LSTM Network for Transportation Mode Detection

Sachin Kumar[1], Agam Damaraju[1], Aditya Kumar[1], Saru Kumari[2], Chien-Ming Chen[3]

[1] Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, India
[2] Department of Mathematics, Chaudhary Charan Singh University, India
[3] College of Computer Science and Engineering, Shandong University of Science and Technology, China
imsachingupta@rediffmail.com, agamdamaraju@hotmail.com, adityakumar24jun@gmail.com, saryusiirohi@gmail.com, chienmingchen@ieee.org

Abstract

The study of Transportation Mode Detection (TMD) has become a popular research field in recent years. It will be a crucial part of Smart mobility and Smart cities in upcoming years. In our study, using the approach of TMDataset[1], we have gathered the data from different user's Smartphones up to 5 different transportation modes. However, as the raw data contains noise, we use Feature Engineering to extract useful features from the raw dataset and convert it into different feature frames to feed into a deep learning model called Long Short-Term Memory (LSTM). We used different sized feature frames to input the LSTM network for efficient transportation mode detection and achieved up to 98% classification accuracy for five transportation modes.

Keywords: Transportation mode, Deep learning, LSTM, Feature engineering, Intelligent transportation system

1 Introduction

Surveys in the travel domain are crucial for transportation engineers and researchers to understand human behavior. It can help to develop and maintain transportation systems.

These surveys help to gather the travel data, including the origin, destination, duration, and other factors. This data must be used to improve travel understanding concerning the location and choice [1]. This data collection plays an essential role in inspecting transportation behavior. This data type provides more elaborated knowledge on travel patterns and selections during a long period to those extracted from traditional survey methods. One of its purposes is to make arrangements for transportation in rural and urban areas.

Researchers are focusing on citizen movement in regular life to manage city transportation effectively. The details of these points would help the government estimate the needs of a city or a country [2]. The detection of transportation types may also allow us to showcase advertisements for the required users. For example, for a person traveling in a car, car service advertisements could be shown, or for a person who is transporting on a bus, PCs and books can be advertised. The main target of transportation mode for locating systems [3-4] is location-based services [5].

Smartphone sensors' improved capabilities, combined with their easy programming and effective distribution channels, helped Smartphone develop into an impactful tool for efficiently monitoring travel behavior. Instead of the accuracy and utilization of Smartphone sensors, there remain challenges to overcome. Pointing to the data size and complexity, more advanced algorithms are required to gather travel information.

In section 2 will review the related work of transportation mode detection and mention its limitations. In Section 3, we describe our approach for overcoming the drawbacks. We will discuss the data collection and its use in transportation mode detection in section 4. Following our proposed methodology, in Section 5. The detailed study with experiment results and model comparison is in Section 6, section 7, and section 8. Eventually, we state the overall conclusion of this research in Section 8.

2 Related Work

Here, we list the methods and the sources of data used in related studies. We present the relevant work on transportation mode detection. Then we identify some limitations in various aspects for further improvement in the performance of travel mode detection.

TMD can be used as a significant step in activity recognition studies. It includes the two main tasks: (i) determining the movement of participants; (ii) what type of transportation methods one uses; for example, the study conducted by Broach et al. [6] shows that

*Corresponding Author: Chien-Ming Chen; E-Mail: chienmingchen@ieee.org
DOI: 10.53106/160792642021072204016
[1] http://cs.unibo.it/projects/us-tm2017/tutorial.html

transportation mode can be efficiently detected using the Multinomial Logit Model. The authors used four transportation modes: Walk, Bike, Auto, and transit, and collected the data using accelerometer and GPS sensors. Their proposed model performed slightly efficiently, and the accuracy reached up to 91%. However, Wang et al. [7] took a traditional approach for transportation mode detection. They used a random forest technique that was able to achieve up to 93% accuracy. However, they have collected the data from the GPS of six different transportation modes: walking, bicycle, e-bike, bus, car, and subway.

Another transportation mode detection study was done by Xiao et al. [8]. They have used the Continuous Hidden Markov Model and the GPS data of five transportation modes: walk, bicycle, e-bicycle, bus, and car. The authors used the feature extraction process has been used by the authors for the collection of various features such as Average speed, 95th percentile speed, average absolute acceleration, traveled distance, low-speed point rate, and average heading change. Using a continuous hidden Markov model classifier, the authors achieved up to 94% accuracy.

An attempt by Soares et al. [9] takes the help of AutoML for selecting algorithms and optimizing the hyperparameters with the data from sensors employed as input variants on the TMDataset[1]. In the Auto ML component, the authors performed meta-learning using AutoSklearn [10] to determine the configurations that will get tested. The authors compared the accuracy of classification efficiently up to 97% while training the machine learning classifiers using the sensors' features.

Reviewing all these studies, the common challenge while using machine learning algorithms lies in configuring their hyperparameters and the data selected by us. Setting the parameters for the extracted feature's data can directly affect the generated model's accuracy and training loss than the machine learning algorithm itself [11]. That is why Feature Engineering plays a crucial role in any transportation mode detection process. The choices were made based on some statistical insights and some other previous works; this led authors to create the Automated Machine Learning frameworks to automatically identify machine learning algorithms' filtered configuration.

In terms of Deep Learning techniques for TMD, Soares et al. [12] performed a study using SFNN, DFNN, and LSTM networks. They used orientation sensors and collected the data from 16 users. For data preparation, they extracted some time domain as well as some frequency-domain features. Eventually, they performed the classification of five transportation modes, i.e., still, walking, train, bus, and car, and gained an overall 90% performance accuracy.

2.1 Limitations

Previously, many studies have achieved high detection rates. However, there remain some limitations with sample size, feature selection to classify the modes, and quantity of extracted features. Feature selection should be the higher priority, which is necessary before implementing classification to the transportation modes. Proper feature selection could increase detection accuracy and decrease the algorithm's complexity to a significant level.

3 Our Approach

Several Deep Learning architectures are available in the literature, and Each works well with a particular type of data set; for example, Convolutional Neural Networks works well with image datasets [13-14]. Since we experiment with sensor responses concerning time, we use the Recurrent Neural Networks (RNNs) as RNNs support sequential data efficiently [14]. There are two popular versions of RNNs, i.e., Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). Since LSTMs outperform in comparison to GRUs [14] and can avoid the vanishing gradient problem [14].

Our work collected the 226 labeled data files of 5 different transportation modes of 16 users with nine sensors using the TMD Data collection technique described by the University of Bologna researchers. We apply data cleaning, Window Partitioning, and feature extraction methods to extract crucial features and frame the collected data for sequence learning of the model. We use normalization and feed the prepared data in our LSTM model for scaling the values of different parameters of the stated data. For loss convergence, Hyperparameter optimization is used after getting optimal accuracy by observing the results.

We have also compared our work with Soares et al. [12] that uses Bayesian Network as a classifier. The authors use Knowledge Discovery in Database (KDD) technique to perform Transportation Mode Detection. The overall accuracy metrics could only reach up to 91%. The Precision, recall, and f-score metrics could reach only up to 44%, 50%, and 47%, respectively. In our work, before feeding the data from sensors to the LSTM classifier, we use feature vector framing, i.e., grouping extracted features in different vector frames and the Normalization technique following an additional Feature Engineering technique. Our model performed comparatively well with accuracy, Precision, Recall, and f-score of 98% in each.

In brief, our primary contribution as listed:

- We have taken the LSTM network as RNN's suffer vanishing gradient problem [15], and Convolutional Neural Networks works well with image datasets.
- Before feeding the data from sensors to the LSTM classifier, we use feature vector framing, i.e., grouping extracted features in different vector frames and Normalization technique following an additional Feature Engineering technique.

- We also compare our study with some existing studies that used SFNN, DFNN, LSTM, AutoML frameworks, and other traditional machine learning techniques.
- Our model performed comparatively well with accuracy, Precision, Recall, and f-score of 98% in each.

4 Data Collection Technique for Transportation Mode Detection

Researchers of the University of Bologna demonstrated the TMD data collection technique and created a sample TMDataset[1]. We have collected our transportation mode data using the same process described in TMDataset[1].

4.1 Role of Sensors in Data Collection

This technique consists of some phases. Data preprocessing at an initial stage is one of them, including data cleaning operations. To make the values of speed and sound sensors positive, we deleted the measures from the sensors. Further, the sensors with single data value outputs were used directly. In this technique, we will use an orientation-independent metric applicable to sensors based on coordinate systems. We can find the magnitude using equation (1).

$$S(magnitude) = |v| = \sqrt{(v_{(x,s)}^2 + v_{(y,s)}^2 + v_{(z,s)}^2)} \quad \textbf{(1)}$$

Here, $v_{x,s}$, $v_{y,s,}$ and $v_{z,s}$ are the given values of sensors on the x, y, and z axes. For the processing of these sensor data, samples need to be cut in time windows. The size of the time window depends on the types of actions to be recognized. As per our data samples, we came up with two optimal window sizes, i.e., 0.5 seconds and 5 seconds. However, for complex activity recognition, it is advisable to use a large window size [16]. Hence, we consider 5 seconds of window size.

Apart from the orientation-based sensors, we use some proximity sensors and ambient sensors. This method helps to obtain the dataset into 5 seconds time windows. We have also extracted different features such as the mean, standard deviation, maximum, and minimum from these sensors. We used two resampling methods, i.e., downsampling and Up-sampling. However, we removed bias and infrequently occurred classes with Down-sampling [17] without making the dataset denser.

4.2 Transportation Modes of the Dataset

The dataset has 226 labeled files.

Before Down-sample the dataset, as shown in Figure 1 and Figure 2, the dataset consists of 16 users, and the data collected using the android application installed in their Smartphones observed, 24% were not moving, 26% were walking, 25% were using the car, 20% were using train and only 5% of users using the bus.

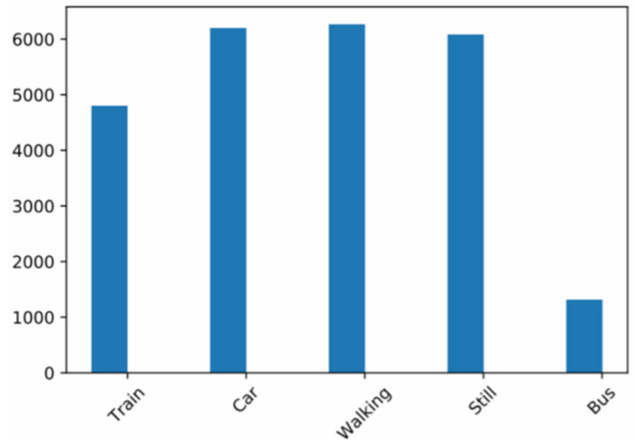


Figure 1. Samples collected by each transportation mode

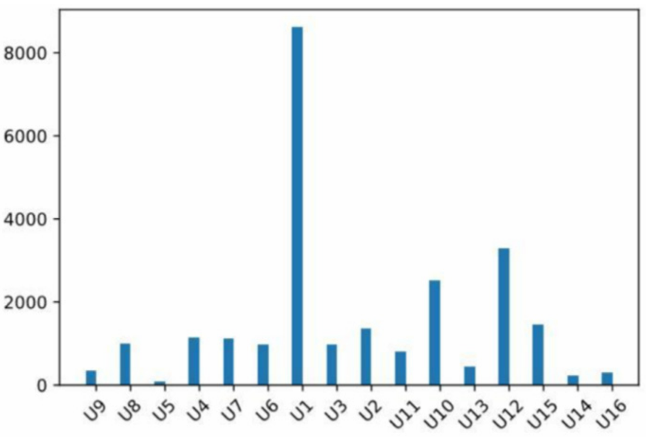


Figure 2. Samples collected by each user

We have used only nine sensors (Table 1) as per their expected availability in every user's smartphones [18].

Table 1. Considered sensors-set

Sensors	Accelerometer Sensor Orientation Sensor Linear-Acceleration Sensor Uncalibrated Gyroscope Sensor Gyroscope Sensor Game-Rotation Vector Sensor Rotation-Vector Sensor Sound Sensor Speed Sensor

4.3 Discussion

We use an accuracy metric to evaluate the performance of each classifier. It can be defined as equation (2).

$$Accuracy = \frac{TP + TN}{TI} \quad \textbf{(2)}$$

Here, TP represents True Positives, TN represents True Negatives, and TI represents Total Interference.

The accuracy provides a fraction of the optimum classifications. However, it cannot evaluate the correct insights about the sensitivity of the classifier. Hence, Precision, Recall, and F-Score metrics are crucial to calculating [19-20]. (3, 4, 5) shows the Precision, Recall, and f-score metrics.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

Here, TP represents True Positives, FP represents False Positives, and FN represents False Negatives.

5 Proposed Methodology

We propose a method to implement the dimensionality reduction technique for evaluating the effect on classification accuracy. Besides, we describe the proposed model (Figure 3). For Feature Engineering, we sample the data from the Smartphone's built-in sensors and the timestamps according to the transportation modes for performing data cleaning and window partitioning tasks, as described in section 4. Afterward, we extract the features from the time window for dimensionality reduction using Principal Component Analysis (PCA) [21].

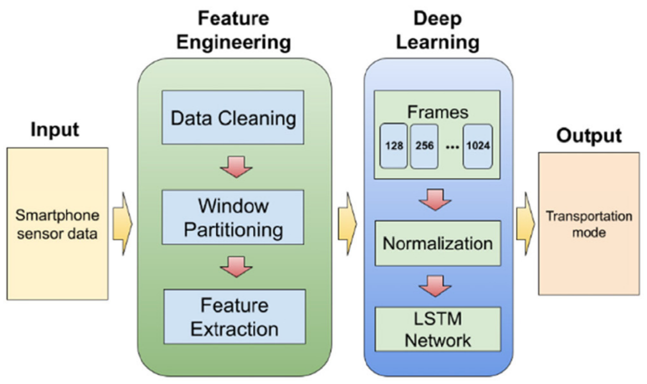


Figure 3. Our proposed real-time TMD model

5.1 Feature Engineering

Feature Engineering is the technique that addresses the challenges we face during extracting relevant features from the raw dataset for training a regression or a classification model using traditional ML algorithms. This technique requires domain knowledge because it involves the data cleaning process that includes transformation, construction, combination, and selecting appropriate features from extracted features [22].

An alternative of an LSTM network with additional Feature Engineering is the *ConvLSTM network* [23-24]. Since most of the data readings in the collected raw TMDataset[1] are based on the orientation of a 3-dimensional coordinate system, there are still some other sensors like Ambiental sensors and proximity sensors that return only a single data value. Hence, data cleaning and time window partitioning become necessary for the sake of the proposed model's performance efficiency [25]. This additional Feature Engineering technique is more beneficial for the overall model performance than the proposed deep learning architecture. There is another technique called Principal Component Analysis (PCA), which transforms the large variable-set into the small-variable set without losing some of the most valuable information. In this stage, we traditionally extracted various features like mean, standard deviation, maximum, and minimum from all the sensor's 5 second time window and applied PCA for reducing the dimensionality of the data. Hence, the additionally applied Feature Engineering phase makes our model less complex and more performance efficient. A basic self-experiment strategy is applied for selecting different components consisting of choosing the first 's' components. For example, initially, there were 12 features extracted using three sensors, namely Accelerometer, sound, and Gyroscope, in the smallest sensor setting. From those 12 features, we have selected the first three components using PCA, reducing the number of predictor variables by 75%, which reduces the time and computational costs related to model fitting. While performing experiments for performance evaluation, we compare classification cost and accuracy and the sensors' extracted features to evaluate the trade-off for online transportation mode detection.

5.2 Normalization

Every extracted feature's value lies in different ranges. For example, one sensor's mean goes for thousands, while that sensor's standard deviation often stays close to zero. For scaling each feature's value, normalization has been applied to all the features in the range of 0 to 1. Normalization is represented by (6).

$$x = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{6}$$

5.3 Long Short-Term Memory (LSTM) Network

Training the model is the last step of our proposed method. The Recurrent Neural Network is almost a traditional Deep Learning's Deep Neural Network architecture. However, it consists of additional weight for the hidden layer's every cell, termed as a Hidden State. In this, past activities store to use them with a new set of inputs on the future iterations. The Bi-directional RNNs are known for using data from the present and the past. Hence, it is the best model for feedbacked sequential data. Here, window overlapping has been avoided because RNN remembers the previous inputs and the patterns.

Considering the input sequence as x, such as x_1, x_2, x_3, ..., x_n, the hidden state h_t can be represented by (7).

$$h_t = \begin{cases} 0, & t = 0 \\ \Phi(h_{t-1}, x_t), & otherwise \end{cases} \quad \textbf{(7)}$$

Where 'Φ' is an Activation Function, we can update the hidden state of RNN using (8) [26].

$$h_t = g(wx_t + Uh_{t-1}) \quad \textbf{(8)}$$

Nevertheless, RNN is difficult for training for the long-term sequences due to a gradient-based optimization algorithm. Long sequences are impossible to train sufficiently because of the vanishing gradient problem, i.e., the change ratio to the weights that reduce slowly concerning the time. Thus, initial inputs are getting worthless because of future information. Therefore, Long Short-Term Memory (LSTM) was developed by Sepp Hochreiter and Jurgen Schmidhuber in 1997 [27]. LSTM can remember the data for a long time because it consists of a cell or a memory capable of reading, writing, deleting, and updating it. We can see the structure of LSTM in Figure 4.

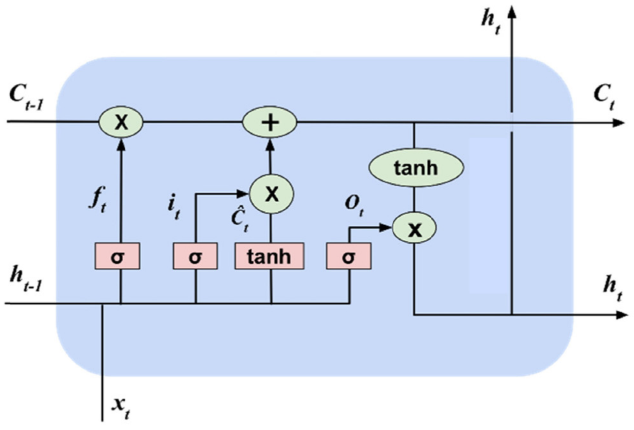


Figure 4. A basic LSTM cell structure

C_t represents the line following signal, the hidden state is represented by h_t, and the input vector is represented by x_t. f_t is the output that can be found after the input and past hidden state enter the forgetting gate. It can be defined as (9).

$$f_t = \delta(w_f(h_{(t-1)}, x_t) + b_f) \quad \textbf{(9)}$$

The following two steps are for selecting the inputs as described in (10).

$$i_t = \delta(w_i(h_{(t-1)}, x_t) + b_i)$$
$$\hat{C} = \tanh(w_c(h_{(t-1)}, x_t) + b_c) \quad \textbf{(10)}$$

Now the LSTM will update C_t using these two gate equations in (11).

$$C_t = ((f_t \times C_{(t-1)}) + (i_t \times \hat{C}_t)) \quad \textbf{(11)}$$

Eventually, these changes are applied to h_t in (12), after updating the hidden state.

$$o_t = \delta(w_o(h_{(t-1)}, x_t)) + b_o$$
$$h_t = o_t \times \tanh(C_t) \quad \textbf{(12)}$$

In our method, LSTM is used because it is very efficient for different sequence modeling. We have used only one layer yet multiple cells of LSTM in our study. There are 256 features in each frame, and these features are the input-set of the LSTM. Say there are 'x' features in a window, then the input-set is being iterated 'x' times throughout the LSTM network.

After features (F) are extracted, we arrange them in various frames $(\{F_t\})$ We are shown in (13) and (14).

$$\{F_t\} = \frac{F_n - F_{\min}}{F_{\max} - F_{\min}} \quad \textbf{(13)}$$

$$(H_t^m, C_t^m) = LSTM_m((F_t)_t^m, C_{t-1}^m) \quad \textbf{(14)}$$

Here, m, t = 1, 2, …., n.

The most commonly used number of LSTM cells, i.e., 128, 256, 512, and 1024 in the literature, are being tested. Overall, we used *Adam* optimizer with a dropout of 0.1 to maintain the rate of 73% and obtained prediction and loss using *Categorical Cross-Entropy* and *Softmax* activation function as there are five modes of transportation in our dataset. We can see our proposed LSTM model in Figure 5.

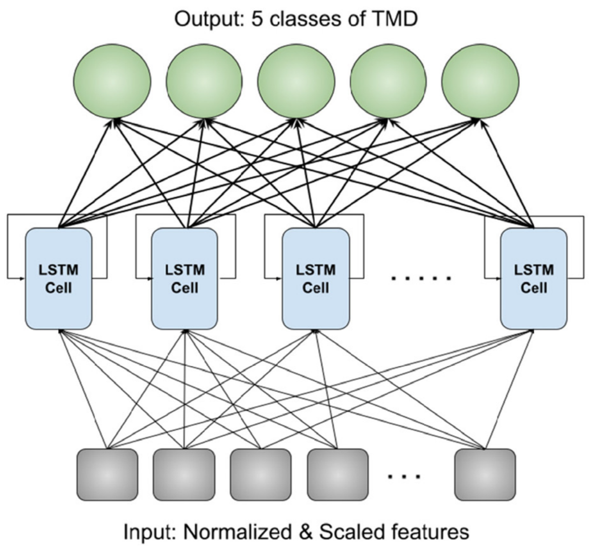


Figure 5. Structure of LSTM network for the proposed model

5.4 Hyperparameter Tuning

Hyperparameters are the neural network model's different parameters that play a crucial role in model performance. Sometimes, choosing optimum hyperparameters manually one-by-one becomes difficult due to the complexity of the model. There were some hyperparameter tuning techniques introduced recently. Using these hyperparameter tuning techniques, we can improve the computation cost [28]. It is the process of choosing the best hyperparameters to maintain the efficiency of model performance. Essential hyperparameter tuning techniques are given as Grid search, Bayesian optimization, and Cross-validation.

5.4.1 Grid Search

Grid search is one of the best hyperparameter tuning techniques. Although it can be computationally costly, it increases the model performance metrics such as accuracy, Precision, Recall, and F1-value while being non-adaptive [9]. We used *GridSearchCV* from *scikit-learn* in our proposed model.

5.4.2 Bayesian Optimization

Bayesian optimization is the most advanced hyperparameter tuning technique [29]. It chooses the optimum hyperparameters based on previous performances of the model automatically. Due to its dependency on the Gaussian process, it can be computationally faster. However, it might sometimes perform poorly.

5.4.3 Cross-validation

Cross-validation is the commonly used technique to compute bias-variance trade-offs. It is referred to as *k-fold cross validation* because we divide the dataset in 'k' folds; each fold has a 'k-1' training division and 1 test division [30]. It does not provide the exact best hyperparameters, but it efficiently estimates the model performance through a bias-variance trade-off [31]. However, Cross-validation is practically very hard to perform with our hardware. Thus, as we are working with a large dataset, Cross-Validation can make the process computationally costlier [32]. Hence, we prefer to use the Holdout method in our proposed model.

5.5 Experimental Setup

All our paper experiments are conducted on a high-performing laptop with the specifications of Intel Core i7-9750H, 2.40 GHz processor, 16GB RAM, Nvidia GeForce GTX 1080Ti GPU on Windows 10 workstation. For performing the experiments and observing the results, Anaconda's Spyder IDE [33] is used. The data preparation, i.e., Feature Engineering, is performed using Pandas and NumPy computational libraries and scikit-learn scientific library [34]. For implementing our proposed LSTM network, a high-level Neural Network API, i.e., the Keras library and the TensorFlow library for backend processing, is used [35].

6 Empirical Study

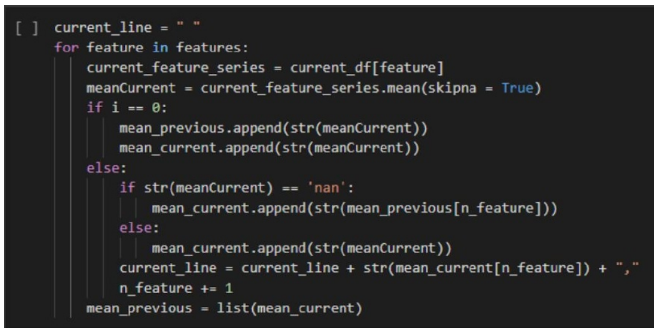
Our proposed algorithm for transportation mode detection using Feature Engineering and LSTM network (deep learning technique) is mentioned in Table 2.

Table 2. Proposed algorithm

```
Algorithm: LSTM network for Transportation Mode Detection
1.  initialize: m framed data ← sensors
2.  for data in a range (1, len(m)):
3.     [Data] ← cleansing (data)
4.     For window in data:
5.        [Windows] ← partitioning (window)
6.        For a feature in windows
7.           [Fv] ← n frames (k sized features)
8.  [fv] ← (timestamp × k features (normalization ([fv])))
9.  lstm ← LSTM network
10. [traning_set].[validation_set] ← train_test_split ([fv]))
11. lstm ← Dropout (lstm)
12. lstm ← lstm (softmax, categorical_cross_entropy)
13. for q_epochstrain (fit ([training_set], lstm)):
14.    loss_function, lstm_updated ← GridSearchCV (lstm)
15. performance_metrics ← lstm_updated [validation_set]
16. End
```

First of all, we sample data with 'm' sized frames. We apply Feature Engineering, including data cleaning, window partitioning, and selecting appropriate features from the data. For example, as mentioned in the sample Python code snippet: 1, we divide the Bus transportation mode into different sample time windows. We have extracted its mean, minimum, maximum, and standard deviation features for each time window.

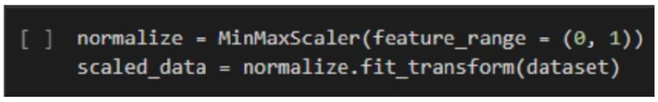
Snippet 1: Feature Extraction



```
[ ] current_line = " "
    for feature in features:
        current_feature_series = current_df[feature]
        meanCurrent = current_feature_series.mean(skipna = True)
        if i == 0:
            mean_previous.append(str(meanCurrent))
            mean_current.append(str(meanCurrent))
        else:
            if str(meanCurrent) == 'nan':
                mean_current.append(str(mean_previous[n_feature]))
            else:
                mean_current.append(str(meanCurrent))
            current_line = current_line + str(mean_current[n_feature]) + ","
            n_feature += 1
        mean_previous = list(mean_current)
```

We create frames of vectors and store 256 features, including some new attributes, for example, turn frequency, stationary duration, and signal strength of the base station in each frame. Then for normalization, we use *MinMaxScaler* from *scikit-learn's preprocessing* to normalize the data in the range of 0 to 1, as mentioned in Python code Snippet: 2.

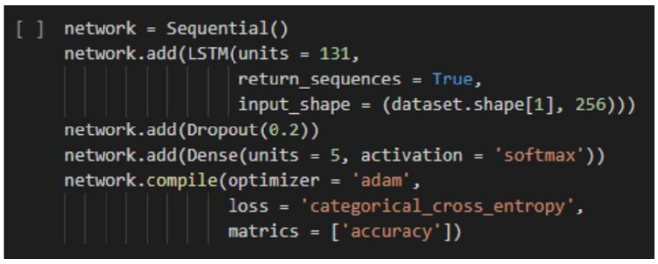
Snippet 2: Normalization



```
[ ] normalize = MinMaxScaler(feature_range = (0, 1))
    scaled_data = normalize.fit_transform(dataset)
```

We set a time step to reshape the data by multiplying the normalized data with its time step. As discussed earlier, the scaled data goes into the LSTM network as its input, but we input different sized samples. Then, we perform forward propagation with some dropout to calculate training performance and loss using *Softmax* activation function and *Categorical cross-entropy*. We use *Keras's Sequential*, *Dense*, *LSTM,* and *Dropout* procedures to create an LSTM network. Further, we compile the system using *Adam* optimizer and adjust the weight according to gathered loss and training performance after the forward propagation for each set of the input sample, as shown in Python code snippet: 3.

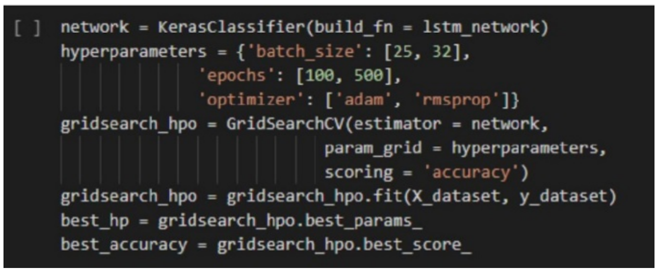
Snippet 3: The LSTM network



```
[ ] network = Sequential()
    network.add(LSTM(units = 131,
                     return_sequences = True,
                     input_shape = (dataset.shape[1], 256)))
    network.add(Dropout(0.2))
    network.add(Dense(units = 5, activation = 'softmax'))
    network.compile(optimizer = 'adam',
                    loss = 'categorical_cross_entropy',
                    matrics = ['accuracy'])
```

In the end, we apply *scikit-learn's GridSearchCV* hyperparameter optimization technique to get optimal parameters to the convergence of the loss function. In sample Python code snippet: 4, we perform hyperparameter optimization using *GridSearchCV*. *KerasClassifier* function is used from Keras's *wrapper* class to join the hyperparameter tuning operation to the LSTM network.

Snippet 4: Hyperparameter tuning



```
[ ] network = KerasClassifier(build_fn = lstm_network)
    hyperparameters = {'batch_size': [25, 32],
                       'epochs': [100, 500],
                       'optimizer': ['adam', 'rmsprop']}
    gridsearch_hpo = GridSearchCV(estimator = network,
                                  param_grid = hyperparameters,
                                  scoring = 'accuracy')
    gridsearch_hpo = gridsearch_hpo.fit(X_dataset, y_dataset)
    best_hp = gridsearch_hpo.best_params_
    best_accuracy = gridsearch_hpo.best_score_
```

7 Experimental Results

We have explained our model in the previous section, where we have proposed an end-to-end process of transportation mode detection, where our model performed up to 98%. This section will focus on the results and observations of the proposed model based on various parameters.

7.1 Frame Dimension (FD)

Selecting the dimension of the frame is a crucial factor that can easily affect model performance. Frame dimensions can slightly disturb frame count, increasing or decreasing the input of the LSTM network. The window dimension used here is 720, and the size of every feature is 72. That gives us ten feature frames (LSTM input) for each window. In this way, we focused on the frame dimension because if the frame increases to the optimum size, it provides us with more features and the highest possible accuracy.

7.2 Window Dimension (WD)

The window dimension can have an impact on the model's performance rate. More oversized windows can contain the highest number of feature frames. Nevertheless, it can increase the performance time of the model as well. In the proposed method, we used 360 and 720 sample dimensions of windows to observe the model's accuracy.

7.3 LSTM Cell Dimension (LCD)

As discussed, we have run several experiments on the different dimensions of LSTM cells. However, we have used only one layer of LSTM cells in the proposed method. More LSTM cells could give better accuracy, but it could also increase the whole network's complexity. 128, 256, 512, and 1024 are the optimum dimensions of the LSTM cells, as used in the literature.

7.4 Training Data after the Split (TDS)

We split our dataset into training dataset and validation dataset before we train our classification model. The training data covers 70%-80% of the dataset, which allows the model to train and validate the performance using the remaining 20%-30% of the validation dataset. The training data takes most of the data because it helps the model learn more from the training data. However, if the training data is not optimum, it can lead the model to either overfitting or underfitting. In the proposed method, we split the training data into 75%.

7.5 Count of Epoch (CE)

One-time forward propagation and backward propagation combined is called an Epoch during the model's training phase. An optimum number of epochs can make the model learn more efficiently. We have experimented with several Epoch counts and found the suitable count of the epoch for our model.

Here, we use the notation *FD-WD-LCD-TDS-CE* to represent the result. Our experiment used different frames and window dimensions with constant LSTM cell dimensions, train data split, and epoch count. For example, we have used 36 sample frame dimensions and 360 sample window dimensions. LSTM cell of 256 measurements with a train data split rate is 75% used, requiring a 20-optimum number of epochs. These parameters give us an accuracy of 94.17%, mentioned in Table 3. The following parameters can be denoted as 36-360-256-75-20.

Table 3. Frames and windows dimensions comparison

FD	WD	LCD	TDS	EC	Acc (%)
18	360	256	75	20	94.46
36	**360**	256	75	20	**96.17**
36	720	256	75	20	95.43
72	720	256	75	20	95.54

Here: FD = Frame Dimension, WD = Window Dimension, LCD = LSTM Cell Dimension, TDS = Train Data Split, CE = Count of Epoch, Acc = Accuracy

In Table 4, we experiment with different LSTM cell dimensions, and we let the other parameters be constant. Table 3, 36-360-256-75-20, gives an accuracy of 96.17%, the highest among all the additional accuracy present in the table.

Table 4. Different LSTM cell dimension comparison

FD	WD	LCD	TDS	CE	Acc (%)
36	360	128	75	20	95.68
36	360	**256**	75	20	**96.17**
36	360	512	75	20	95.78
36	360	1024	75	20	96.10

Here: FD = Frame Dimension, WD = Window Dimension, LCD = LSTM Cell Dimension, TDS = Train Data Split, CE = Count of Epoch, Acc = Accuracy

In Table 5, we experiment with different train data split rates while the remaining parameters are constant. Surprisingly, the parameters in Table 5 that resulted in the highest accuracy and the highest accuracy are the same as the parameters in Table 3 and Table 4, which resulted in their highest accuracy and accuracy.

Table 5. Different training data split rate comparison

FD	WD	LCD	TDS	CE	Acc (%)
36	360	256	65	20	94.87
36	360	256	70	20	94.83
36	360	256	**75**	20	**96.17**
36	360	256	80	20	95.83

Here: FD = Frame Dimension, WD = Window Dimension, LCD = LSTM Cell Dimension, TDS = Train Data Split, CE = Count of Epoch, Acc = Accuracy

Therefore, in Table 3, Table 4, and Table 5, we observe that only 36-360-256-75 facilitates the highest accuracy among all the other Accuracies. After comparing all three tables, we find that these tables' highest Accuracies are the same and result in the same parameters. Hence, we can conclude that 36 frame dimensions, 360 window dimensions, 75% train data split rate, and 256 LSTM cell dimensions are the optimum parameters. Now, as we have found all the optimum parameters in the above observations, however, the epoch's count remained constant in all the experiments that we have performed. Count of the epoch plays a vital role in deep learning. Epoch can lead a model to the underfitting or the overfitting state as more epochs result in more training with the data and fewer epochs result in less training with the data. Hence, we will be careful while experimenting with different epoch counts and focusing on the optimum number of epoch counts. As shown in Table 6, we have tested with varying counts of the epoch. Initially, when we increase the count from 10 to 20, the accuracy improves by 2.21%. Nevertheless, the accuracy's increase rate is 0.04% between epoch count of 40 and epoch count of 50.

Table 6. Different count of the epoch comparison

FD	WD	LCD	TDS	CE	Acc (%)
36	360	256	75	10	94.00
36	360	256	75	20	96.21
36	360	256	75	40	97.68
36	360	256	75	**50**	**98.11**

Here: FD = Frame Dimension, WD = Window Dimension, LCD = LSTM Cell Dimension, TDS = Train Data Split, CE = Count of Epoch, Acc = Accuracy

Figure 6 demonstrates the loss convergence graph; we can observe that the graph convergence rate is in saturation point from 40 to 50 epochs. Hence, 50 epochs result in 98.11% accuracy, which is the highest optimum accuracy facilitated by 36-360-256-75-50 parameters.

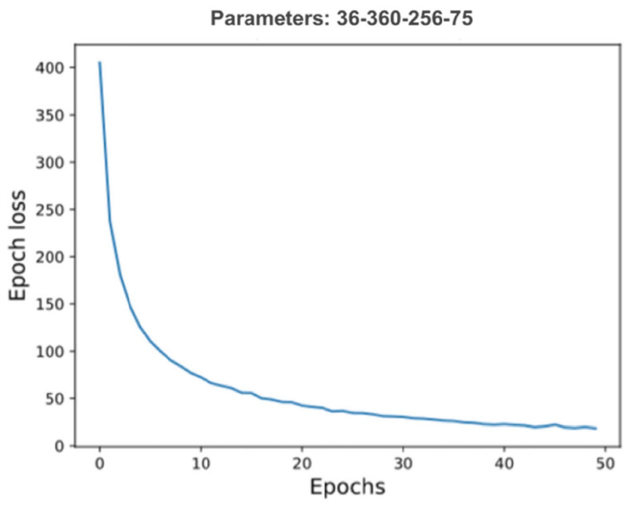


Figure 6. Drop rate of the loss concerning epoch count

Observing the above experiments, 36-360-256-75-50 are suitable parameters. Using these parameters, data collection for Feature Engineering consumes 5 seconds. However, Feature Engineering and deep learning processes consume 90 milliseconds and 722 µ-seconds.

Table 7 demonstrates the confusion matrix for different class labels of our proposed classification model. In our confusion matrix, we observe that the predicted walk class performed almost entirely. The car and the bus classes confuse me the most. Overall, using our confusion matrix, we have calculated Precision and Recall for each class label.

Table 7. Confusion matrix of our proposed model

Pred. → True ↓	Still	Car	Bus	Walk	Train	Recall (%)
Still	639	0	1	13	4	97.26
Car	2	3070	34	5	18	98.11
Bus	0	27	1180	7	9	96.48
Walk	9	4	4	2474	6	99.07
Train	0	9	6	10	1387	98.22
Precision (%)	98.30	98.71	96.32	98.60	97.40	

The accuracy of the model describes the performance of the model. Similarly, Recall and Precision demonstrate the model's overall relevance to the results [36]. We calculate F-Score, the calculated Precision, and Recall average to represent the perfect Precision and Recall for each class. Table 8 shows the average Precision, Recall, and F-Score.

Table 8. Average Recall, Precision, and F1-Score calculation

Parameter	Calculation (%)
Recall	97.82
Precision	97.86
F-Score	97.83

8 Model Performance

This section will compare our model's performance with the other different models from the study done by Broach et al. [6], Wang et al. [7], Xiao et al. [8], Soares et al. [9] and Soares et al. [12]. Table 9 contains a comparison of the observations of various performance parameters of different models. We have compared Accuracy, Recall, Precision, and F-Score [37] of Multinomial Logit Model, Random Forest, Continuous Hidden Markov Model, Travel Mode Detection Ensemble, and our model Travel Mode Detection with Deep Learning. We observe that the Multinomial Logit Model's performance accuracy is lowest, i.e., 91%, among all the other models, but the overall performance is good in Precision, Recall, and f-score, i.e., 94% in all. Random Forest gives an accuracy of 93%, but the rest of the parameters could not achieve more than 90%. The continuous Hidden Markov Model surprisingly achieves many efficient performance parameters. However, Travel Mode Detection Ensemble (TMDE) performed with 97% Accuracy, Recall, Precision, and F-Score, which is the best among all the above-discussed models. However, the study was done by Soares et al. [12] used Feed-Forward Neural Network and Long Short-Term Memory (TMD-LSTM). Since this study uses a deep learning approach, there can be structural similarities between this study and our study. However, the authors in TMD-LSTM collected the data using only orientation sensors, whereas, in our study, we collected the data using orientation sensors and ambient and proximity sensors. Apart from data collection, we extracted features from only the time domain and fed them in our LSTM network after framing. In TMD-LSTM, authors extracted features from both time and frequency-domain and fed them in FNN and LSTM. However, the performance of TMD-LSTM is slightly efficient with 90% in Accuracy, Precision, Recall, and F-Score parameters compared to the study with Random Forest. Our proposed Transportation Mode detection's performance parameters with LSTM Network resulted slightly well with a 1% increment than the TMDE in all the performance parameters. Our model scored 98% of Accuracy, Precision, Recall, and F-Score. Hence, our proposed model performed more effectively among all the other compared models.

Table 9. Comparison of different models' performance with our model performance

Models	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)
MNL (Broach et al.) [6]	91	94	94	94
RF (Wang et al.) [7]	93	89	88	88
CHMM (Xiao et al.) [8]	94	91	90	90
TMDE (Soares et al.) [9]	97	97	97	97
TMD-LSTM (Soares et al.) [12]	90	90	90	90
TMD-LSTMN (Our)	98	98	98	98

Here: MNL = Multinomial Logit Model, RF = Random Forest,
TMDE = Travel Mode Detection Ensemble, CHMM = Continuous Hidden Markov Model, TMD-LSTM = Travel Mode Detection with FNN and LSTM, TMD-LSTMN = Travel Mode Detection with LSTM Network.

8.1 Impact of Transportation Mode Detection Technique on Intelligent Cities

Transportation mode detection has a substantial impact on intelligent transportation [38-39] and smart cities. Depending on the user location and situation, it can work as the best recommender system for choosing the transportation modes [40]. Unlike other systems dependent mainly on centralized servers, the intelligent transportation mode detection technique focuses on each user's Smartphone separately. Hence, it can reduce complexity and cost compared to the traditional transportation mode recommender systems [41-42]. The governments can also use this system smartly by collecting the users' transportation history data and can provide them with the best recommendation or facilitate the required transportation modes in particular areas.

Similarly, on-demand cab service providers and traditional taxi services can also use this system to serve their customers smartly [31, 21]. These are the fundamental advancements that can take place using intelligent transportation mode detection. The most crucial area where it can benefit is that many businesses use this system for the transportation modes' recommendation systems and create the user profile [1] and mobility as the right platforms [43] to provide efficient facilities future. Hence, it can be optimized in smart transportation mode recommendations and intelligent pricing [44].

9 Conclusion

We have introduced a novel deep learning approach to the transportation mode detection model and studied the impact of feature engineering on transportation modes' classification performance. We have collected the user transportation mode dataset using the technique used in TMDataset[1] using nine mentioned sensors from the smartphone application. We proposed a method that performed the success rate of up to 98% while the other compared models could achieve only up to 97% of the success rate. Moreover, we believe that the Epoch count and frame size are two factors in our study: the number of feature frames that positively affect the performance. However, the limitation of the model's high computational cost, learning rate, drop rate, and error detection could improve in new future transportation mode detection studies.

References

[1] A. C. Prelipcean, G. Gidófalvi, Y. O. Susilo, Transportation mode detection – an in-depth review of applicability and reliability, *Transport Reviews*, Vol. 37, No. 4, pp. 442-464, 2017.

[2] D. Shin, D. Aliaga, B. Tunçer, S. M. Arisona, S. Kim, D. Zünd, G. Schmitt, Urban sensing: Using smartphones for transportation mode classification. *Computers, Environment & Urban Systems*, Vol. 53, pp. 76-86, September, 2015.

[3] C. Zhou, H. Jia, J. Gao, L. Yang, Y. Feng, G. Tian, Travel mode detection method based on big smartphone global positioning system tracking data, *Advances in Mechanical Engineering*, Vol. 9, No. 6, pp. 1-10, June, 2017.

[4] L. Wu, B. Yang, P. Jing, Travel mode detection based on GPS raw data collected by smartphones: a systematic review of the existing methodologies, *Information*, Vol. 7, No. 4, Article No. 67, December, 2016.

[5] S. Kaplan, S. Bekhor, Y. Shiftan, Web-based survey design for unravelling semi-compensatory choice in transport and urban planning, *Transportation Planning and Technology*, Vol. 35, No. 2, pp. 121-143, February, 2012.

[6] J. Broach, J. Dill, N. W. McNeil, Travel mode imputation using GPS and accelerometer data from a multi-day travel survey, *Journal of Transport Geography*, Vol. 78, pp. 194-204, June, 2019.

[7] B. Wang, L. Gao, Z. Juan, Travel Mode Detection Using GPS Data and Socioeconomic Attributes Based on a Random Forest Classifier, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 19, No. 5, pp. 1547-1558, May, 2018.

[8] G. Xiao, Q. Cheng, C. Zhang, Detecting travel modes from smartphone-based travel surveys with continuous hidden Markov models, *International Journal of Distributed Sensor Networks*, Vol. 15, No. 4, pp. 1-15, April, 2019.

[9] E. F. Soares, C. A. Campos, S. C. Lucena, Online travel mode detection method using automated machine learning and feature engineering, *Future Generation Computer Systems*, Vol. 101, pp. 1201-1212, December, 2019.

[10] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, F. Hutter, Auto-sklearn: Efficient and Robust Automated Machine Learning, in: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), *Automated Machine Learning*, Springer, Cham, 2019, pp. 113-134.

[11] M. M. Salvador, M. Budka, B. Gabrys, Automatic Composition and Optimization of Multicomponent Predictive Systems With an Extended Auto-WEKA, *IEEE Transactions on Automation Science and Engineering,* Vol. 16, No. 2, pp. 946-959, April, 2019.

[12] E. F. d. S. Soares, H. Salehinejad, C. A. V. Campos, S. Valaee, Recurrent Neural Networks for Online Travel Mode Detection, *2019 IEEE Global Communications Conference*, Waikoloa, HI, USA, 2019, pp. 1-6.

[13] Q. Wu, K. Ding, B. Huang, Approach for fault prognosis using recurrent neural network, *Journal of Intelligent Manufacturing*, Vol. 31, No. 7, pp. 1621-1633, October, 2020.

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, *Communications of the ACM*, Vol. 60, No. 6, pp. 84-90, June, 2017.

[15] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, *IEEE Transactions on Neural Networks and Learning Systems*, Vol.

28, No. 10, pp. 2222-2232, October, 2017.

[16] O. Banos, J-M. Galvez, M. Damas, H. Pomares, I. Rojas, Window Size Impact in Human Activity Recognition, *Sensors*, Vol. 14, No. 4, pp. 6474-6499, April, 2014.

[17] P. Domingos, A few useful things to know about machine learning, *Communications of the Association for Computing Machinery*, Vol. 55, No. 10, pp. 78-87, October, 2012.

[18] D. Lu, D. Nguyen, T. Nguyen, H. Nguyen, Vehicle Mode and Driving Activity Detection Based on Analyzing Sensor Data of Smartphones, *Sensors*, Vol. 18, No. 4, Article No. 1036, April, 2018.

[19] J. Cook, V. Ramadas, When to consult precision-recall curves, *The Stata Journal*, Vol. 20, No. 1, pp. 131-148, March, 2020.

[20] T. Fawcett, An introduction to ROC analysis, *Pattern Recognition Letters*, Vol. 27, No. 8, pp. 861-874, June, 2006.

[21] A. Anand, M. A. Haque, J. Alex, N. Venkatesan, Evaluation of Machine learning and Deep learning algorithms combined with dimentionality reduction techniques for classification of Parkinson's Disease, *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Louisville, Kentucky, USA, 2018, pp. 342-347.

[22] H. Liu, H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, *The Springer International Series in Engineering and Computer Science*, Springer, 1998.

[23] A. Nawaz, H. Zhiqiu, W. Senzhang, Y. Hussain, I. Khan, Z. Khan, Convolutional LSTM based transportation mode learning from raw GPS trajectories, *IET Intelligent Transport Systems*, Vol. 14, No. 6, pp. 570-577, June, 2020.

[24] S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, M. Akbar, Fake news detection using deep learning models: A novel approach, *Transactions on Emerging Telecommunications Technologies*, Vol. 31, No. 2, Article No. e3767, February, 2020.

[25] Y. Zhong, S. Fong, S. Hu, R. Wong, W. Lin, A Novel Sensor Data Pre-Processing Methodology for the Internet of Things Using Anomaly Detection and Transfer-By-Subspace-Similarity Transformation, *Sensors*, Vol. 19, No. 20, Article No. 4536, October, 2019.

[26] S. Sivakumar, S. Sivakumar, Modulation of Activation Function in Triangular Recurrent Neural Networks for Time Series Modeling, *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019, pp. 1-8.

[27] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735-1780, November, 1997.

[28] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization, *Journal of Machine Learning Research*, Vol. 18, pp. 1-52, 2018.

[29] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. Collins, J, Schneider, B. Poczos, E. Xing, Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly, *Journal of Machine Learning Research*, Vol. 21, pp. 1-27, 2020.

[30] G. Dupret, M. Koda, Bootstrap re-sampling for unbalanced data in supervised learning, *European Journal of Operational Research*, Vol. 134, No. 1, pp. 141-156, October, 2001.

[31] M. Belkin, D. Hsu, D. Ma, S. Mandal, Reconciling modern machine-learning practice and the classical bias-variance trade-off, *Proceedings of the National Academy of Sciences*, Vol. 116, No. 32, pp. 15849-15854, July, 2019.

[32] J. Awwalu, F. Ogwueleka, On Holdout and Cross Validation: A Comparison between Neural Network and Support Vector Machine, *International Journal of Trend in Research and Development*, Vol. 6, No. 2, pp. 235-239, March-April, 2019.

[33] R. Patra, B. Khuntia, Predictive Analysis of Rapid Spread of Heart Disease with Data Mining, *2019 IEEE International Conference on Electrical, Computer and Communication Technologies*, Coimbatore, India, 2019, pp. 1-4.

[34] F. Nelli, *Python Data Analytics*, Apress, 2015.

[35] K. Ramasubramanian, A. Singh, *Machine Learning Using R*, Apress, 2019.

[36] F. Tortorella, Preface: Special issue on "ROC Analysis in Pattern Recognition", *Pattern Recognition Letters*, Vol. 27, No. 8, pp. 859-860, June, 2006.

[37] L. A. Jeni, J. Cohn, F. Torre, Facing Imbalanced Data--Recommendations for the Use of Performance Metrics, *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII)*, Geneva, Switzerland, 2013, pp. 245-251.

[38] K. Wang, C. Chen, M. S. Hossain, G. Muhammad, S. Kumar, S. Kumari, Transfer reinforcement learning-based road object detection in next generation IoT domain, *Computer Networks*, Vol. 193, Article No. 108078, July, 2021.

[39] J. Zhang, D. He, N. Kumar, K. R. Choo, An Efficient and Secure Authentication Scheme for Vehicle Sensor Networks, *Journal of Internet Technology*, Vol. 20, No. 2, pp. 617-627, March, 2019.

[40] S. Porru, F. E. Misso, F. E. Pani, C. Repetto, Smart mobility and public transport: Opportunities and challenges in rural and urban areas, *Journal of Traffic and Transportation Engineering (English Edition)*, Vol. 7, No. 1, pp. 88-97, February, 2020.

[41] L. Sarv, K. Kibus, R. M. Soe, Smart city collaboration model: a case study of university-city collaboration, *13th International Conference on Theory and Practice of Electronic Governance (ICEGOV 2020)*, Athens, Greece, 2020, pp. 674-677.

[42] V. Karachay, D. Prokudin, O. Kononova, D. Pilyasova, Sociocultural information urban space in smart city context, *13th International conference on Theory and practice of Electronic Governance (ICEGOV 2020)*, Athens, Greece, 2020, pp. 568-575.

[43] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, A personalised travel recommender system utilising social network profile and accurate GPS data, *Electronic Government, an International Journal (EG)*, Vol. 14, No. 1, pp. 90-113, January, 2018.

[44] C. Hsieh, J. Chen, C. Kuo, P. Wang, End-to-End Deep Learning-Based Human Activity Recognition Using Channel State Information, *Journal of Internet Technology*, Vol. 22, No. 2, pp. 271-281, March, 2021.

Biographies

Sachin Kumar is currently working as a Professor of Computer Science and Engineering at Ajay Kumar Garg Engineering College, Ghaziabad, India. His area of interest includes Computer Networks and Applied Cryptography. He has published more than 30 papers in the SCI/ Scopus Index Journals from IEEE, Elsevier, Springer, John Wiley, etc. Some of his research findings are published/ accepted in top-cited journals such as IEEE Transactions on Industrial Informatics, Computer Networks of Elsevier, Peer to Peer Networking and Applications of Springer, and International Journal of Communication System of Wiley.

Agam Damaraju is a highly motivated technology enthusiast. Currently, he is a final year student of Bachelors of Technology in Computer Science and Engineering at Ajay Kumar Garg Engineering College, Ghaziabad, India. He is a diploma holder in Engineering with a specialization in Electronics. He has hands-on experience in numerous research projects related to Data Engineering, Machine Learning, Deep Learning, Computer Vision, IoT and Automation. His current research interests include Computer Vision, Automation, and Cognitive Computing. He is seeking to explore research applications in the field of Autonomous Systems and Healthcare. He also published a research article on Technological Trends aftermath of Covid-19 pandemic in PHD Chamber Journal of ideas and innovation.

Aditya Kumar is currently a Bachelor of Technology student with Computer Science and Engineering major at Ajay Kumar Garg Engineering College, Ghaziabad, India. His area of interest includes Artificial Intelligence, Internet of Things(IoT), Robotics, Data Analysis, and Automation. He has hands-on experience in programming Embedded Systems, various Machine Learning, and Deep Learning techniques to solve complex problems. He also has hands-on experience on some industry-based Embedded System projects and Industrial Robot Arm programming. He has the certification in Robot Arm programming valid globally.

Saru Kumari is currently an Assistant Professor with the Department of Mathematics, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India. She received her Ph.D. degree in Mathematics in 2012 from Chaudhary Charan Singh University, Meerut, UP, India. She has published more than 215 research papers in reputed International journals and conferences, including more than 195 research papers in various SCI-Indexed Journals. She is on the editorial board of more than a dozen of International Journals, of high repute, under IEEE, Elsevier, Springer, Wiley and others including IEEE Systems Journal. She has served as the Guest Editor of many special issues in many SCI Journals under IEEE, Elsevier, Springer and Wiley. She has been involved in the research community as Technical Program Committee (TPC) member or PC chair for more than a dozen of International conferences of high repute. She is also serving as a reviewer of dozens of reputed Journals including SCI-Indexed of IEEE, Elsevier, Springer, Wiley, Taylor & Francis, etc.

Chien-Ming Chen (Senior Member, IEEE) received the Ph.D. degree from National Tsing Hua University, Taiwan. He is currently an Associate Professor with the Shandong University of Science and Technology, China. His current research interests include network security, the mobile Internet, IoT, and cryptography. He also serves as an Associate Editor for IEEE ACCESS and an Executive Editor for the International Journal of Information Computer Security.