trees based predictive models. They also aimed to identify the resource predictability indicators and the factors that incite prediction error. But these works do not cover the availability prediction of mobile devices in a non-dedicated mobile grid environment.

Vaithiya and Bhanu [40] proposed a task scheduling algorithm for the mobile grid, predicting the dynamic availability of mobile resources. Selvi *et al*. [41] dealt with the issue of node mobility in an ad-hoc mobile grid by profiling the regular movements of a user over time. But none of these considers the historical characteristics of the devices, which may hinder in achieving the optimal effects in SMD selection.

In FemtoClouds, a mobile device cloud control system, presented by Habak *et al*. [8], the presence time prediction of mobile devices is incorporated. In this work, it is assumed that the controller has knowledge of the exact departure time of each device for each session. But, in practice, some dishonest users may depart before the declared departure time, while some may be forced to cut off from the network due to some genuine reasons such as battery used up. To counter this problem, Zhou *et al*. [7] suggested considering the historical characteristics of the devices to evaluate the record of honoring their departure promise. They proposed a mobile device selection method, considering the status and stability of the devices. However, both of these works assume that the SMDs declare the departure time voluntarily, which may not be practical.

Sipos and Ekler [42] estimated the availability of mobile devices in a P2P storage system. They predicted the actual availability based on the nodes' self-declared availability or unavailability for the subsequent considered time period. Different classifiers were used to check the accuracy of the prediction model in the simulated mobility scenario.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

Pramanik *et al*. [43], based on the device mobility patterns in a P2P mobile cloud, estimated the relative stability of a group of SMD users over a period with respect to each other.

Haryanti and Sari [44] predicted the mobility of a group of resource-providing nodes with respect to a resource-seeking node. The purpose was very much similar to ours (presented in this paper) that the task from the requesting node should be given only to those resource-providing nodes which are supposed to be in contact with the requesting node until the task is completed. Farooq and Khalil [45] also proposed a method to predict a time duration for which a resource requesting node would remain within reach of the resource requesting node in a mobile grid. Based on the predicted time, the task assignment decision is taken. The prediction is based on the previous record of the time duration of their contact, whereas the contact is calculated by the distance between them based on their locations, assessed by their GPS coordinates.

Nevertheless, in spite best of our effort, we could not find any significant work that endeavors to predict the periodical availability of the public-owned SMDs in a non-dedicated and dynamic MCC environment.

B. DEEP LEARNING FOR RESOURCE MANAGEMENT AND PREDICTION

Considering the potential, deep learning has been applied in various domains and applications for different purposes [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56]. Specifically, in time-series forecasting, LSTM [57] [58] [59] and GRU [60] [61] [62] are widely used.

Many researchers exploited the convolutional aspect of CNN in combination with LSTM to improve the performance of time-series prediction/forecasting in various applications, such as for inventory prediction [30], stock price prediction [63] [64] [65] [66], gold price forecasting [28], Bitcoin price forecasting [67], tourist flow forecasting [68], sentiment prediction of social media users [69], household power consumption prediction [70] [27], photovoltaic power prediction [71], wind power forecasting [72], $PM_{2.5}$ prediction [73] [74], predicting NO_x emission in processing of heavy oil [75], forecasting natural gas price and movement [29], urban expansion prediction [76], predicting waterworks operations at a water purification plant [77], predicting sea surface temperature [78], typhoon formation forecasting [79], crop yield prediction [80], COVID-19 detection and predictions [81] [82] [83], human age estimation [84], and so on.

Deep learning based techniques are being used for efficient resource management and prediction in cloud [85] [86] [87] [88] [89] [90], edge computing [91] [92] [93] and other wireless distributed systems [94] [95] [96] [97] [98].

The inherent capability of capturing short-term as well as long-term instances has led LSTM [99] [100] [101], CNN [102], and convolutional LSTM [103] [104] to be popularly used in mobility predictions. In his master's thesis [105], Pamuluri compared different deep learning methods, including LSTM, CNN-LSTM, GRU, to predict users' mobility with respect to a mobile base station. Cui *et al*. [106] used LSTM to predict the availability of mobile edge computing-enabled base stations depending on the vehicle's mobility for offloading the computation jobs from the vehicle to the base station. Li *et al*. [107] used LSTM to track user mobility for efficient dynamic resource allocation across different network slices in a 5G network.

In our recent work [108], addressing the same problem as in this paper, we applied ConvLSTM on the user mobility dataset. The ConvLSTM module is a predefined implementation provided by Keras Python API [109]. The model exhibited an average accuracy of 78.43%. Further, a sampling phase was introduced to eliminate the overfitting and underfitting problems. However, here, we just used the readily available ConvLSTM model that is specially meant for 2D spatial data and requires advanced transformations to work with time-series data. The limitation of the model is that it does not have the required flexibility. The only available tuning option for the model was the selection of the hyper-parameters and the transformation technique. This opens up the scope for exploring other options which are more generalized and flexible, which would lead to better model performance.

III. SYSTEM MODEL AND HYPOTHESIS

A typical MCC system comprises three major entities a) coordinator, b) computing resources (crowdworkers), and c) the communication media between the two. The coordinator is responsible for managing the complete system by performing task farming, execution time estimation, crowdworker discovery, assessing crowdworker's resource usefulness, availability prediction, resource reallocation, result collection, etc. The crowdworkers execute the assigned tasks and return the results to the coordinator. SMDs and the coordinator communicate with each other via a Wi-Fi access point. A typical MCC setup is shown in Fig. 1.

An SMD user needs to install the MCC client application on his SMD if he is willing to share his device's resources, i.e., agrees to be a crowdworker in an MCC setup. The MCC coordinator would automatically be informed whenever a crowdworker enters its network (the access point to which the coordinator is connected). In some cases, certain SMDs may join MCC several times a day. The coordinator keeps the record of each SMDs that joins the MCC.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

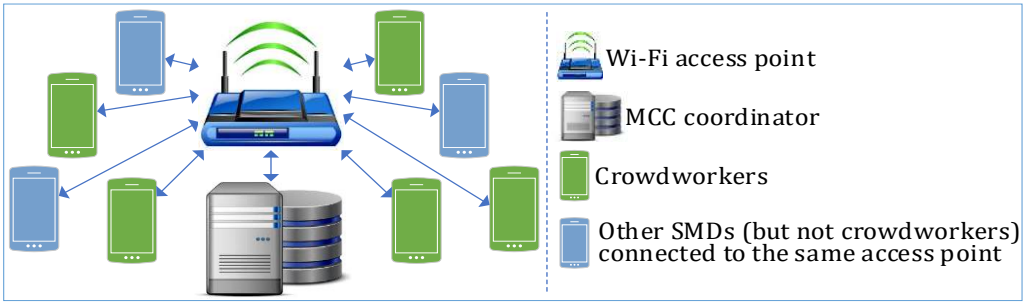


Fig. 1. A typical MCC setup

In this paper, we have considered an indoor environment (campus) where most users (with their SMDs) are often available for a certain duration. For example, in a classroom and a workplace, the students and the workers are regularly available for a specific duration in regular intervals. If they take public transport for commuting to reach their institute and workplaces, in most probability, they would be available for the duration from boarding point to the destination. Similarly, some people spend a certain amount of time in the library regularly while some go to the same coffee shop or canteen regularly. In all these cases, the availability of the users can be predicted by analyzing their presence history.

The accuracy of the availability prediction depends on the campus type. For example, in a classroom or a typical office, the availability is somewhat predetermined. Whereas the predictability in a coffee shop (where customers come regularly) varies as per its location and the services it offers. Likewise, in public transport (regularly used by a group of commuters), the availability is very much fixed (usually one drops at his stop regularly). The crowdworker predictability gradient based on the availability is shown in Fig. 2.

Fig. 2. Predictability gradient of crowdworker's availability in a local MCC

To model the working of a local MCC, we have assumed the following:

- We consider a general task execution model where the SMDs receive some compute-intensive tasks either individually or in batches.
- Each task has its own computation requirements, input and output data size, and finite execution time. We assume that these parameters are known.
- Each crowdworker avails a fixed and equal bandwidth.
- Each crowdworker completes the assigned subtask within a finite time and sends back the results before leaving the MCC network.
- A crowdworker would share its resources until it is present in the network.
- All SMDs, which have the MCC client installed, are considered as crowdworker and willing to share their resources, either on a profit or non-profit basis.
- The SMDs in MCC are uniquely identified by the UIDs.

To model our proposed availability prediction, we further assume the followings:

- In need of job submission, the coordinator looks for the most appropriate crowdworker(s).
- The MCC coordinator already has a list of suitable crowdworkers (based on the criteria mentioned in Section I.B).
- The coordinator decides to pick the top-ranked crowdworker(s) from the list.
- Just before submitting the job to the selected crowdworker, the coordinator wants to be sure of the probability of the crowdworker being available until the job is finished.

This paper addresses the last point, i.e., before the job is actually be submitted, the stability of the selected crowdworker is to be assessed for the duration of execution of the assigned job. If the crowdworker's presence time is greater than the task size (estimated execution time), then only it is finally considered for the job assignment. The workflow diagram of the whole process is depicted in Fig. 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

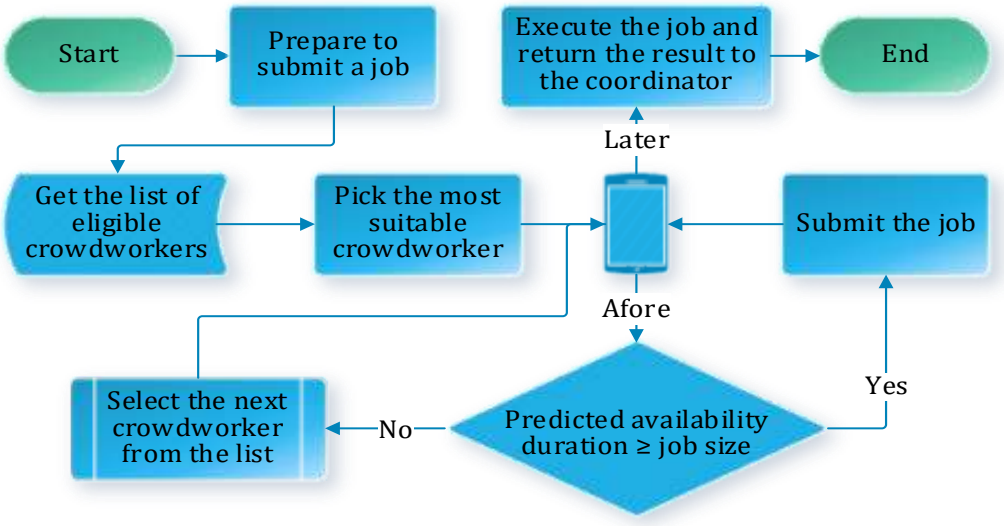


Fig. 3. Workflow diagram of crowdworker selection based on availability

IV. CROWDWORKER AVAILABILITY PREDICTION

A. PROBLEM DEFINITION

Let T_i be the job size (execution time) of a job J_i and M_i be the preferred crowdworker for J_i. At the time of job submission (t_i), we need to know how long M_i might be available after t_i; let this be M_a. J_i should be submitted to M_i if and only if Eq. 1 is satisfied.

$$M_a \geq T_i + k \quad (1)$$

Where, k is some constant.

The details of the M_a calculation and crowdworker selection criteria are discussed in the next subsection.

B. AVAILABILITY PREDICTION OF AN SMD

The major components/modules of the crowdworker selection procedure are as follows:

Calculate completion time: The job completion time is an approximate higher bound value of the time required for the particular job to complete. This function needs two parameters, namely, job size (T_i) and time of job submission (t_i). The completion time (T_i^c) of the job J_i is defined by Eq. 2. In this paper, we assume that for each J_i, T_i is the same for all crowdworkers.

$$T_i^c = T_i + t_i \quad (2)$$

Get selected device: As depicted in Fig. 3, the top-ranked SMD in the crowdworker's list for the reckoned job would be considered.

Get session history: According to the UID from the 'get selected device' module, the history of the SMD is extracted from the log. The details of data collection are discussed in Section I.A.

Predict out-time: This module takes the session history of the device's previous session durations to predict the expected session duration in the current time using CLSTM (convolutional LSTM). The current session in-time (S_i) is added with the forecasted duration (P_i) to get the predicted out-time (S_o) of the device in the current session, as shown in Eq. 3. The predicted availability duration (M_a) is calculated by Eq. 4.

$$S_o = S_i + P_i \quad (3)$$

$$M_a = t_i + S_o \quad (4)$$

So, Eq. 1 can be rewritten as Eq. 5, where $k_1$is the runtime of the prediction algorithm and k_2 is the padding time between decision making and job dispatching.

$$M_a \geq T_i^c + k_1 + k_2 \quad (5)$$

Crowdworker selection: This function checks for the availability of the SMD for the specified duration and returns a Boolean for selection. The SMD will be selected if Eq. 5 is satisfied.

Fig. 4 depicts the combined workflow of the above-mentioned modules, whereas Fig. 5 shows the important steps followed towards SMD selection.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

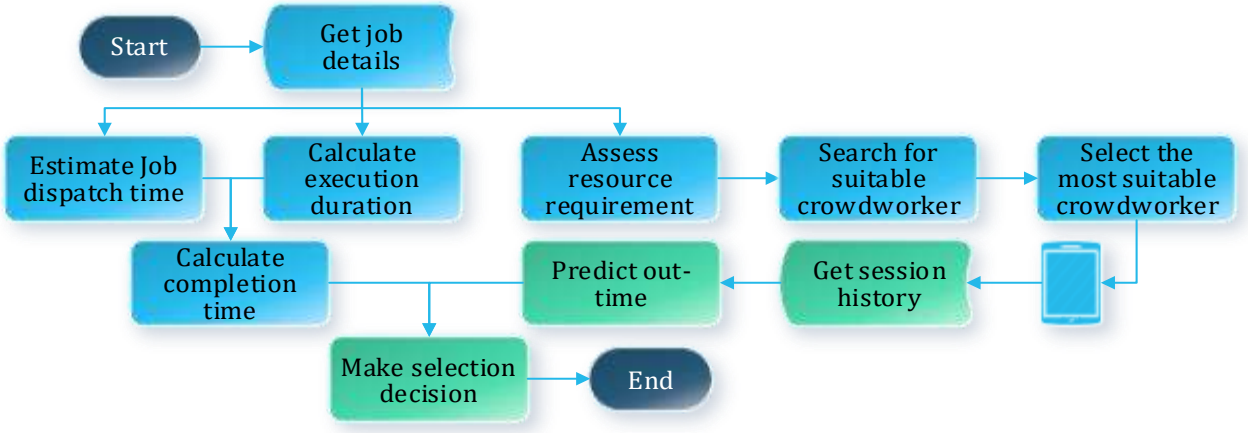


Fig. 4. Availability prediction process of an SMD in MCC

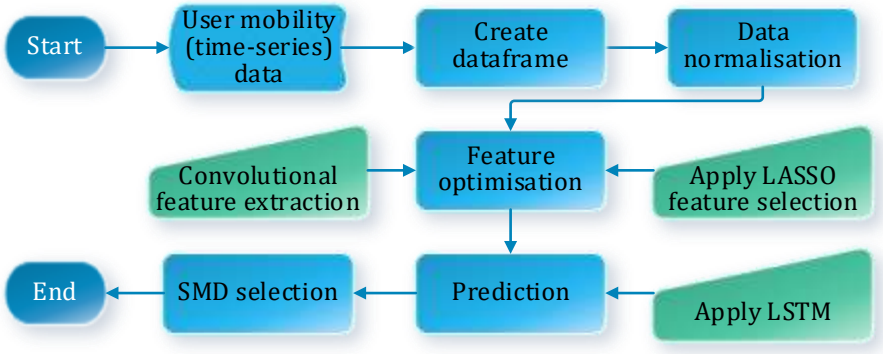


Fig. 5. Important steps for SMD selection

C. DATA COLLECTION

For the experimental purpose, we considered a computer laboratory scenario in an educational institute. To generate the experimental data (i.e., SMD's presence time and duration), the digital simulation could have been opted, but it might not have the uncertainties that are associated with the user presence pattern. Without the uncertainties, the simulation may not behave like the real case scenario. Introducing uncertainties artificially in the simulation may not be feasible as it might break the co-similarities among the data points.

Therefore, we counted on user data traces from a real network with respect to a particular Wi-Fi access point covering a mid-size hall. We collected user data from the Wi-Fi access point deployed at the Data Engineering Lab of the Department of Computer Science & Engineering at National Institute of Technology, Durgapur. The lab is generally accessed by the institute's research scholars, the project students, faculty members, and the technical staff.

For every entry, the duration for the SMD that remains in that particular Wi-Fi network was logged. This has enabled us to collect more data efficiently as well as validating the prediction algorithm using real availability data. The overall view of the network and programs for data collection is shown in Fig. 6. The database schema is shown in Fig. 7.

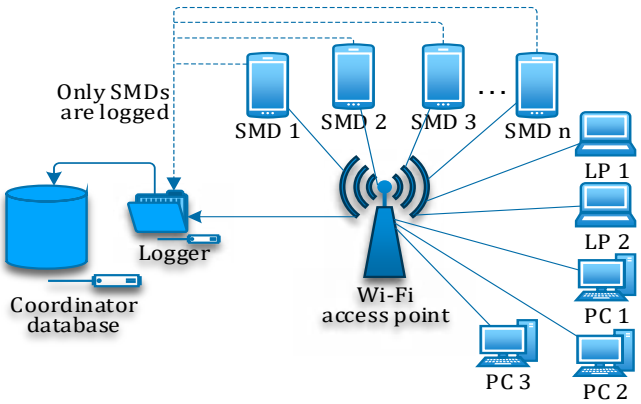


Fig. 6. Data collection and log maintenance of SMDs

For gathering the data, a logger program was developed using Python 3.6 environment. The program script was developed as a service, which constantly looked in the WLAN for the devices connected to the Wi-Fi access point. The complete procedure is as follows:

1. The Wi-Fi access point at Data Engineering Lab was selected.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

2. The Python script constantly monitored the wireless network interfaces. All the devices connected to the access point were identified (UID) using their MAC addresses.
3. Logging was skipped for other connected devices than SMDs (such as PCs and laptops) by filtering the MAC addresses.
4. The SMDs were logged by the program, identified by their MAC address and the session in-time and out-time. These three parameters were logged and stored in a local MySQL database for further analysis.

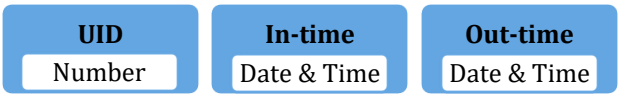


Fig. 7. The database schema for SMD availability logging

D. DATA SELECTION

We collected user data for about eight months. From the complete dataset, we selected data of 150 days, considering the data quality after applying the normal distribution over the dataset. Out of total days, the maximum concentration of the connected devices was on these 150 days (T_d). However, we also wanted to check the performance of the prediction model when the collected data are less. For this, we had a dataset of 120 days which is a subset of the 150 days data. The model was applied on both these datasets. Opting for two datasets with a difference of a month of data would allow us to evaluate the applicability of the model in different crowdsourcing applications. Further, out of all recorded users, we considered 50 users for whom there was high presence frequency and less sparsity.

E. DATA PREPARATION

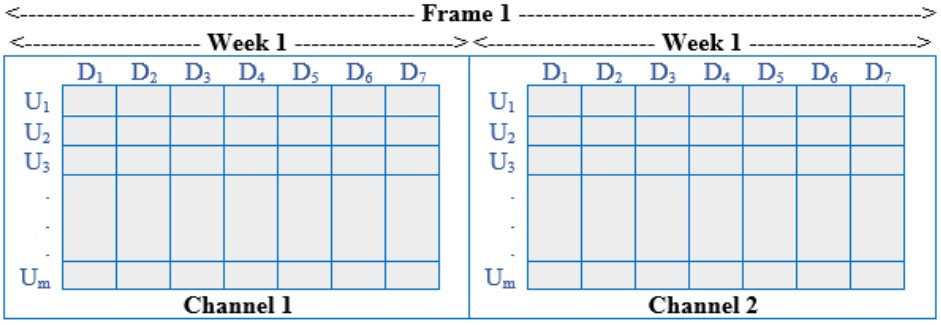
Since we used the time-series data as the raw data, they need to be converted into a suitable form so that the convolutional filters could be applied to extract the features. The steps that we performed before sending the data to the LSTM model are elaborated in the following subsections.

1. DATA FRAME CREATION

To represent the users' mobility, we followed the following steps for creating the required data frames, as shown in Fig. 8.

- The data frames, each representing one week's data of users' mobility, were created. Each frame has two channels - channel 1 and channel 2, representing the in- and out-time records of the users, respectively.
- The data frames have UxD dimensions, where U (number of users) = 50 and D (number of days) = 7.
- The total number of frames for in-time and out-time is calculated by $2*T_d/D$.
- Each cell in channel 1 contains the in-time of the user on a particular day. Similarly, the out-time is recorded in channel 2.
- A user might have multiple entries on a single day. In that case, we normalized the entries by keeping only the entry for the longest duration (for example, if U1 entered four times on D1 and the durations are of 4, 23, 37, and 57 minutes, only the entry for 57 minutes is considered). We adopted this approach to implement a fair share policy so that one SMD would be given a job in only one session. Furthermore, a deep learning model works better with data that is consistent and has less deviation. In the case of MCC, the data may contain session information for multiple short and inconsistent sessions. To avoid fitting inconsistent data, we selected the longest continuous session duration to be the final session if the gap between the sessions is smaller than a particular threshold time value. Algorithm 1 presents the procedure of calculating the longest continuous session duration. Here, S_n denotes the n^{th} session, IN and OUT represent the in- and out-time for the respective session, and λ is the threshold criteria for merging two sessions. In our experiment, we considered $\lambda = 0.05$, i.e., 5% of the entire duration.

Algorithm 1: Selecting longest continuous session duration
Input: Raw session data
Output: Updated session data
while (S_{n+1}) if $(S_{n+1}^{IN} - S_n^{OUT}) < \{(S_{n+1}^{OUT} - S_{n+1}^{IN}) + (S_n^{OUT} - S_n^{IN})\} \times \lambda$ then concatenate (S_{n+1}, S_n) end while



This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

Fig. 8. A sample frame for in- and out-time

2. DATA NORMALIZATION

Each of the cells in the channels contains time values that are not appropriate for direct input to the prediction model. For this reason, we represented the collected user mobility data (time-series data) as image data. Since CNN works on image data that have the channel intensity values ranging between 0-255, we normalized the time values for both the channels between 0 and 255. The pixel-wise normalization of the time-series data (x) into image intensity (y) is achieved by applying a linear equation, as shown in Eq. 6.

$$y = 10.625x \tag{6}$$

A sample of data normalization based on input data is shown in Fig. 9. The darker to lighter shade indicates the increasing hour of a day, and the black color indicates the unavailability of the particular user on that particular day. For example, the black color denotes that U1 is absent on Saturday and Sunday.

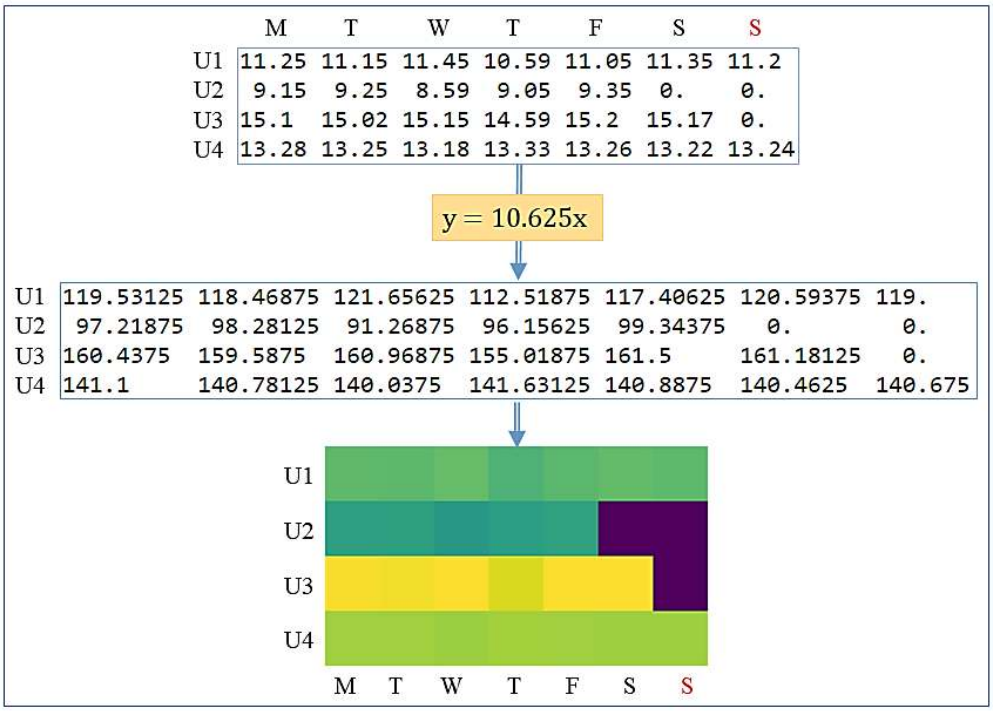


Fig. 9. A sample of data normalization based on input data

F. FEATURE OPTIMIZATION

To increase the model accuracy and training and inference speed, we needed to optimize the feature sets. Feature optimization includes extracting new features from the input data and removing the unwanted features. Feature optimization improves the model's performance and makes it more interpretable. The followed steps for feature optimization are discussed in this section.

1. FEATURE EXTRACTION

Typically, in the time-series datasets, the features (e.g., length of time-series, period, mean value, standard deviation value, etc.) are not sufficient for prediction modeling and cannot be used straightforwardly. The format of the existing data features may not be suitable for direct analysis and comparison.

The new features are generated by reformatting, combining, and transforming the original features. This makes the data suitable for modeling and increases the model's training and prediction accuracy.

a) Issues with Popular Feature Extraction Methods

There are various methods for feature extraction, which are used depending on the type of the data and problem. PCA (principal component analysis), GLCM (grey level cooccurrence matrix), etc., are the popular feature optimization methods for dimension reduction in image data. These methods have been proven to work well in time-series predictions and image classifications. However, these methods are not suitable for the problem addressed in this paper because the resource availability prediction in MCC is a generalized time-series prediction problem without any prior knowledge of the important features.

Further, PCA requires some hyperparameter tuning to generate quality features, which is not trivial. GLCM can extract only certain known features and is highly dependent on the characteristics of the data. However, our problem

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

demands a dynamic and generalized feature extraction methodology that is not affected by the data size and quality.

b) Need for Convolutional Feature Extraction

In our dataset, except the in-time and out-time of the users, no other information is available. This means there are not sufficient features to model the user availability pattern. To elaborate further, let us consider the in-time and out-time of three randomly chosen users over a period of 30 days, as shown in Fig. 10. It can be observed that there is a high variance in the in- and out-time patterns for all users. It also varies day-wise for each individual user. It implies that even if a user's availability seems to follow a pattern, it might not hold true throughout the considered period. This inconsistency could be either intentional or driven by several factors that are not apparently visible from the raw dataset.

However, these nonobvious features might provide some valuable information. But it is impossible to unearth these features manually. For this, we need some automated and dynamic feature extraction mechanism, which would extract the useful features from the dataset.

We found the convolutional feature extraction method as a suitable option for our problem. In many of the dynamic feature extraction problems, CNN has popularly been used. CNN is a supervised classification model comprised of two major segments: a) a convolutional feature extractor and b) a SoftMax classifier. In a traditional CNN, feature optimization (extraction and selection) is automatic. But when using only the convolutional feature extractor, we need a separate feature selection model. A convolutional feature extractor is known for its capability to generate dynamic and new features. Therefore, we transformed our time-series data so that convolutional feature extraction can be applied.

c) The Convolutional Feature Extraction Process

In this section, we present the details of the convolutional feature extraction method designed specifically for the problem presented in this paper. The input to the considered convolutional feature extraction model is shown in Fig. 11. We considered the stride or the window size as 1, i.e., the data frame window slides for each day, as shown in Fig. 12.

A frame represents the number of values considered in a single instance of the model. The architecture for convolutional feature extraction is shown in Fig. 13.

To train the model, we needed to feed the data into it. We did not have a much larger dataset, which was generated by acquiring mobility data for only a few months. Hence, we fed the training data into the model serially, adding one day at the end of the window and removing one day from the front in each iteration.

The distribution of the frames to train the feature extractor model is shown in Fig. 12. In the figure, x_{i_m} represents the input frame for a particular user m, while i is the set of days available as prediction input. Here, we considered the value of i as 50 (however, the value of i would vary according to the total number of available samples). Since we considered the mobility data of 50 users, the maximum value of m in this problem would be 50, and m would iterate for all the users, i.e., 50 times. Here $d_n$represents the data instance for a particular day n. Since we considered two datasets consisting of user mobility data for 120 and 150 days, the maximum value of n would be either 120 or 150.

After creating the input frames, we proceeded to the feature extraction phase. A novel feature extraction model is developed specifically for this work, as presented in Fig. 14. This CNN model contains the following five segments:

- **Weekly mobility data:** Each week of mobility data for all the users in the considered duration is represented by one data frame. These frames are the input to the feature extraction model.
- **Channels for in-time and out-time:** Each frame is split into two channels; one for in-time and the other for out-time for all the users. The subsequent functions were repeated for each channel separately.
- **Frame-by-frame training:** For training the convolutional feature extractor, the frames in a group of 50 were arranged in a single block. In the next timestep, a single stride of each frame was made for further predictions till all the frames were considered, as shown in Fig. 12.
- **Model:** This is the CNN model for convolutional feature extractor without the classifier, as shown in Fig. 11. The model is architected using three blocks of varied convolutional and max-pooling layers, as shown in Fig. 13. We considered a filter of dimension 3x3. Block 1 comprises a single convolutional layer with a dimension of 50x50 and 16 filters and a max-pooling layer of dimension 25x25. Block 2 comprises two convolutional layers with a dimension of 23x23 and 21x21, along with 32 and 64 filters, respectively. It also has a max-pooling layer of dimension 10x10. In block 3, there are two convolutional layers, 8x8 and 6x6, along with 32 and 16 filters, respectively.
- **Feature extraction:** The extracted features from each input data frame were stored in a vector form, which was fed into the LSTM prediction model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

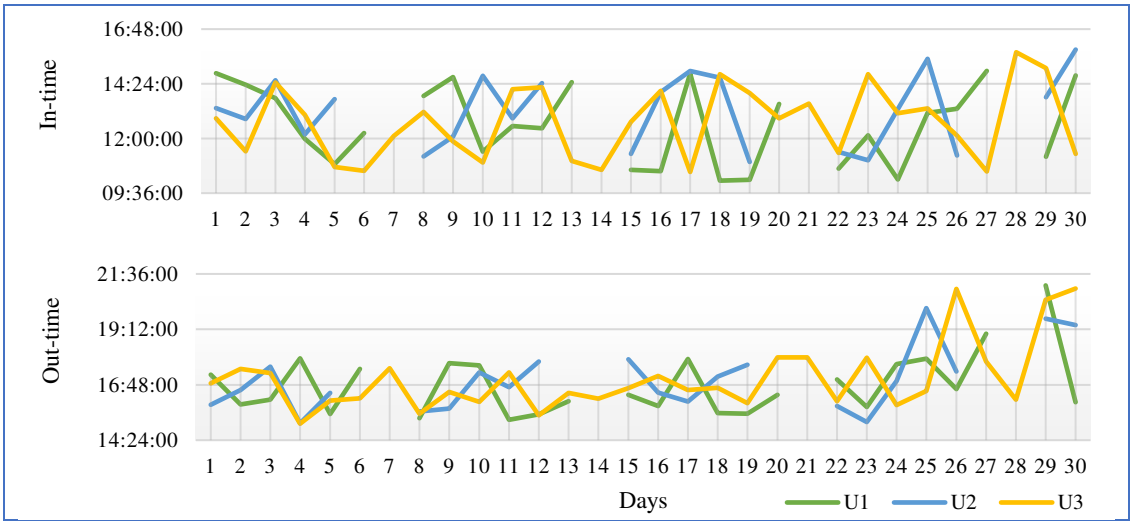


Fig. 10. The in- and out-times of three sample users over a period of 30 days

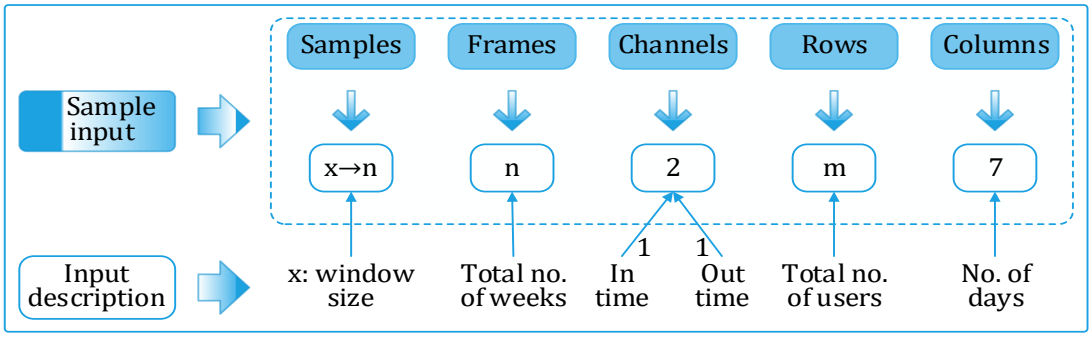


Fig. 11. Input parameters for the considered CNN model

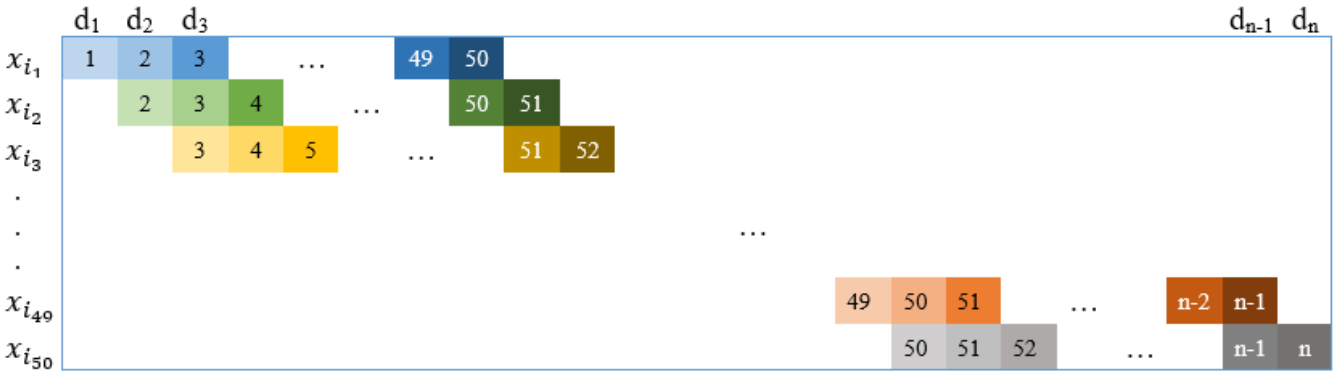


Fig. 12. Distribution of the frames for training the feature extractor model

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

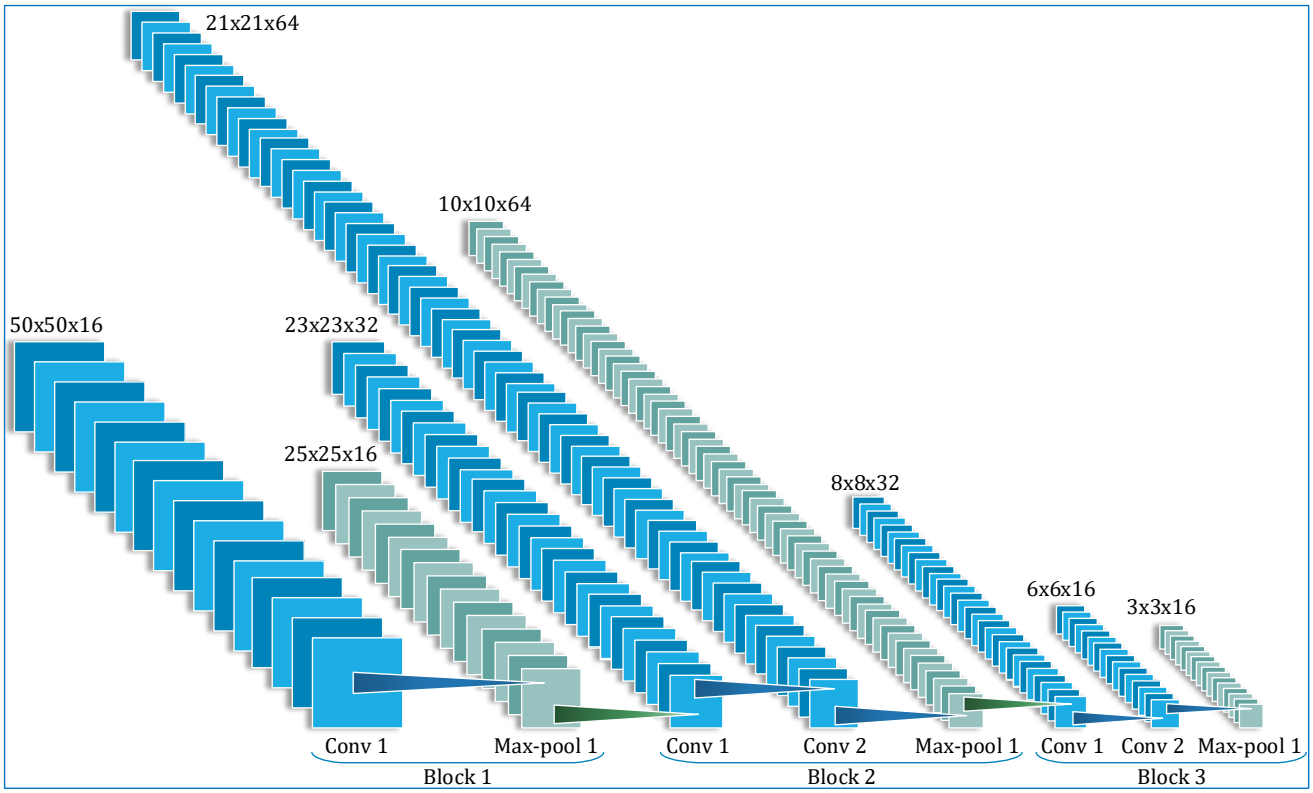


Fig. 13. Convolutional feature extraction architecture

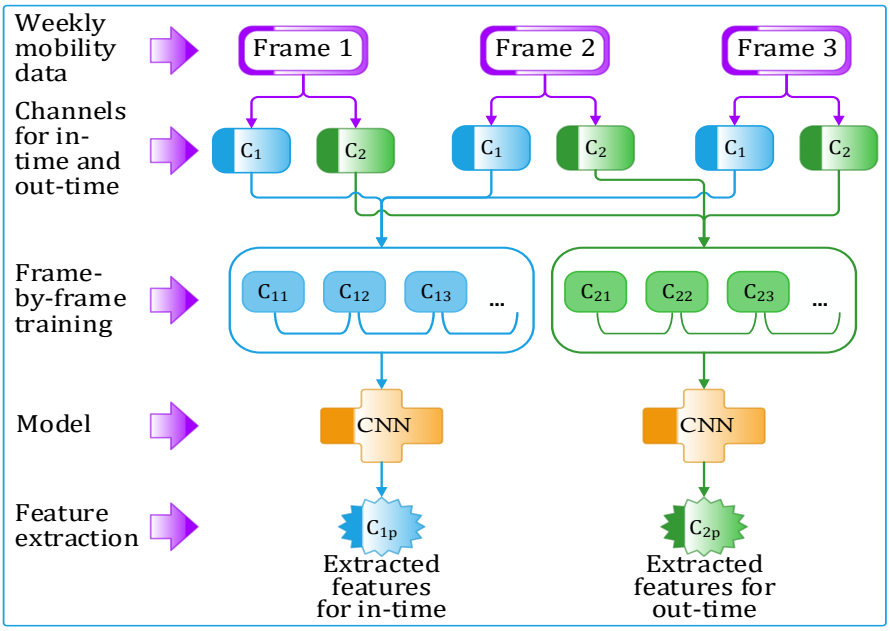


Fig. 14. Feature extraction for in- and out-time using CNN

2. FEATURE SELECTION

Not all the features in the dataset are really useful. Irrelevant and redundant features increase the training time, decrease the accuracy, and make it complex to interpret. That is why, for model construction, it is important to select only those features that are essential and can represent all the features. Feature selection is used for selecting relevant features from the dataset by eliminating the redundant or irrelevant features or the features that are strongly correlated in the data without losing much information. The primary reasons for using feature selection and the popular regression methods are mentioned in Fig. 15. The features that contribute most to the desired prediction or output are generally retained. Even after convolutional feature extraction, the model may

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

contain some features that may cause performance degradation due to multi-collinearity. Feature selection not only removes multi-collinearity and improves the prediction accuracy but also reduces training time, simplifies the model for better interpretation, and improves the chances of generalization, thus, avoiding overfitting.

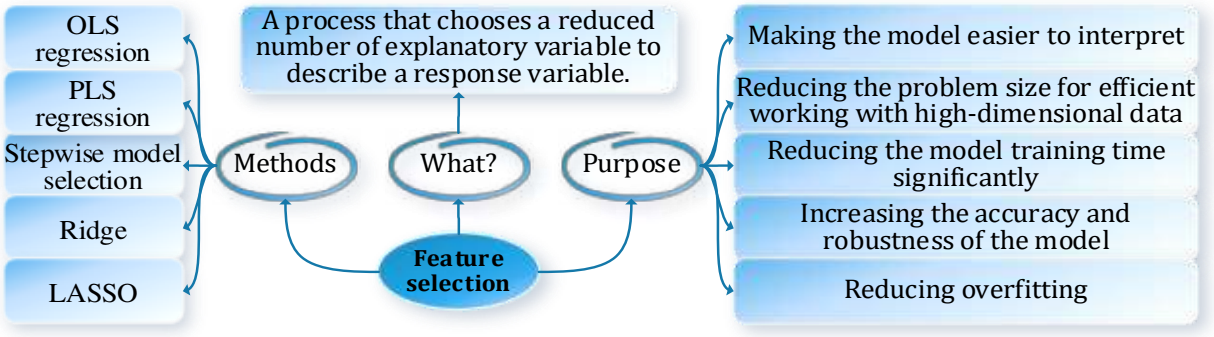


Fig. 15. Purpose of feature selection and the popular regression methods

Though there are a few regression methods for feature selection, as shown in Fig. 15, we avoided the traditional methods such as Ordinary Least Squares (OLS) regression, stepwise model selection, and partial least squares (PLS) regression, etc., due to their sensitiveness to random errors. In the case of multi-collinearity in the input values, Ridge and LASSO methods perform effectively. However, we preferred LASSO (least absolute shrinkage and selection operator) because the major problem with Ridge is that though it shrinks the coefficients nearly to zero but not exactly to zero. Hence the ridge regression fails to provide an unambiguous and easily interpretable sparse model, especially when the number of predictors is large [110]. On the other hand, LASSO offers a better prediction accuracy and model interpretability by eliminating the irrelevant variables/coefficients that are not associated with the response variable. If there is a high correlation in a set of predictors, LASSO picks only one among them while shrinking the others exactly to zero. The leftover non-zero values are selected to be used as features in the model. This method leads to a reduction in variance without increasing the bias much. This is especially beneficial when the dataset consists of a small number of observations and a large number of features. The cost function of LASSO is defined by Eq. 7.

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - (w_0 + \sum_{j=1}^{X} x_{ij} w_j))^2 + \lambda \sum_{j=1}^{X} |w_j| \quad (7)$$

where, m is the total number of training samples or instances in the dataset, X is the total number of features, y_i represents the value of target variable for i[th] training example, x_{ij} is the i[th] observation for j[th] feature, w_0 is the intercept term, w_j represents the weight of the j[th] feature, and λ is the tuning parameter that controls the feature reduction. The larger λ becomes, the more feature coefficients shrink to zero. Also, as λ increases, bias increases, and variance decreases. Here, the goal is to minimize the error function $\sum_{i=1}^{n}(y_i - (w_0 + \sum_{j=1}^{m} x_{ij} w_j))^2$, subject to the regularization term $\lambda \sum_{j=1}^{m} |w_j|$.

After applying the convolutional feature extraction, we had a total of 46,384 features in our considered user mobility dataset. After using LASSO on this feature set, the total number of features was reduced to 4,976.

G. PREDICTION METHOD

1. BASIC LSTM ARCHITECTURE

An LSTM cell is a special variant of an RNN cell that can handle information of a more extended sequence of information, which can help make the prediction more accurate for longer sequences. Each cell is capable of barring information from flowing through or allowing it to flow through without any change. Allowing the information without change enables LSTM to remember the information from the previous timesteps. Through the LSTM cell sequence chain, there are several inputs and outputs which allow adding or removing information to the cell state. Adding or eliminating information to a cell is done through gates. The gates are the neural networks used to regulate the information flow through the sequence chain of LSTM cells. These gates or the sigmoid layers turn all output values in a value between 0 and 1, where 0 indicates nothing of the component should pass through, and 1 is for the opposite, i.e., everything would be through. The three gates that control the cell states of an LSTM are briefed below, and a typical LSTM block is shown in Fig. 16.

Forget gate: This gate gets rid of the information we want to remove from the cell state. The forget gate (f_t) is defined by Eq. 8.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (8)$$

where, σ is the sigmoid gate activation function, W_f and U_f are the weight matrices for mapping the current input layer and previous output layer into the forget gate, h_{t-1} is the output from the previous cell, x_t is the input layer, and b_f is the bias vector for the forget gate calculation.

Input gate: The input gate (i_t) controls how much information from the current input layer (x_t) pass to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

current input cell state ($\check{c}_t$). This gate, defined by Eq. 9, gives the outputs between 0 and 1 and decides which values to update. The candidate values which are to be used to update the cell state are calculated by a tanh layer, as shown in Eq. 10. The input gate combined with the current cell state updates the current output cell state (c_t), as defined by Eq. 11.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (9)$$

$$\check{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (10)$$

$$c_t = f_t \times c_{t-1} + i_t \times \check{c}_t \quad (11)$$

where, W_i and U_i are the weight matrices for mapping the current input layer and previous output layer into the input gate, W_c and U_c are the weight matrices for mapping the current input layer and previous output layer into the current input cell state, b_i and b_c are the bias vectors for the input gate and input cell state calculation, and *tanh*, a hyperbolic tangent function, is the activation function for current input cell state.

Output gate: The output gate (o_t) controls the amount of information passed from the current cell state to the current output cell state. To get the filtered output, the current cell state is passed through a sigmoid layer, as shown in Eq. 12, which decides what parts of the cell state would be considered as output. The final output (h_t) is derived, as shown in Eq. 13, by multiplying the output of the sigmoid gate, with the output cell state that is passed through a tanh layer (for squeezing the values between -1 and 1).

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (12)$$

$$h_t = o_t \times \tanh(c_t) \quad (13)$$

where, W_o is the weight matrix for mapping the current input layer into the output gate, b_o is the bias vector for the output gate calculation, and c_{t-1} is the previous output cell state.

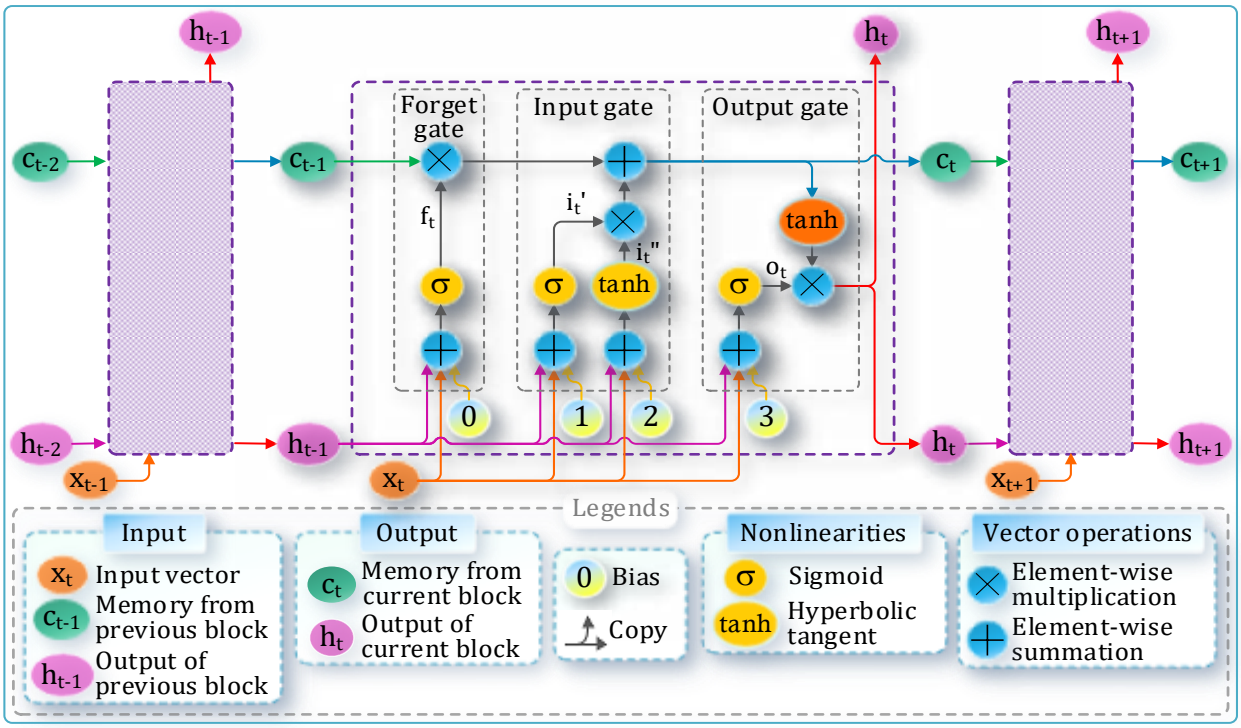


Fig. 16. A typical LSTM block

2. BASIC GRU ARCHITECTURE

Unlike LSTM, a GRU unit does not have any output gate; rather, it has only two gates – a) update gate and b) reset gate. The input and forget gates of LSTM are combined into an update gate in GRU. A GRU model uses the update gate to determine how much of the information from the previous blocks are to be forwarded to the next block. The update gate (z_t) at timestep t is defined by Eq. 14. Here, x_t is the input at timestep t, and h_{t-1} is the hidden state that holds the information of the previous t-1 units. W_z and U_z are the respective weights of x_t and h_{t-1}. σ is the activation function that keeps the value of z_t between 0 and 1. The reset gate (r_t), defined by Eq. 15, decides the amount of past information to forget. The current memory content (h'_t) uses the reset gate to keep the relevant information from the past. The current block's information is held by the final memory (h_t), which is thereafter passed to the next block. h'_t and h_t are estimated using Eq. 16 and 17, respectively, where Θ denotes the elementwise product. How much information to be retained from the current (h'_t) and previous (h_{t-1}) memory contents is determined by the update gate. A typical GRU architecture is shown in Fig. 17.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \quad (14)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \quad (15)$$

$$h'_t = \tanh(W x_t + r_t \odot U h_{t-1}) \quad (16)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \quad (17)$$

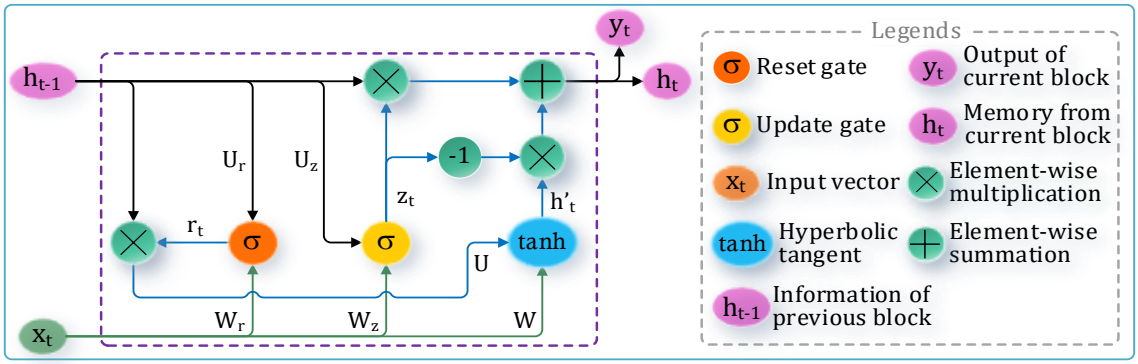


Fig. 17. A typical GRU block

3. CONVOLUTIONAL LSTM AND GRU MODELLING

To model the CLSTM and CGRU (convolutional GRU), we used two layers of the LSTM and GRU networks, respectively, with an input of frame groups with 50 samples, as shown in Fig. 12.

The layered representation of the CLSTM/CGRU prediction model is shown in Fig. 18. The objective of both models is to maximize the conditional probability of the convolutional features at the current timestep (C) over the input (N), for which the prediction is to be made, at the next timestep, as given in Eq. 18. This implies that the model optimizes the current prediction based on N. The timestep and the input frames can be modified during the training phase to check for improvements. In our experiment, the number of input vectors is quite low; therefore, we proceeded with a single stride over the input vectors for each timestep.

$$p(C|N) = \prod_{j=1}^{m} p(x_{(i+j)_j}, N) \quad (18)$$

where, m is the total number of days, x_{i_j} is the input frame for user j, and $i = 50$.

The input to the initial LSTM/GRU cell is the convolutional feature vector of the input data at timestep t, while the current LSTM/GRU cell output at timestep t is q_t, and the hidden states are h_t. The input to the next LSTM/GRU cell is the output of the previous LSTM/GRU cell, which then passes into a SoftMax layer for classification, as shown in Eq. 19.

$$p(x_t|(x_{t+1})) = SoftMax(W_f h_t) \quad (19)$$

where, W_f is a learnable parameter, and x_t and x_{t+1} are two adjacent input vectors to the model.

The output hidden states at the current timestep for CLSTM and CGRU are generated using Eq. 20 and Eq. 21, respectively.

$$h_t = LSTM(x_t, c_{t-1}, h_{t-1}) \quad (20)$$

$$h_t = GRU(x_t, h_{t-1}) \quad (21)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

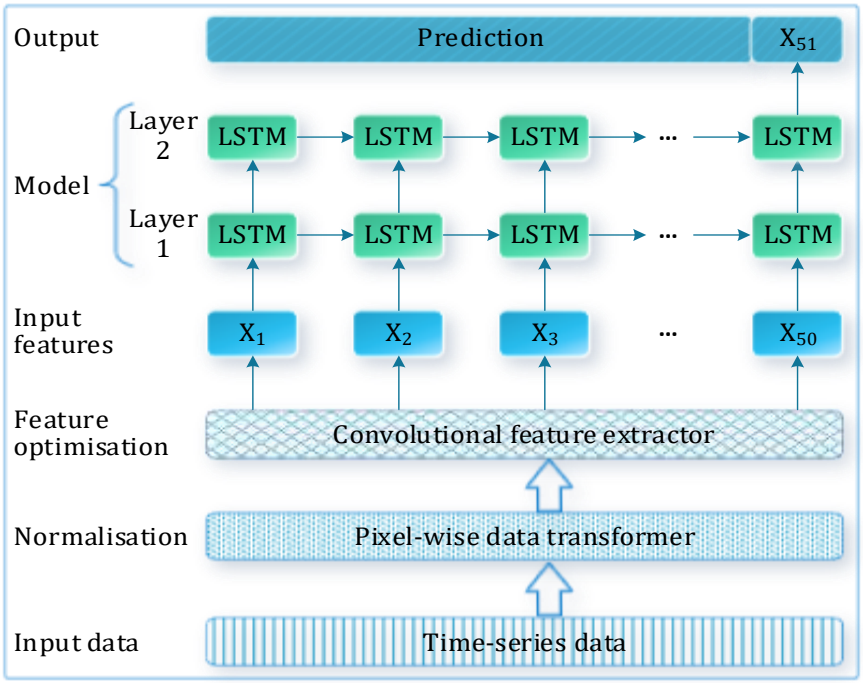


Fig. 18. Layered representation of the CLSTM prediction model

4. TRAINING

Training a model is used for making the model learn the trainable parameters and tuning the hyperparameters. The objective of the training phase is to decrease the error in the training dataset *d* of size *m*. The training objective function T_t is defined by Eq. 22.

$$T_t = \sum_{i=1}^{m} -\log p\,(p_i|a_i) \quad (22)$$

where, p_i is predicted output and a_i is the actual output.

H. EXPERIMENT, RESULTS, AND ANALYSIS

1. EXPERIMENTAL SETUP

The hardware specifications of the system used in training and testing the prediction model is as follows:

- Operating system: Windows 10 Professional
- CPU: AMD® Ryzen™ 7-3700X Processor
- RAM: 32GB DDR4
- GPU: NVIDIA GeForce® GTX 1080 Ti

The Windows version of the Python (64-bit) with IPython notebook [111] was used to build the models. Several important APIs including TensorFlow [112], NumPy packages [113], SciPy [114], scikit-learn [115] and Matplotlib [116] were used in the experiment. NVIDIA CUDA Version 9.1 [117] for Windows environment was used to avail GPU (graphics processing unit) computing.

2. PERFORMANCE MEASUREMENT METRICS

We assessed the performance of the proposed prediction model by calculating the followings metrics:

- Accuracy: It is the measurement of a prediction model's performance based on the total number of correct predictions made. Higher accuracy indicates better efficacy of the prediction model.
- Perplexity: It measures how well a probability model predicts an output and is often used for comparing probabilistic prediction models [118]. Although perplexity is popularly used in NLP (natural language processing) [119], here, we used it to represent the prediction loss for an input sample in the current timestep. A lower perplexity signifies a better prediction.
- Mean Absolute Error (MAE): An error, in this problem, is defined as the difference between the actual out-time and the predicted out-time of each SMD. MAE is calculated as the average of the prediction errors where all the error values are forced to be positive, as given in Eq. $MAE = \frac{1}{m}\sum_{i=1}^{m}|(p_i - a_i)|$ (23.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|(p_i - a_i)| \quad (23)$$

where,

Since

- Root Mean Squared Error (RMSE): It measures the root of the average of the errors' squares, as shown

i

n

R

E

F

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

important error measure for a model's performance if the main purpose of the model is prediction. Since RMSE tends to exaggerate large errors, it might be insightful when comparing different prediction methods. Also, it is easier to interpret because the RMSE values are in the same units as the samples. A lower RMSE value suggests a better prediction, while a zero value means no error in prediction.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(p_i - a_i)^2} \quad (24)$$

- R-squared (R^2): In comparison to RMSE, which is an absolute measure of correct prediction, R-squared is a relative measure of it. The value of R^2 ranges between 0 to 1, while a higher R^2 generally suggests a better prediction performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(a_i - p_i)^2}{\sum_{i=1}^{m}\left(a_i - \frac{1}{m}\sum_{j=1}^{m} a_j\right)^2} \quad (25)$$

where, m is the number of samples, p_i is predicted out-time, and a_i is the actual out-time.

3. TRAINING AND TESTING SPLIT

To conduct a prediction experiment and evaluate the model performance, the complete dataset is generally split into two parts as training and testing sets. This ensures that the model trains on the known data and is able to perform predictions properly for unknown data, which is validated using the test set. Though in this problem, the dataset is too small (150 data instances for each user) to split into two sub-datasets, we had to do it because no other data was available for validating the accuracy of the model in case of unfamiliar data. Hence, the existing data was split into training and testing sets in the ratio of 7:3, as it is found that 70% as the training set can sufficiently represent the data patterns. This splitting ratio is used throughout the experiment. Further, the sequence of the entire data is maintained properly after splitting.

4. ERROR ESTIMATION OF CLSTM

In this section, we estimate the prediction error of CLSTM while comparing the same with other prediction methods: ARIMA, LSTM, GRU, and CGRU. In Section I.D, we categorically stated that the traditional statistical prediction methods would not work well for this problem. However, to prove our claim, in this comparison, we included ARIMA, the most popularly time-series prediction method, along with other deep learning based prediction methods.

Fig. 19 shows the error estimation results in terms of MAE, RMSE, and R^2. The MAE and RMSE errors are calculated in minutes, while R^2 is presented in percentage. It can be observed that for each error estimation, CLSTM produces least errors than other methods. It was also observed that not only the CLSTM but other models also performed better if the size of the training dataset increased.

It can be further discerned that ARIMA exhibits the worst error estimations. This can be due to the fact that ARIMA cannot handle datasets with missing values. In this, the missing values should be handled by some fillers. Furthermore, ARIMA is suitable for short-time prediction because due to the absence of memory, the prediction window is very limited. That is why it fails in long-term prediction.

Since ARIMA generates a very high degree of prediction error which is unacceptable for our problem, we did not consider it for further comparative analysis.

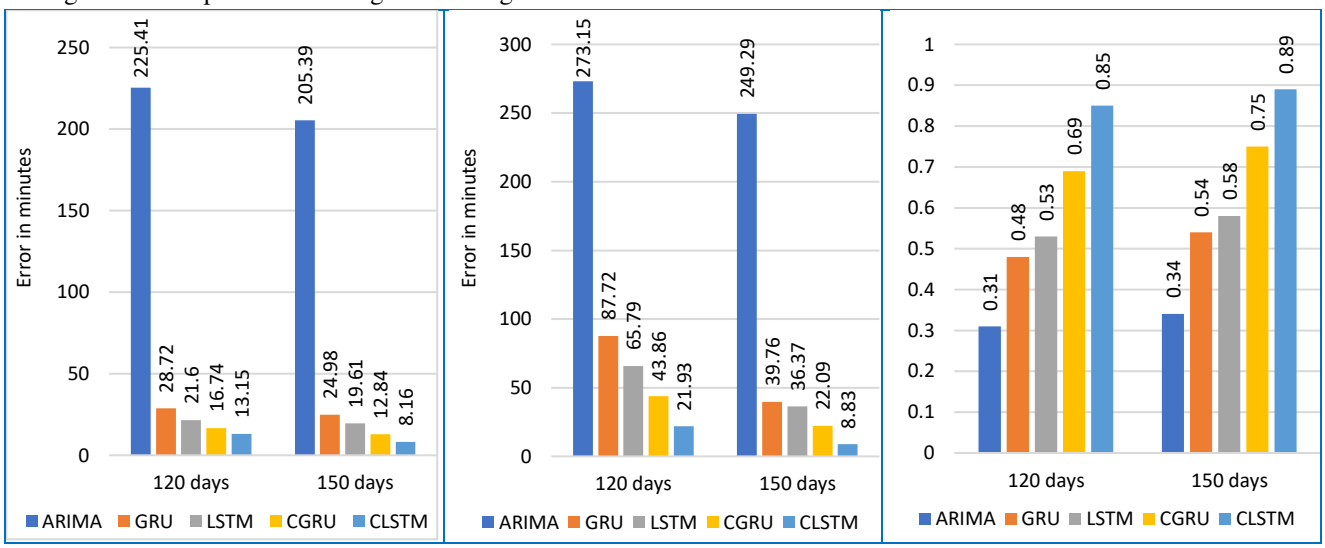


Fig. 19. Error comparison of CLSTM with ARIMA, GRU, LSTM, and CGRU based predictions: (a) MAE (b) RMSE, and (c) R^2

5. PREDICTION RESULTS USING CONVENTIONAL LSTM

The training and testing statistics of the LSTM prediction model are shown in Fig. 20. The training and testing accuracy for 150 days are 45.6 and 44.35, respectively, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

for 120 days, 41.9 and 32.89. The accuracy improvements of CLSTM over LSTM are 104.9% and 90.03% for 120 and 150 days, respectively.

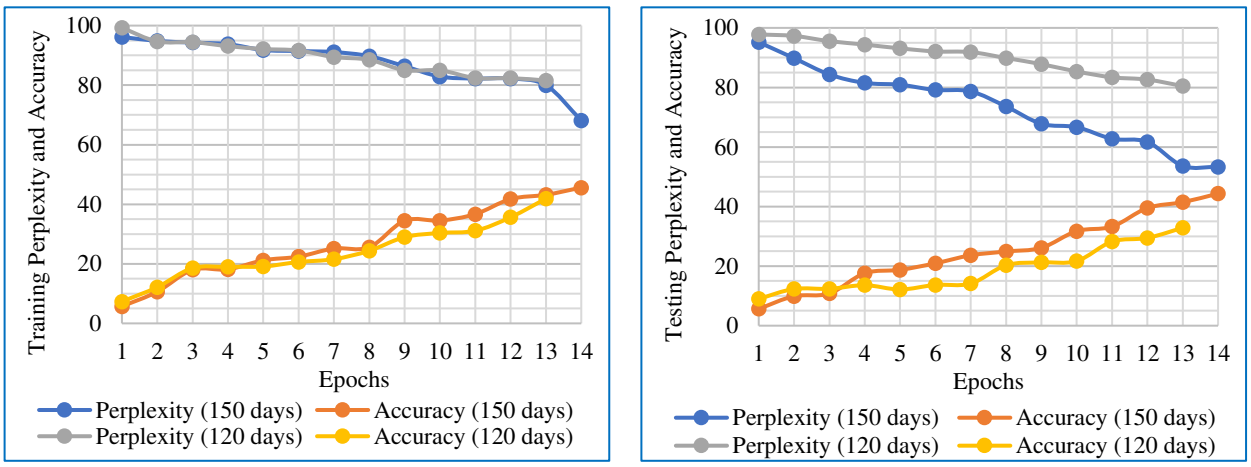


Fig. 20. Statistics of LSTM for two datasets of (a) Training and (b) Testing

6. PREDICTION RESULTS USING CONVOLUTIONAL LSTM

To evaluate the proposed CLSTM model's performance, we considered evaluating these metrics over 22 epochs because the training model's perplexity and accuracy did not improve after 22 epochs. The perplexity vs. accuracy graph for training and testing is shown in Fig. 21. It is observed that for 150 days of data, the achieved training accuracy of the model over 22 epochs is 89.97%, and the testing accuracy is 84.28%, while for 120 days, it is 80.44% and 67.39%, respectively. A difference of 5.69% and 13.05% are seen between the training and testing models in the two cases, which signifies that the model is not overfitting.

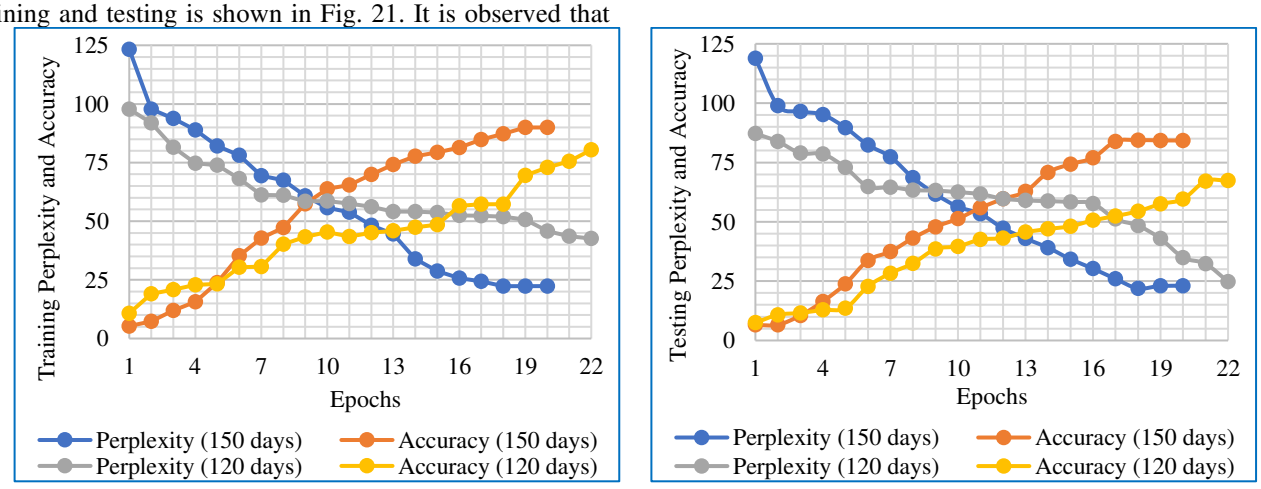


Fig. 21. Statistics of CLSTM for two datasets of (a) Training and (b) Testing

7. PREDICTION RESULTS USING CONVENTIONAL GRU

The training and testing statistics of the GRU prediction model are shown in Fig. 22. The training and testing accuracy for 150 days are 39.9 and 38.1, respectively, and for 120 days, 33.4 and 28.9. The accuracy improvements of CLSTM over GRU are 133.18% and 121.2% for 120 and 150 days, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

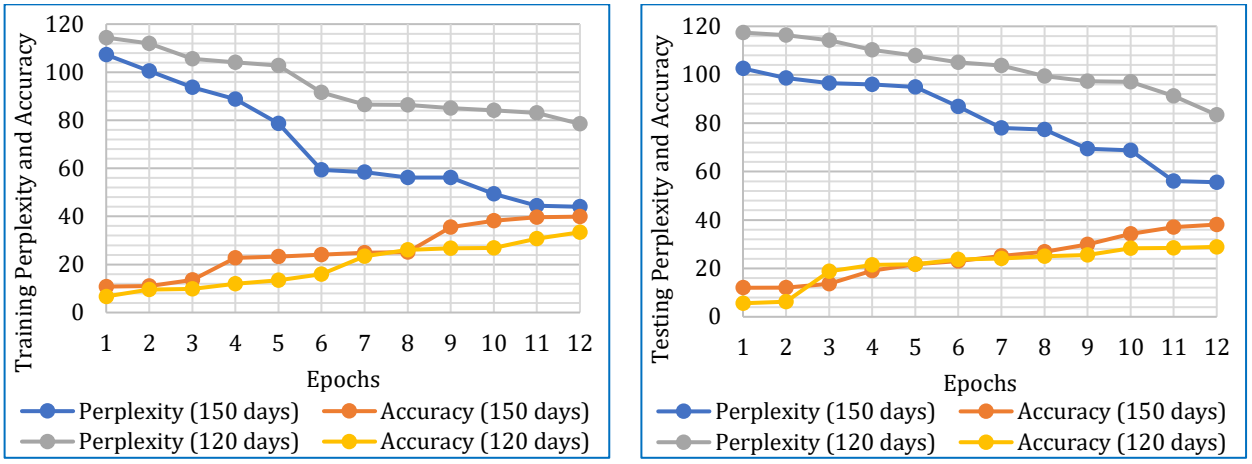


Fig. 22. Statistics of GRU for two datasets of (a) Training and (b) Testing

8. PREDICTION RESULTS USING CONVOLUTIONAL GRU

The training and testing statistics of the CGRU prediction model are shown in Fig. 23. The training and testing accuracy for 150 days are 79.7 and 76.8, respectively, and for 120 days, 69.8 and 66. The accuracy improvements of CLSTM over GRU are 2.11% and 7.48% for 120 and 150 days, respectively.

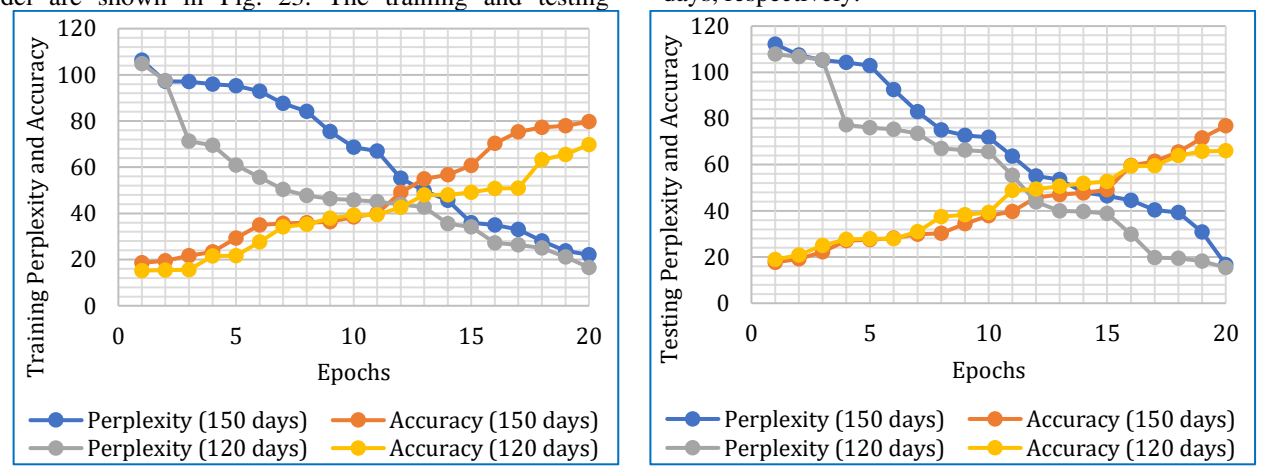


Fig. 23. Statistics of CGRU for two datasets of (a) Training and (b) Testing

9. COMPARING CLSTM WITH OTHER METHODS

The performance of the proposed CLSTM was compared with the other three prediction models. Fig. 24 shows the accuracy comparison between the proposed CLSTM model and the other compared models. It is observed that CLSTM has significantly higher accuracy over GRU and LSTM. However, there is not much difference between CLSTM and CGRU in terms of accuracy. This suggests that when the traditional LSTM and GRU are combined with our proposed convolutional feature extractor, they perform considerably better. This proves the efficacy of the proposed model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

(a) (b) (c)

Fig. 24. Accuracy comparison between (a) GRU and CLSTM (b) LSTM and CLSTM and (c) CGRU and CLSTM

To compare the model's sensitiveness towards the input data size, we checked the model's accuracy by varying set sizes. It is observed from Fig. 25 that all the models perform better with the larger data size. However, for the traditional LSTM and GRU models, the accuracy improvement with larger data set is comparatively greater than CLSTM and CGRU. It is always desirable to have a higher improvement percentage, but the final attained accuracy value also needs to be considered. This implies that even if the traditional models have the highest improvement percentage, their final attained accuracies are too low compared to the convolutional models. Furthermore, the accuracy improvement of CLSTM is much higher than CGRU, from which we can expect to have further higher accuracy for CLSTM with a larger data set.

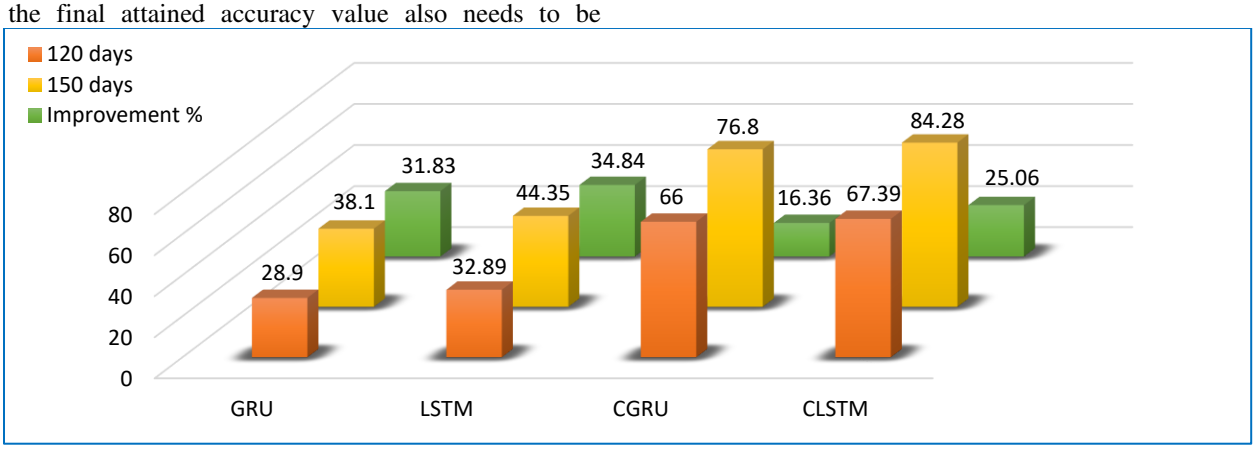


Fig. 25. Improvement percentage of testing accuracy of each model with respect to the number of days of data used

I. SMD SELECTION

After the initial ranking, the top-ranked (as per other criteria) SMD is forwarded to the prediction model along with its in-time and the job duration. The minimum required out-time is calculated by adding current time and job duration. The CLSTM model predicts the out-time against the in-time of the SMD and forwards it to the selection module. Now, if the predicted out-time is more than the minimum required out-time, then the selection module selects the SMD.

J. DISCUSSION

In this section, we present a brief discussion on two crucial aspects of the proposed solution method.

1. CONCERN OVER SPACE AND TIME COST OF LSTM

LSTM models provide satisfactory accuracy in predicting or forecasting. However, there is a general concern over LSTM models is that they usually involve high processing and memory costs because of linear layers present in each cell.

However, nowadays, the space requirement issue is not a big deal considering the availability of cheap and compact memory. Similarly, the processing requirement can be minimized by exploiting the parallel execution capability of the modern many-core GPUs, which are powerful enough to carry out computing-intensive jobs quickly.

Furthermore, the complexity of the machine learning model mainly contributed to its training process. During the testing, when the model is already built, the runtime is minimal. Hence, the time requirement of implementing the LSTM-based availability prediction process would be negligible, and the proposed system can be deployed for real-time purposes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103903, IEEE Access

In our proposed MCC system, the prediction model runs on a dedicated MCC coordinator. Moreover, as we considered a local MCC environment, the number of SMDs or users is limited. Also, the time span of data gathering is not much wide. Therefore, the data volume is not exceptionally huge. As a result, the hardware requirement would not be too extensive. In our experiment, we leveraged the parallel processing capability of NVIDIA GPU using CUDA API, and it was found quite acceptable.

However, if the MCC is implemented in an environment where such a powerful system is not present, or if the data size is hugely voluminous such as in a smart city scenario, the cloud services can always be availed. Today's commercial cloud services such as AWS, Google, etc., not only offer on-demand highly scalable GPU and TPU (tensor processing unit) stacks but they are also proffered with viable and affordable prices. Therefore, the LSTM model can be trained on the cloud, leveraging the powerful heterogeneous hardware environments to achieve significant speedups.

2. CNN FOR TEMPORAL DATA

It is observed that LSTM performs quite satisfactorily with higher accuracy in many forecasting applications while combined with CNN. In our experiment also, we observed significant performance improvement by using CLSTM compared to only LSTM. It is known that LSTM works well on temporal data, whereas CNN is designed to exploit the spatial correlation in the data and works well on the data having spatial features; they are not generally capable in efficient handling of complex and long temporal dependencies [120]. While in our SMD availability problem, the dataset is only temporal in nature. Even then, the CLSTM provides much better performance than LSTM. In our dataset, there is some inherent inconsistency that cannot be reflected using traditional LSTM. So, to avail the advantages of CNN, the sequential data is needed to be mapped somehow into a spatio-temporal pattern.

To the best of our knowledge, besides extracting the new features using CNN, the technicality behind mapping temporal or sequential data to spatio-temporal data for prediction performance improvement is still not quite obvious. However, from Fig. 10, it is intuitive that identifying the shape of the users' in- and out-time patterns may significantly improve the performance of the predictive model. The obvious reason behind this assumption is that the mobility pattern shape can capture the hidden features associated with the dataset. If we treat the users' in- and out-time patterns as an object, then acquiring the object's shape would allow us to extract the unknown features, as in the case of image analysis.

V. CONCLUSIONS AND FUTURE SCOPE

Due to unpredicted user mobility, the uncertainty of the availability of the SMDs (smart mobile devices), considered as computing resources, is a great challenge in maintaining QoS in mobile crowd computing (MCC). There is always a high probability that a task is assigned to a crowdworker, but it leaves the network without finishing it. In this paper, we proposed an SMD availability prediction method and an availability-aware crowdworker selection scheme for a local MCC where people join the MCC regularly and stay connected for varying periods. Before submitting the job to a crowdworker, the probability of the considered crowdworker being available until the job execution is finished is evaluated. If the job execution time is greater than the predicted availability duration, an alternative crowdworker is considered. This will improve the QoS of MCC by minimizing the job reassignment. Utilizing the real user mobility traces for a Wi-Fi access point deployed in a research lab, a convolutional LSTM based prediction method was applied to predict the out-time of the user for each in-time. The prediction model was implemented on two datasets of different volumes. It was able to forecast the availability better with an accuracy of 84.28% for the larger dataset, which suggests that with an increase in dataset size, the performance of the model improves significantly. Also, with the increase in the dataset, the error estimation of the model gets better. This justifies the correctness of the proposed model. The proposed model competes favorably against other compared models, viz. ARIMA, LSTM, GRU, and convolutional GRU for both datasets.

However, there is a scope for improvement of this work. Since the job completion time of different devices will be different, the variance of the completion time over various devices could be measured separately to make the model more accurate and justified.

Further, we considered only the longest duration of a user's presence time in the network for each day. This somehow curbs the effectiveness of the crowdworker selection. It may happen that a session duration of a crowdworker is sufficiently larger than a job to be assigned, but since this session is not this crowdworker's longest session on that day, it would not be listed as the probable candidate crowdworker, even it fulfills all other criteria adequately. Increasing the depth and, hence, dimension of the data frame may allow the model to consider all the sessions.

The prediction may further be enriched by considering the difference in the predicted out-time and the job duration to get a confidence value that may be used to rank the SMDs based on availability in the absence of any prior ranking scheme, where the minimum confidence threshold for selection criterion will depend on the particular application, job execution time, etc.

ACKNOWLEDGMENTS

We would like to sincerely thank Dr. Gautam Bandyopadhyay, Associate Professor, Dept. of Management Studies, National Institute of Technology Durgapur, India, for his valuable comments and suggestions to improve the manuscript. We also thankfully acknowledge the help of Mrs. Tumpa Banerjee, Assistant Professor, Dept. of

Computer Applications, Siliguri Institute of Technology, Siliguri, India, in modeling and calculating the prediction error of ARIMA.

REFERENCES

[1] H. Tamimi, N. AlMazrooei, S. Hoshang and F. Abu-Amara, "Factors Influencing Individuals to Switch from Personal Computers to Smartphones," in *5th HCT Information Technology Trends (ITT)*, Dubai, UAE, 2018.

[2] P. K. D. Pramanik, N. Sinhababu, B. Mukherjee, S. Padmanaban, A. Maity, B. K. Upadhyaya, J. B. Holm-Nielsen and P. Choudhury, "Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-art Review on Smartphone Battery and Energy Usage," *IEEE Access,* vol. 7, no. 1, pp. 182113-182172, 2019.

[3] P. K. D. Pramanik, P. Choudhury and A. Saha, "Economical Supercomputing thru Smartphone Crowd Computing: An Assessment of Opportunities, Benefits, Deterrents, and Applications from India's Perspective," in *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2017.

[4] P. K. D. Pramanik, S. Pal and P. Choudhury, "Smartphone Crowd Computing: A Rational Solution towards Minimising the Environmental Externalities of the Growing Computing Demands," in *Emerging Trends in Disruptive Technology Management*, R. Das, M. Banerjee and S. De, Eds., New York, Chapman and Hall/CRC, 2019, pp. 45-80.

[5] P. K. D. Pramanik, S. Pal and P. Choudhury, "Green and Sustainable High-Performance Computing with Smartphone Crowd Computing: Benefits, Enablers, and Challenges," *Scalable Computing: Practice and Experience,* vol. 20, no. 2, pp. 259-283, 2019.

[6] P. K. D. Pramanik, S. Pal, G. Pareek, S. Dutta and P. Choudhury, "Crowd Computing: The Computing Revolution," in *Crowdsourcing and Knowledge Management in Contemporary Business Environments*, R. Lenart-Gansiniec, Ed., IGI Global, 2018, pp. 166-198.

[7] A. Zhou, S. Wang, J. Li, Q. Sun and F. Yang, "Optimal mobile device selection for mobile cloud service providing," *Journal of Supercomputer,* vol. 72, no. 8, pp. 3222-3235, 2016.

[8] K. Habak, M. Ammar, K. A. Harras and E. Zegura, "Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge," in *IEEE 8th International Conference on Cloud Computing*, New York, USA, 2015.

[9] B. Zhu and Y. Wei, "Carbon price forecasting with a novel hybrid ARIMA and least squares support vector machines methodology," *Omega,* vol. 41, no. 3, pp. 517-524, 2013.

[10] N. H. Miswan, N. A. Ngatiman, K. Hamzah and Z. Z. Zamzamin, "Comparative performance of ARIMA and GARCH models in modelling and forecasting volatility of Malaysia market properties and shares," *Applied Mathematical Sciences,* vol. 8, no. 140, pp. 7001-7012, 2014.

[11] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Wuhan, China, 2016.

[12] N. N. Zakaria, M. Othman, R. Sokkalingam, H. Daud, L. Abdullah and E. A. Kadir, "Markov Chain Model Development for Forecasting Air Pollution Index of Miri, Sarawak," *Sustainability,* vol. 11, p. 5190, 2019.

[13] S. Elgharbi, M. Esghir, O. Ibrihich, A. Abarda, S. El Hajji and S. Elbernoussi, "Grey-Markov Model for the Prediction of the Electricity Production and Consumption," in *Big Data and Networks Technologies (BDNT 2019). Lecture Notes in Networks and Systems*, vol. 81, Y. Farhaoui, Ed., Springer, Cham, 2020, pp. 206-219.

[14] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[16] A. Yadava, C. K. Jhaa and A. Sharan, "Optimizing LSTM for time series prediction in Indian stock market," *Procedia Computer Science,* vol. 167, pp. 2091-2100, 2020.

[17] Y.-g. Zhang, J. Tang, Z.-y. He, J. Tan and C. Li, "A novel displacement prediction method using gated recurrent unit model with time series analysis in the Erdaohe landslide," *Natural Hazards,* vol. 105, pp. 783-813, 2021.

[18] Q. Tan, M. Ye, B. Yang, S. Liu, A. J. Ma, T. C.-F. Yip, G. L.-H. Wong and P. Yuen, "DATA-GRU: Dual-Attention Time-Aware Gated Recurrent Unit for Irregular Multivariate Time Series," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 34, no. 1, pp. 930-937, 2020.

[19] M. d. Caux, F. Bernardini and J. V. Viterbo, "Short-Term Forecasting in Bitcoin Time Series Using LSTM and GRU RNNs," in *Symposium on Knowledge Discovery, Mining and Learning (KDMILE 2020)*, Rio Grande, Brazil, 2020.

[20] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics,* vol. 36, pp. 193-202, 1980.

[21] Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel and D. Henderson, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems,* vol. 2, pp. 396-404, 1990.

[22] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[23] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang and W. Xu, "CNN-RNN: A Unified Framework for Multi-label Image Classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.

[24] A. Borovykh, S. Bohte and C. W. Oosterlee, "Conditional Time Series Forecasting with Convolutional Neural Networks," *arXiv,* no. 1703.04691v5, 2018.

[25] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao and S. Yan, "HCP: A Flexible CNN Framework for Multi-Label Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 9, pp. 1901-1907, 2016.

[26] C.-L. Liu, W.-H. Hsaio and Y.-C. Tu, "Time Series Classification With Multivariate Convolutional Neural Network," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 6, pp. 4788-4797, 2019.

[27] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy,* vol. 182, no. Sept, pp. 72-81, 2019.

[28] I. E. Livieris, E. Pintelas and P. Pintelas, "A CNN–LSTM model for gold price time-series forecasting," *Neural Computing and Applications,* vol. 32, pp. 17351-17360, 2020.

[29] I. E. Livieris, E. Pintelas, N. Kiriakidou and S. Stavroyiannis, "An Advanced Deep Learning Model for Short-Term Forecasting U.S. Natural Gas Price and Movement," in *Applications and Innovations (AIAI 2020), IFIP WG 12.5, International Workshopsmhdw 2020 and 5G-PINE 2020*, Neos Marmaras, Greece, 2020.

[30] N. Xue, I. Triguero, G. P. Figueredo and D. Landa-Silva, "Evolving Deep CNN-LSTMs for Inventory Time Series Prediction," in *IEEE Congress on Evolutionary Computation (CEC)*, Wellington, New Zealand, 2019.

[31] A. Mtibaa, A. Fahim, K. A. Harras and M. H. Ammar, "Towards resource sharing in mobile device clouds: power balancing across mobile devices," in *2nd ACM SIGCOMM workshop on Mobile cloud computing (MCC '13)*, Hong Kong, China, 2013.

[32] S. W. Loke, K. Napier, A. Alali, N. Fernando and W. Rahayu, "Mobile Computations with Surrounding Devices: Proximity Sensing and MultiLayered Work Stealing," *ACM Transactions on Embedded Computing Systems,* vol. 14, no. 2, 2015.

[33] M. Hirsch, C. Mateos and A. Zunino, "Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey," *Future Generation Computer Systems,* 2018.

[34] D. M. Shila, W. Shen, Y. Cheng, X. Tian and X. S. Shen, "AMCloud: Toward a Secure Autonomic Mobile Ad Hoc Cloud Computing System," *IEEE Wireless Communications,* vol. 24, no. 2, pp. 74-81, 2017.

[35] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran and S. Guizani, "Mobile ad hoc cloud: A survey," *Wireless Communications and Mobile Computing,* vol. 16, no. 16, pp. 2572-2589, 2016.

[36] V. Balasubramanian and A. Karmouch, "An infrastructure as a Service for Mobile Ad-hoc Cloud," in *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, USA, 2017.

[37] A. Khalifa, M. Azab and M. Eltoweissy, "Resilient hybrid Mobile Ad-hoc Cloud over collaborating heterogeneous nodes," in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Miami, USA, 2014.

[38] J. Brevik, D. Nurmi and R. Wolski, "Automatic methods for predicting machine availability in desktop Grid and peer-to-peer systems," in *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, Chicago, USA, 2004.

[39] A. Andrzejak, D. Kondo and D. P. Anderson, "Ensuring Collective Availability in Volatile Resource Pools Via Forecasting," in *Managing Large-Scale Service Deployment (DSOM 2008), Lecture Notes in Computer Science*, vol. 5273, F. De Turck, W. Kellerer and G. Kormentzas, Eds., Berlin, Heidelberg, Springer, 2008, pp. 149-161.

[40] S. S. Vaithiya and S. M. S. Bhanu, "Mobility and Battery Power Prediction Based Job Scheduling in Mobile Grid Environment," in *International Conference on Parallel Distributed Computing Technologies and Applications (PDCTA 2011)*, 2011.

[41] V. V. Selvi, S. Sharfraz and R. Parthasarathi, "Mobile Ad Hoc Grid Using Trace Based Mobility Model," in *Advances in Grid and Pervasive Computing (GPC 2007)*, Paris, France, 2007.

[42] M. Á. Sipos and P. Ekler, "Predicting Availability of Mobile Peers in Large Peer-to-Peer Networks," in *3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, Budapest, Hungary, 2013.

[43] P. K. D. Pramanik, G. Bandyopadhyay and P. Choudhury, "Predicting Relative Topological Stability of Mobile Users in a P2P Mobile Cloud," *SN Applied Sciences,* vol. 2, no. 11, article no. 1827, 2020.

[44] S. C. Haryanti and R. F. Sari, "Improving Resource Allocation Performance in Mobile Ad Hoc Grid with Mobility Prediction," in *International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, Taipei, Taiwan, 2014.

[45] U. Farooq and W. Khalil, "A Generic Mobility Model for Resource Prediction in Mobile Grids," in *Proceedings of the International Symposium on Collaborative Technologies and Systems*, Las Vegas, USA, 2006.

[46] S. Dargan, M. Kumar, M. R. Ayyagari and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Archives of Computational Methods in Engineering,* vol. 27, pp. 1071-1092, 2020.

[47] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen and S. S. Iyengar, "A Survey on Deep Learning: Algorithms, Techniques, and Applications," *ACM Computing Surveys,* vol. 51, no. 5, pp. 1-36, 2019.

[48] C. A. Dhawale, K. Dhawale and R. Dubey, "A Review on Deep Learning Applications," in *Deep Learning Techniques and Optimization Strategies in Big Data Analytics*, J. J. Thomas, P. Karagoz, B. B. Ahamed and P. Vasant, Eds., USA, IGI GLobal, 2020, pp. 21-31.

[49] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing,* vol. 107, pp. 241-265, 2018.

[50] R. Miotto, F. Wang, S. Wang, X. Jiang and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics,* vol. 19, no. 6, pp. 1236-1246, 2018.

[51] B. Yang and Y. Xu, "Applications of deep-learning approaches in horticultural research: a review," *Horticulture Research,* vol. 8 (Article number: 123), 2021.

[52] A. Carrio, C. Sampedro, A. Rodriguez-Ramos and P. Campoy, "A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles," *Journal of Sensors,* vol. 2017 (Article ID 3296874), 2017.

[53] J. Huang, J. Chai and S. Cho, "Deep learning in finance and banking: A literature review and classification," *Frontiers of Business Research in China,* vol. 14 (Article number: 13) , 2020.

[54] A. Sarraf, M. Azhdari and S. Sarraf, "A Comprehensive Review of Deep Learning Architectures for Computer Vision Applications," *American Scientific Research Journal for Engineering, Technology, and Sciences,* vol. 77, no. 1, pp. 1-29, 2021.

[55] Y. Zhang, J. Yan, S. Chen, M. Gong, D. Gao, M. Zhu and W. Gan, "Review of the Applications of Deep Learning in Bioinformatics," *Current Bioinformatics,* vol. 15, no. 8, pp. 898-911, 2020.

[56] M. I. Tariq, N. A. Memon, S. Ahmed, S. Tayyaba, M. T. Mushtaq, N. A. Mian, M. Imran and M. W. Ashraf, "A Review of Deep

Learning Security and Privacy Defensive Techniques," *Mobile Information Systems,* vol. 2020 (Article ID 6535834), 2020.

[57] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing,* vol. 323, no. 5, pp. 203-213, 2019.

[58] J. Y. Choi and B. Lee, "Combining LSTM Network Ensemble via Adaptive Weighting for Improved Time Series Forecasting," *Mathematical Problems in Engineering,* vol. 2018 (Article ID 2470171), 2018.

[59] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems,* vol. 11, no. 2, 3, pp. 68-75, 2017.

[60] P. B. Weerakody, K. W. Wong, G. Wang and W. Ela, "A review of irregular time series data handling with gated recurrent neural networks," *Neurocomputing,* vol. 441, pp. 161-178, 2021.

[61] Z. Che, S. Purushotham, K. Cho, D. Sontag and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," *Scientific Reports,* vol. 8, p. 6085, 2018.

[62] K. Lu, X. R. Meng, W. X. Sun, R. G. Zhang, Y. K. Han, S. Gao and D. Su, "GRU-based Encoder-Decoder for Short-term CHP Heat Load Forecast," *IOP Conference Series: Materials Science and Engineering,* vol. 392, no. 6, p. 062173, 2018.

[63] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017.

[64] J. M.-T. Wu, Z. Li, N. Herencsar, B. Vo and J. C.-W. Lin, "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators," *Multimedia Systems,* 2021.

[65] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," *PLoS ONE,* vol. 14, no. 2, p. e0212320, 2019.

[66] W. Lu, J. Li, Y. Li, A. Sun and J. Wang, "A CNN-LSTM-Based Model to Forecast Stock Prices," *Complexity,* vol. 2020 (Article ID 6622927), 2020.

[67] Y. Li and W. Dai, "Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model," *The Journal of Engineering,* vol. 2020, no. 13, pp. 344-347, 2020.

[68] T. Ni, L. Wang, P. Zhang, B. Wang and W. Li, "Daily tourist flow forecasting using SPCA and CNN-LSTM neural network," *Concurrency and Computation: Practice and Experience,* vol. 33, no. 5, p. e5980, 2021.

[69] J. Zhao, J. Lin, S. Liang and M. Wang, "Sentimental prediction model of personality based on CNN-LSTM in a social media environment," *Journal of Intelligent & Fuzzy Systems,* vol. 40, no. 2, pp. 3097-3106, 2021.

[70] T.-Y. Kim and S.-B. Cho, "Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks," in *Intelligent Data Engineering and Automated Learning (IDEAL 2018). Lecture Notes in Computer Science*, vol. 11314, H. Yin, D. Camacho, P. Novais and A. Tallón-Ballesteros, Eds., Springer, Cham, 2018, pp. 481-490.

[71] M. Tovar, M. Robles and F. Rashid, "PV Power Prediction, Using CNN-LSTM Hybrid Neural Network Model. Case of Study: Temixco-Morelos, México," *Energies,* vol. 13, no. 24, p. 6512, 2020.

[72] H. Kuang, Q. Guo, S. Li and H. Zhong, "Short-term wind power forecasting model based on multi-feature extraction and CNN-LSTM," *IOP Conference Series: Earth and Environmental Science,* vol. 702, p. 012019, 2021.

[73] C. Ding, G. Wang, X. Zhang, Q. Liu and X. Liu, "A hybrid CNN-LSTM model for predicting PM2.5 in Beijing based on spatiotemporal correlation," *Environmental and Ecological Statistics,* 2021.

[74] T. Li, M. Hua and X. Wu, "A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5)," *IEEE Access,* vol. 8, pp. 26933-26940, 2020.

[75] W. He, J. Li, Z. Tang, B. Wu, H. Luan, C. Chen and H. Liang, "A Novel Hybrid CNN-LSTM Scheme for Nitrogen Oxide Emission Prediction in FCC Unit," *Mathematical Problems in Engineering,* vol. 2020 (Article ID 8071810), 2020 .

[76] W. Boulila, H. Ghandorh, M. A. Khan, F. Ahmed and J. Ahmad, "A novel CNN-LSTM-based approach to predict urban expansion," *Ecological Informatics,* vol. 64, no. Sept, p. 101325, 2021.

[77] K. Cao, H. Kim, C. Hwang and H. Jung, "CNN-LSTM Coupled Model for Prediction of Waterworks Operation Data," *Journal of Information Processing Systems,* vol. 14, no. 6, pp. 1508-1520, 2018.

[78] P. K. Jonnakuti and U. B. T. V. Sai, "A hybrid CNN-LSTM based model for the prediction of sea surface temperature using time-series satellite data," in *22nd EGU General Assembly (EGU2020-817)*, Online, 2020.

[79] R. Chen, X. Wang, W. Zhang, X. Zhu, A. Li and C. Yang, "A hybrid CNN-LSTM model for typhoon formation forecasting," *Geoinformatica,* vol. 23, no. 3, pp. 375-396, 2019.

[80] S. Khaki, L. Wang and S. V. Archontoulis, "A CNN-RNN Framework for Crop Yield Prediction," *Frontiers in Plant Science,* vol. 10, p. 1750, 2020.

[81] Md. ZabirulIslam, Md. Milon Islam, Amanullah Asraf, "A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images," *Informatics in Medicine Unlocked,* vol. 20, p. 100412, 2020.

[82] A. G. Dastider, F. Sadik and S. A. Fattah, "An integrated autoencoder-based hybrid CNN-LSTM model for COVID-19 severity prediction from lung ultrasound," *Computers in Biology and Medicine,* vol. 132, no. May, p. 104296, 2021.

[83] S. Dutta, S. K. Bandyopadhyay and T.-H. Kim, "CNN-LSTM Model for Verifying Predictions of Covid-19 Cases," *Asian Journal of Research in Computer Science,* vol. 5, no. 4, pp. 25-32, 2020.

[84] S. A. Rahman and D. A. Adjeroh, "Deep Learning using Convolutional LSTM estimates Biological Age from Physical Activity," *Scientific Reports,* vol. 9 (Article number: 11425), 2019.

[85] Y. Zhang, J. Yao and H. Guan, "Intelligent Cloud Resource Management with Deep Reinforcement Learning," *IEEE Cloud Computing,* vol. 4, pp. 60-69, 2017.

[86] Y. Lu, L. Liu, J. Panneerselvam, B. Yuan, J. Gu and N. Antonopoulos, "A GRU-Based Prediction Framework for Intelligent Resource Management at Cloud Data Centres in the Age of 5G," *IEEE Transactions on Cognitive Communications and Networking,* vol. 6, no. 2, pp. 486-498, 2020.

[87] H. Jing, Y. Zhang, J. Zhou, W. Zhang, X. Liu, G. Min and Z. Zhang, "LSTM-Based Service Migration for Pervasive Cloud

Computing," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax, Canada, 2018.

[88] Y. Zhu, W. Zhang, Y. Chen and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking,* vol. 2019 (Article number: 274), 2019.

[89] J. Kumar, R. Goomer and A. K. Singh, "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters," *Procedia Computer Science,* vol. 125, pp. 676-682, 2018.

[90] Anupama K C, Shivakumar B R, Nagaraja R, "Resource Utilization Prediction in Cloud Computing using Hybrid Model," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 4, pp. 373-381, 2021.

[91] H. Li, K. Ota and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network,* vol. 32, no. 1, pp. 96-101, 2018.

[92] J. Shuja, K. Bilal, W. Alasmary, H. Sinky and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge," *Journal of Network and Computer Applications,* vol. 181, p. 103005, 2021.

[93] J. Violos, E. Psomakelis, D. Danopoulos, S. Tsanakas and T. Varvarigou, "Using LSTM Neural Networks as Resource Utilization Predictors: The Case of Training Deep Learning Models on the Edge," in *Economics of Grids, Clouds, Systems, and Services. GECON 2020. Lecture Notes in Computer Science, vol*, vol. 12441, K. Djemame, J. Altmann, J. Á. Bañares, O. Agmon Ben-Yehuda, V. Stankovski and B. Tuffin, Eds., Springer, Cham, 2020, pp. 67-74.

[94] F. Hussain, S. A. Hassan, R. Hussain and E. Hossain, "Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges," *IEEE Communications Surveys & Tutorials,* vol. 22, no. 2, pp. 1251-1275, 2020.

[95] B. Gu, X. Zhang, Z. Lin and M. Alazab, "Deep Multi-Agent Reinforcement Learning-Based Resource Allocation for Internet of Controllable Things," *IEEE Internet of Things Journal,* vol. 8, no. 5, pp. 3066-3074, 2021.

[96] K. Li, W. Ni, E. Tovar and A. Jamalipour, "Deep Q-Learning based Resource Management in UAV-assisted Wireless Powered IoT Networks," in *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.

[97] U. Challita, L. Dong and W. Saad, "Proactive Resource Management for LTE in Unlicensed Spectrum: A Deep Learning Perspective," *IEEE Transactions on Wireless Communications,* vol. 17, no. 7, pp. 4674-4689, 2018.

[98] R. C. Bhaddurgatte, B. P. Vijaya Kumar and S. M. Kusuma, "Machine Learning and Prediction-Based Resource Management in IoT Considering Qos," *International Journal of Recent Technology and Engineering,* vol. 8, no. 2, pp. 687-694, 2019.

[99] Y. Peng, G. Zhang, J. Shi, B. Xu and L. Zheng, "SRA-LSTM: Social Relationship Attention LSTM for Human Trajectory Prediction," *arXiv:2103.17045v1 [cs.CV],* 2021.

[100] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.

[101] H. Manh and G. Alaghband, "Scene-LSTM: A Model for Human Trajectory Prediction," *arXiv:1808.04018v2 [cs.CV],* 2019.

[102] N. Nikhil and B. T. Morris, "Convolutional Neural Network for Trajectory Prediction," in *Computer Vision – ECCV 2018 Workshops. Lecture Notes in Computer Science*, vol. 11131, L. Leal-Taixé and S. Roth, Eds., Springer, Cham, 2019, pp. 186-196.

[103] X. Song, K. Chen, X. Li, J. Sun, B. Hou, Y. Cui, B. Zhang, G. Xiong and Z. Wang, "Pedestrian Trajectory Prediction Based on Deep Convolutional LSTM Network," *IEEE Transactions on Intelligent Transportation Systems,* 2020.

[104] G. Xie, A. Shangguan, R. Fei, W. Ji, W. Ma and X. Hei, "Motion trajectory prediction based on a CNN-LSTM sequential model," *SCIENCE CHINA Information Sciences,* vol. 63, no. 11, p. 212207, 2020.

[105] H. R. Pamuluri, "Predicting User Mobility using Deep Learning Methods," Master's Thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 2020.

[106] C. Cui, M. Zhao and K. Wong, "An LSTM-Method-Based Availability Prediction for Optimized Offloading in Mobile Edges," *Sensors (Basel),* vol. 19, no. 20, p. 4467, 2019.

[107] R. Li, C. Wang, Z. Zhao, R. Guo and H. Zhang, "The LSTM-Based Advantage Actor-Critic Learning for Resource Management in Network Slicing With User Mobility," *IEEE Communications Letters,* vol. 24, no. 9, pp. 2005-2009, 2020.

[108] P. K. D. Pramanik, N. Sinhababu, A. Nayyar and P. Choudhury, "Predicting Device Availability in Mobile Crowd Computing using ConvLSTM," in *7th International Conference on Optimization and Applications*, Wolfenbüttel, Germany, May 2021.

[109] François Chollet and others, "Keras," 2015. [Online]. Available: https://keras.io/. [Accessed 13 February 2021].

[110] R. Muthukrishnan and R. Rohini, "LASSO: A Feature Selection Technique In Predictive Modeling for Machine Learning," in *IEEE International Conference on Advances in Computer Applications (ICACA)*, Coimbatore, India, 2016.

[111] F. Perez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science & Engineering,* vol. 9, no. 3, 2007.

[112] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Google, 2015.

[113] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, et al., "Array programming with NumPy," *Nature,* vol. 585, pp. 357-362, 2020.

[114] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods,* vol. 17, pp. 261-72, 2020.

[115] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* vol. 12, no. Oct, pp. 2825-2830, 2011.

[116] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[117] NVIDIA, P. Vingelmann and F. H. Fitzek, "CUDA, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit. [Accessed 3 July 2021].

[118] W. Shen, Z. Wei, C. Yang and R. Zhang, "Travel pattern modelling and future travel behaviour prediction based on GMM and GPR," *Int. J. Simulation and Process Modelling,* vol. 13, no. 6, pp. 548-556, 2018.

[119] F. Jelinek, R. L. Mercer, L. R. Bahl and J. K. Baker, "Perplexity - a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America,* vol. 62, no. S1, p. S63, 1977.

[120] Y. Bengio, A. Courville and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, pp. 1798-1828, 2013.