IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Local bus fleet management system using ML

[1]ABDURRAHMAN JAINULABIDEEN, [1] S. JAYA HARISH, [1] SAYAN CHATTERJEE, [1] WALEED, [2] SRIKRISHNAN A

[1]UG Student, [2]Assistant professor,
[1,2]Department of Mechanical Engineering,
[1,2]Dr. M.G.R. Educational and Research Institute, Chennai, India,

Abstract: This local bus transportation management system is in charge of ensuring the perfect solution for the passengers. To accomplish this, organisations must form different teams to handle the many tasks related to the transportation process. Nevertheless, manually completing operations like order segregation, order mapping with an appropriate vehicle, route planning, and others cannot ensure accuracy and speed in the process. In this project, we offer a solution for predicting the number of bus allocations using a machine learning algorithm to solve these issues. ESP32 and IR sensors were used in the module of the hardware kit. Infrared sensors will be used to count bus entry and exit and will analyse the data with the previous data for the allotment of the bus in real time in a frontend.

***Index Terms* – Infrared sensors, Machine Learning algorithm, ESP32, Ejs.**

I. INTRODUCTION

The movement of vehicles in a bus management system is impacted by unpredictable variables as the day goes on, such as traffic congestion, unanticipated delays, random passenger demand, irregular vehicle incidents and dispatching times. Researchers have worked hard to create adaptable control strategies that consider the unique characteristics of public transportation systems in a real-time scenario. Any company's transportation management system oversees ensuring the perfect solution for the passengers. To accomplish this, organisations must form different teams to handle the many tasks related to the transportation process. However, manual completion of activities such as order segregation and order mapping with an appropriate vehicle, route planning, and others cannot ensure accuracy and speed in the process. Yet, these are not the only two difficulties a company must deal with. The scientific discipline of machine learning enables computers to learn without explicit programming. One of the most intriguing technologies that has ever been developed is machine learning. The ability to learn is what, as the name suggests, gives the computer the potential to become more like humans. Today, machine learning is being actively used, possibly in a lot more places than one might think. Computational statistics is a topic that is closely related to statistics and focuses on utilising computers to generate predictions. Machine learning is the method by which computers learn how to do tasks without being explicitly instructed to do so. To learn how to perform specific tasks, computers use the data that is already available. Algorithms that teach the computer how to perform all processes for simple activities can be created. Creating the necessary algorithms by hand for more complex tasks can be challenging for a human. In real-world applications, it may be more efficient to assist the machine in developing its own algorithm than relying on human programmers to specify every step that is necessary. According to the type of machine learning approach, there are generally three basic categories, depending on the nature of the "signal" or "feedback" available to the learning system.

Supervised learning involves a supervisor serving as an instructor. Fundamentally, supervised learning is a type of learning where we instruct or train the computer using data that has been properly labelled, or where some of the data has already been annotated with the right response. For the supervised learning algorithm to analyse the training data (set of training examples) and create a proper result from labelled data, the machine is then given a fresh set of examples (data).

In supervised learning, an algorithm is used to learn the function that maps the input to the output when there are input variables (X) and an output variable (Y).

In this project, google colab is used as an open-source IDE. Google Colab is utilised as an open-source IDE in this project.

Our machine learning and deep learning models can be trained using the free, cloud-based Jupyter notebook environment known as Google Colaboratory on the TPU, GPU, and CPU. This is adequate for the majority of data scientists' computation requirements. Three different runtimes are available for our notebooks through Google Colab,Including CPUs, GPUs, and TPUs.

II. OBJECTIVE

- To develop a web application for the number of bus allocation.
- To develop a user-friendly model for manual bus allotment system.
- To develop a machine learning prediction model depending on the number of passenger count.
- To develop a system to reduce overcrowding of passengers in a local public transport.
- To develop a system to reduce carbon emission of buses by providing accurate number of buses in a specific route.

- To develop a model for smart bus system.

III. Scope of the Project

- It can be used for bus management system like city transportation corporation.
- It can be used by public bus passengers who travel for their daily commutation.

IV. Proposed System

4.1 Existing System

Promoting public transit has become a global census as a main countermeasure to reduce bus congestion and air pollution. Creating a solid and trustworthy bus timetable is a crucial first step to boost ridership and cut cost to the transit system. Unfortunately, the majority of earlier research on bus scheduling relies on static plans created using historical journey times and passenger counts, which frequently produces inaccurate results in these unexpected situations like a demand surge or bad weather. Therefore, it is not practical to obtain real-time passenger origin/destination from a small number of operating buses. To address the concerns, this paper models the multi-line dynamic bus timetable optimisation problem using a Markov Decision Process. It also suggests a multiagent deep reinforcement learning framework to ensure effective learning from the imperfect-information game, where the passenger demand and traffic condition are not always known in advance.

4.2 Disadvantages of Existing System

- In the existing system, the disadvantage is the curse of real-world samples.eg, requires careful maintenance.
- Using can lead to an overload of states, which can diminish the results.
- Maintenance cost is high.

4.3 Proposed System

In this project we provide a solution for predict the number of bus allocation using machine learning algorithm. The hardware kit is developed using ESP32 and IR sensors. The entry and exit of busses will be fixed with IR sensors. Whenever the passenger enters in the bus, the count will be increased. And whenever the passenger exits in the bus, the count will be decreased. The passenger count obtained using the sensors will be transmitted wirelessly using MQTT protocol to the web application developed. It has been decided to build a web application with the help of Ejs and include a button option in it. The control rooms can click on the button whenever they have the desire to perform an analysis on the data. The Linear Regression model that is used for the number of bus allocation will be implemented in the algorithm that will be developed. This algorithm will give the prediction output through an application programming interface (API), and it will be able to be displayed in the frontend. As a result, this project offers an accurate forecast for the distribution of the available bus seats, which can be of great assistance to passengers.

4.4 Advantages of Proposed System

- Cheap and Effective Solution for bus management system.
- Develop web application for predict the number of bus allocation.
- Easy to implement in real life.
- Maintenance of this system is easy.

.

4.5 System Architecture

The architecture diagram and hardware block diagrams are shows as below respectively.

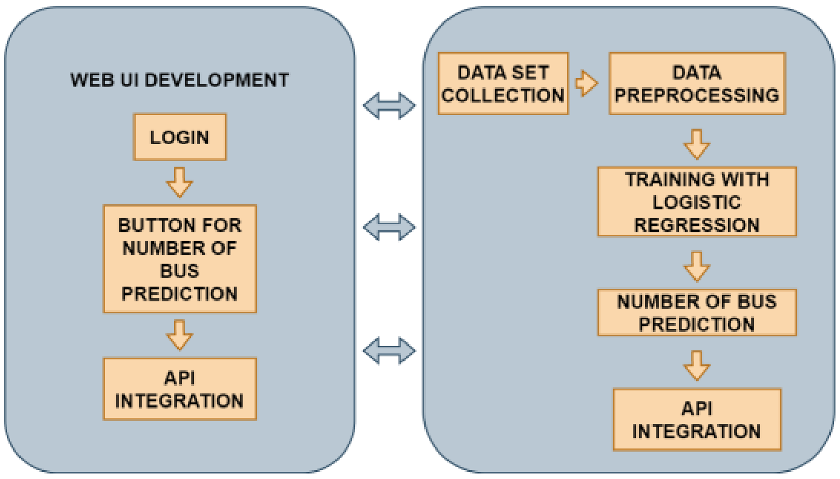


Fig 4.1 Proposed System Architecture.

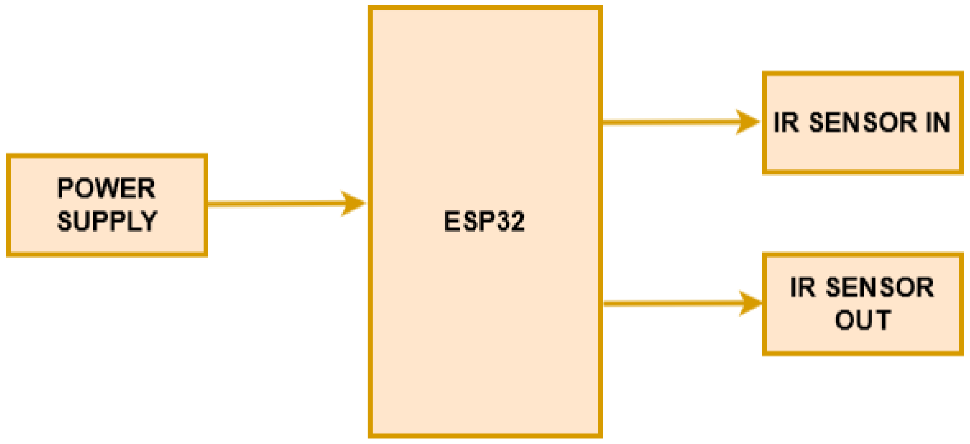


Fig 4.2 Hardware block diagram.

4.6 Working

In this project we provide a solution for predict the number of bus allocation using machine learning algorithm. So, the first step in the project will be collecting the dataset from various resources and then we will be separating these datasets into training as well as testing dataset where the testing dataset will be kept separate, and the training dataset will be used to train the model. Then these datasets are preprocessed to align the datasets into single dimensions. After pre-processing the dataset, we will be ready for training with the architecture. Now, we will be using machine learning algorithm such as Linear Regression is used to train the model. A logistic regression model predicts a dependent data variable by analysing the relationship between one or more existing independent variables. The hardware kit is developed using ESP32 and IR sensors. The entry and exit of busses will be fixed with IR sensors. Whenever the passenger enters in the bus, the count will be increased. And whenever the passenger exits in the bus, the count will be decreased. The passenger count obtained using the sensors will be transmitted wirelessly using MQTT protocol to the web application developed. Whenever the control rooms want to analyse the data, they can click on the button. The logistic regression algorithm gives easy prediction for the number of bus allocation which will give the prediction output via an API, and it can be displayed in the front end. Thus, this project provides an effective prediction for number of bus allocation, and it can be very useful for passengers.

V. SYSTEM ANALYSIS

This section elaborates the proper software module description.

5.1 Software Module Description

- Dataset Collection
- Dataset Pre-processing
- Training with a machine learning algorithm
- Prediction
- Web Application Development

5.2 Hardware Module Description

- Power supply
- Microcontroller ESP32
- IR sensor

VI. SOFTWARE DESCRIPTION

The purpose of the Software Requirement Specification is to produce the specification of the analysis task and also to establish complete information about the requirement, behavior and also the other constraint like functional performance and so on. The main aim of the Software Requirement Specification is to completely specify the technical requirements for the software product in a concise and in unambiguous manner.

6.1 Visual Studio

In this project the Microsoft visual studio is used as an IDE. Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of our way. The delightfully frictionless edit-build-debug cycle means less time fiddling with our environment, and more time executing on our ideas. Visual Studio Code supports macOS, Linux, and Windows - so we can hit the ground running, no matter the platform.

6.2 Google Colab

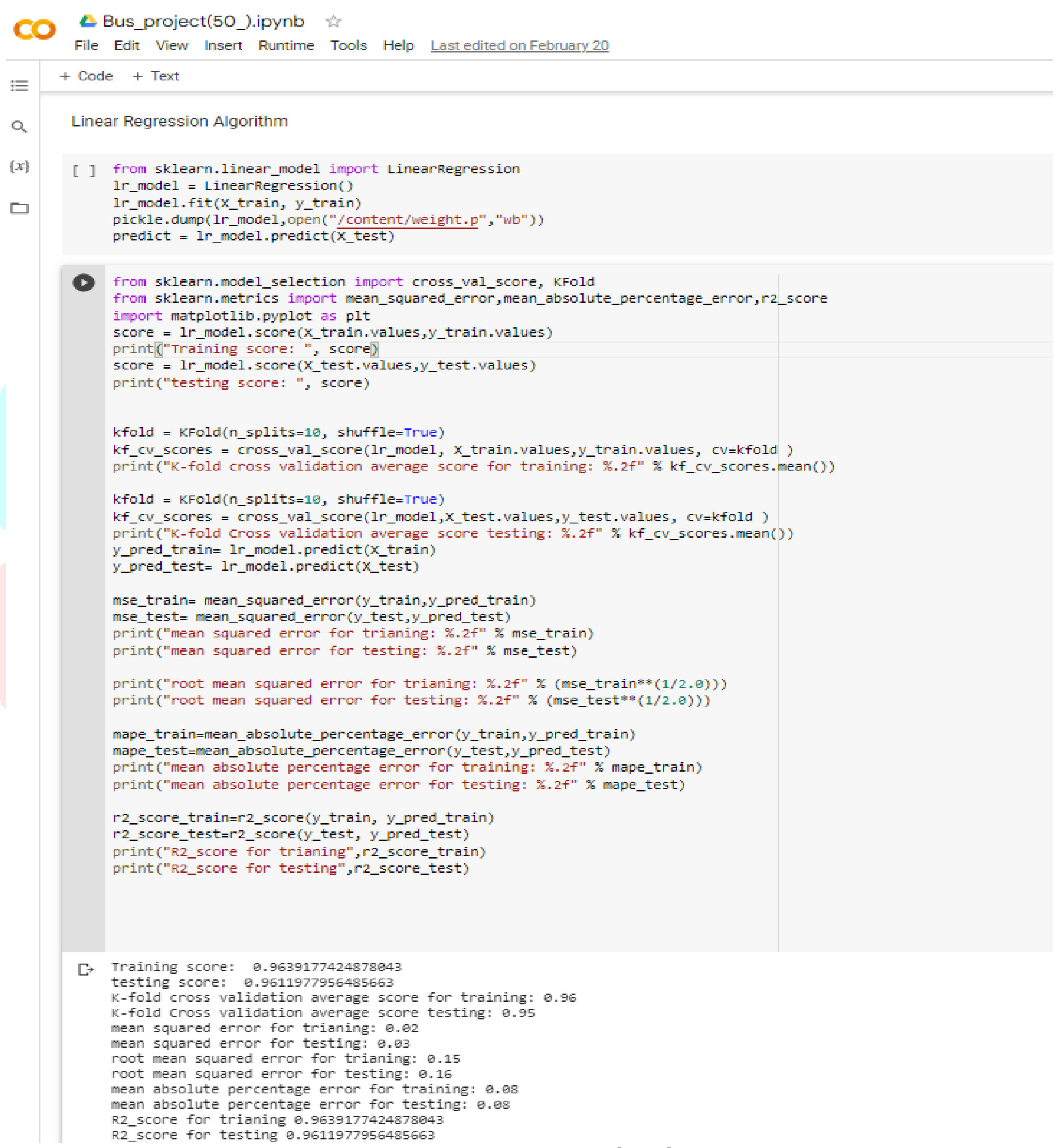
Colab notebooks allow us to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When we create our own Colab notebooks, they are stored in our Google Drive account. We can easily share our Colab notebooks with co-workers or friends, allowing them to comment on our notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook we can use the File menu above or use the following link: create a new Colab notebook. Colab notebooks are Jupyter notebooks that are hosted by Colab.

As a developer, we can perform the following using Google Colab;

- Write and execute code in Python
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

6.3 Linear Regression Algorithm

Linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).


Bus_project(50_).ipynb
File Edit View Insert Runtime Tools Help Last edited on February 20
+ Code + Text
Linear Regression Algorithm

```python
from sklearn.linear_model import LinearRegression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
pickle.dump(lr_model,open("/content/weight.p","wb"))
predict = lr_model.predict(X_test)
```

```python
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error,r2_score
import matplotlib.pyplot as plt
score = lr_model.score(X_train.values,y_train.values)
print("Training score: ", score)
score = lr_model.score(X_test.values,y_test.values)
print("testing score: ", score)


kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(lr_model, X_train.values,y_train.values, cv=kfold )
print("K-fold cross validation average score for training: %.2f" % kf_cv_scores.mean())

kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(lr_model,X_test.values,y_test.values, cv=kfold )
print("K-fold Cross validation average score testing: %.2f" % kf_cv_scores.mean())
y_pred_train= lr_model.predict(X_train)
y_pred_test= lr_model.predict(X_test)

mse_train= mean_squared_error(y_train,y_pred_train)
mse_test= mean_squared_error(y_test,y_pred_test)
print("mean squared error for trianing: %.2f" % mse_train)
print("mean squared error for testing: %.2f" % mse_test)

print("root mean squared error for trianing: %.2f" % (mse_train**(1/2.0)))
print("root mean squared error for testing: %.2f" % (mse_test**(1/2.0)))

mape_train=mean_absolute_percentage_error(y_train,y_pred_train)
mape_test=mean_absolute_percentage_error(y_test,y_pred_test)
print("mean absolute percentage error for training: %.2f" % mape_train)
print("mean absolute percentage error for testing: %.2f" % mape_test)

r2_score_train=r2_score(y_train, y_pred_train)
r2_score_test=r2_score(y_test, y_pred_test)
print("R2_score for trianing",r2_score_train)
print("R2_score for testing",r2_score_test)
```

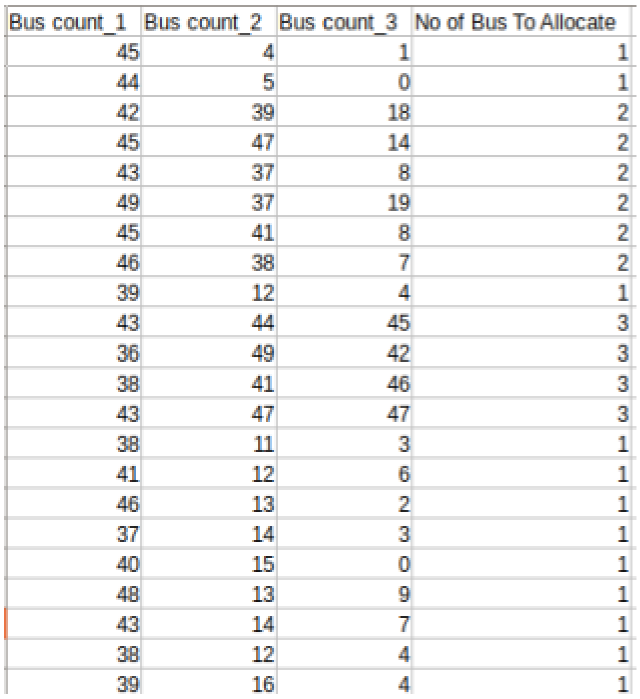
```
Training score:  0.9639177424878043
testing score:  0.9611977956485663
K-fold cross validation average score for training: 0.96
K-fold Cross validation average score testing: 0.95
mean squared error for trianing: 0.02
mean squared error for testing: 0.03
root mean squared error for trianing: 0.15
root mean squared error for testing: 0.16
mean absolute percentage error for training: 0.08
mean absolute percentage error for testing: 0.08
R2_score for trianing 0.9639177424878043
R2_score for testing 0.9611977956485663
```

Fig 6.1 Linear Regression Algorithm.

VII. RESULTS AND DISCUSSION

7.1 Results

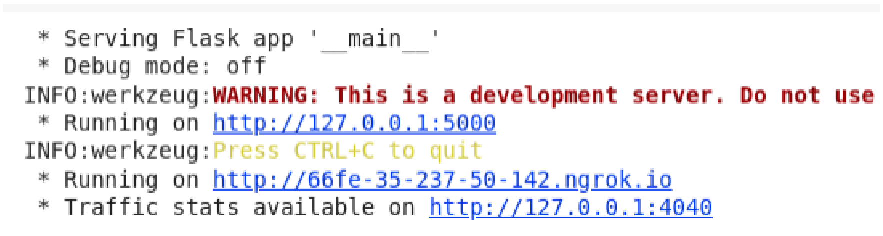
To begin with, testing of the trained model, we can split our project into modules of implementation that is done. Dataset collection involves the process of collecting from various resource. The dataset has been collected for the project and the below figure can be seen as follows:



Bus count_1	Bus count_2	Bus count_3	No of Bus To Allocate
45	4	1	1
44	5	0	1
42	39	18	2
45	47	14	2
43	37	8	2
49	37	19	2
45	41	8	2
46	38	7	2
39	12	4	1
43	44	45	3
36	49	42	3
38	41	46	3
43	47	47	3
38	11	3	1
41	12	6	1
46	13	2	1
37	14	3	1
40	15	0	1
48	13	9	1
43	14	7	1
38	12	4	1
39	16	4	1

Fig 7.1 Past Data.

After getting the input features is processed using linear regression model. Then it sends predicted no of bus count data to front end. Next, the hardware kit is developed using ESP32 and IR sensors. The entry and exit of busses will be fixed with IR sensors. Whenever the passenger enters in the bus, the count will be increased. And whenever the passenger exits in the bus, the count will be decreased.



```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://66fe-35-237-50-142.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```

Fig 7.2 Backend integrate with front end using flask framework and ngrok.

The hardware kit of the buses can be seen in the below figure.

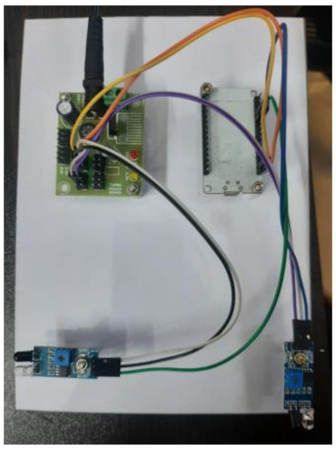


Fig 7.3 Bus Hardware Kit

References

[1] Ampountolas, K., & Kring, M. (2020). Mitigating Bunching with Bus-Following Models and Bus-to-Bus Cooperation. IEEE Transactions on Intelligent Transportation Systems, 1–10. doi:10.1109/tits.2020.2973585

[2] Barabino, B., Coni, M., Olivo, A., Pungillo, G., & Rassu, N. (2019). Standing Passenger Comfort: A New Scale for Evaluating the Real-Time Driving Style of Bus Transit Services. IEEE Transactions on Intelligent Transportation Systems, 1– 14. doi:10.1109/tits.2019.2921807

[3] Chen, X., Wang, Y., & Ma, X. (2021). Integrated Optimization for Commuting Customized Bus Stop Planning, Routing Design, and Timetable Development with Passenger Spatial-Temporal Accessibility. IEEE Transactions on Intelligent Transportation Systems, 22(4), 2060–2075. doi:10.1109/tits.2020.3048520

[4] Gkiotsalitis, K. (2020). Bus Holding of Electric Buses with Scheduled Charging Times. IEEE Transactions on Intelligent Transportation Systems, 1– 12. doi:10.1109/tits.2020.2994538

[5] Gokasar, I., Cetinel, Y., & Baydogan, M. G. (2019). Estimation of Influence Distance of Bus Stops Using Bus GPS Data and Bus Stop Properties. IEEE Transactions on Intelligent Transportation Systems, 1– 8. doi:10.1109/tits.2019.2909645

[6] Handte, M., Foell, S., Wagner, S., Kortuem, G., & Marron, P. J. (2019). An Internet-of-Things Enabled Connected Navigation System for Urban Bus Riders. IEEE Internet of Things Journal, 3(5), 735–744. doi:10.1109/jiot.2016.2554146

[7] Haoyang Yan, Zhiyong Cui, Xinqiang Chen, and Xiaolei Ma. (2021). Distributed Multi-Agent Deep Reinforcement Learning for Multi-line Dynamic Bus Timetable Optimization. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. DOI 10.1109/TII.2022.315865

[8] He, P., Jiang, G., Lam, S.-K., & Tang, D. (2018). Travel-Time Prediction of Bus Journey With Multiple Bus Trips. IEEE Transactions on Intelligent Transportation Systems, 1–14. doi:10.1109/tits.2018.2883342

[9] Liu, Y., Lyu, C., Liu, X., & Liu, Z. (2020). Automatic Feature Engineering for Bus Passenger Flow Prediction Based on Modular Convolutional Neural Network. IEEE Transactions on Intelligent Transportation Systems, 1– 10. doi:10.1109/tits.2020.3004254

[10] Luo, D., Zhao, D., Ke, Q., You, X., Liu, L., Zhang, D., … Zuo, X. (2020). FineGrained Service-Level Passenger Flow Prediction for Bus Transit Systems Based on Multitask Deep Learning. IEEE Transactions on Intelligent Transportation Systems, 1–16. doi:10.1109/tits.2020.3002772

[11] Qiu, G., Song, R., He, S., Xu, W., & Jiang, M. (2018). Clustering Passenger Trip Data for the Potential Passenger Investigation and Line Design of Customized Commuter Bus. IEEE Transactions on Intelligent Transportation Systems, 1– 10. doi:10.1109/tits.2018.2875466

[12] Verma, R., Shrivastava, A., De, K., Mitra, B., Saha, S., Ganguly, N., … Chakraborty, S. (2020). A Smartphone-Based Passenger Assistant for Public Bus Commutes in Developing Countries. IEEE Transactions on Computational Social Systems, 1–12. doi:10.1109/tcss.2019.2961390

[13] Wepulanon, P., Sumalee, A., & Lam, W. H. K. (2019). Temporal Signatures of Passive Wi-Fi Data for Estimating Bus Passenger Waiting Time at a Single Bus Stop. IEEE Transactions on Intelligent Transportation Systems, 1– 11. doi:10.1109/tits.2019.2926577