International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 3 (1) 2025

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 3(1), 2025

Research Paper

ADAPTIVE DEEP-LEARNING FRAMEWORK FOR REAL-TIME HOURLY BUS BOARDING DEMAND CLASSIFICATION IN URBAN TRANSIT

Ritesh Kumar[1], Guneeth Rudrapati[2], Goutham Simha. S[2], Naga Sai Manikanta[2], Balraj Naik Bhukya[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering
[1,2]Kommuri Pratap Reddy Institute of Technology, Hyderabad, Telangana, India.
[1]Email: riteshkumar237@gmail.com.

Received: 05-6-2025 Accepted: 06-7-2025 Published: 14-7-2025

ABSTRACT

Public transportation systems face significant fluctuations in passenger demand, with over 60% of daily boardings occurring during peak hours and nearly 30% of commuters experiencing delays due to overcrowding. In densely networked urban transit systems, about 70% of route optimization efforts fail without accurate boarding predictions. Traditional manual demand estimation techniques such as time-slot passenger counts, field route surveys, and ticket ledger analysis are often hindered by human error, time-consuming procedures, and limited scalability, resulting in inconsistent data and ineffective planning. To address these challenges, this study introduces an AI-driven boarding demand prediction model that utilizes Support Vector Machine (SVM) and Deep Neural Network (DNN) classifiers. The system integrates real-time data inputs, including timestamp, route number, traffic delays, and historical boarding volumes, to train and evaluate the models using accuracy-based metrics for optimal prediction performance. Furthermore, the AI-based results are benchmarked against three conventional analysis methods—time-slot monitoring, on-site route surveys, and ticket data review to ensure result reliability and model robustness. The proposed solution delivers a high-accuracy, real-time prediction framework that supports dynamic scheduling and minimizes the mismatch between supply and demand. This integration of machine learning into public transport planning offers a scalable, data-centric alternative to manual approaches, paving the way for smarter, more efficient transit systems.

Keywords: Public Transportation Planning, Bus Boarding Demand, Passenger Flow Prediction, Smart Mobility Solutions, Deep Neural Network (DNN).

1. INTRODUCTION

Passenger demand prediction plays a critical role in the operational efficiency of public transport systems. According to the International Association of Public Transport (UITP), global public transport ridership exceeded 253 billion passenger journeys in 2022, with rapid urbanization fueling this growth. Urban regions like Tokyo, London, and New York witness over 10 million daily rides, often straining infrastructure and causing inefficiencies due to mismatches in demand and supply. Accurate passenger demand forecasting not only supports dynamic scheduling and route optimization but also ensures optimal resource allocation, reduced waiting times, and improved commuter satisfaction.

Traditional demand estimation methods, relying heavily on historical averages or simple regression models, often fail to capture real-time

fluctuations caused by external factors such as weather, public events, or traffic disruptions. A study by the European Commission revealed that delays and overcrowding stemming from demand mismanagement cost EU economies approximately €8 billion annually. Inaccurate forecasting leads to underutilized vehicles in off-peak hours and overcrowding during peak periods, highlighting the urgent need for advanced, data-driven prediction methods.

The rise of digital ticketing systems, mobile apps, GPS-based vehicle tracking, and Internet of Things (IoT) sensors has enabled real-time data collection across many cities. These rich data streams present an opportunity to apply machine learning and deep learning techniques to better understand passenger behavior. By capturing spatiotemporal patterns and contextual information, such models can significantly outperform traditional approaches, thus transforming public transport into a more resilient, responsive, and commuter-centric system.

2. LITERATURE SURVEY

This approach focuses on formulating operational constraints such as bus capacity, operation schedules, and passenger demand. Generally, these methods aim to minimize operational costs or maximize system efficiency. For example, Borndörfer et al. [1] and Zhou et al. [2] addressed the problem using Mixed Integer Linear Programming (MILP) to minimize operational costs and travel times, considering constraints such as vehicle capacity, synchronization between routes, and minimization of transfer times. In these studies, the problem was formulated as a multi-commodity flow model, which allowed for dynamically generating optimized transit lines. However, the second approach uses column generation to handle the computational complexity of capacity constraints, achieving solutions applicable to real networks.

On the other hand, Pei et al. [3] explored the use of modular vehicles in transport networks through an MILP approach that dynamically adjusts vehicle capacities and frequencies to respond to fluctuations in demand. This solution showed a significant reduction in operational costs and travel times compared to traditional systems. Guan et al. [4] proposed a dispatch and route optimization model for public transportation systems with variable demand, using a hybrid approach based on a hybrid LNS-genetic algorithm, which resulted in significant improvements in operational efficiency and user service. Van Oudheusden et al. [5] introduced a nonlinear programming approach to optimize frequencies and schedules, focusing on minimizing empty trips performed by the fleet and increasing the number of transported passengers. To avoid stochastic uncertainties, Van Berkum et al. [6] formulated the problem within a rolling horizon framework, dividing an operational day into predetermined intervals. This approach uses a convex nonlinear formulation that allows solving the problem to global optimality with a limited computational cost, simultaneously optimizing the dispatch times of all scheduled trips in each interval. Gkiotsalitis [7] demonstrated that periodic optimization through convex quadratic programming can minimize variations in vehicle departure intervals, improving service regularity in high-frequency lines, as evidenced in the case study presented in the 302 bus network in Singapore.

Another notable approach in the reviewed literature is that of Chen and Zhou [8], who implemented a Dynamic Programming (DP) algorithm to solve the dispatch problem in oversaturated systems. This approach efficiently handles constraints such as vehicle capacities and waiting times, applying valid inequalities to reduce the computational complexity of DP. The results demonstrated significant reductions in operational costs and waiting times, with

practical applications in networks such as the Beijing metro and Tampa Bay.

Although exact methods guarantee optimal solutions, they face limitations in terms of scalability and applicability in dense networks or highly dynamic systems, as they require computationally expensive methods such as Branch and Bound to achieve optimality [9]. In these cases, heuristics provide fast and flexible solutions, sacrificing precision for efficiency. These methodologies explore the solution space using operational rules or adaptive algorithms. Hadas et al. [10] combined an analytical and iterative approach that jointly optimizes frequencies, schedules, and transfers in a public transportation network. This methodology adjusts bus dispatching based on stochastic simulations using historical data, balancing service quality and operational efficiency. Similarly, Eranki [11] developed an iterative heuristic focused on assigning departure schedules to maximize simultaneous arrivals at transfer points, classifying nodes according to their relevance in the allocation process.

3. PROPOSED SYSTEM

The proposed system introduces a hybrid ensemble-based demand prediction framework that integrates classical and deep learning models with an optimized demand classification mechanism. This framework uniquely combines Count Vectorization for structured categorical route identifiers, Support Vector Machine (SVM) for high-dimensional sparse classification, and a Deep Neural Network (DNN) to capture non-linear relationships across temporal-spatial boarding data. The fusion of their outputs is used not just for accuracy evaluation, but also for real-time ratio tracking of boarding demand types (More/Less) across stops. Unlike existing works that focus purely on single model prediction or general ML pipelines, this system incorporates a decision-level ensemble, post-prediction demand ratio profiling, and temporal route plotting over 7-day intervals, all wrapped into a service-provider-driven visualization panel with XLS export and live model assessment dashboards — a combination not reported in prior transit demand prediction literature.

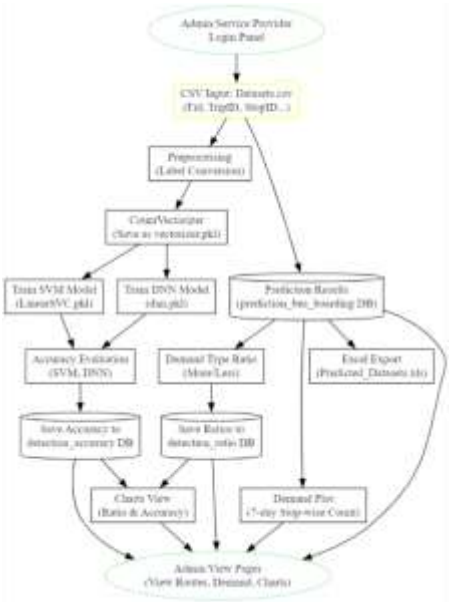

Fig. 1: Proposed System Architecture for Service Provider module.

Django Framework

Figure 2 illustrates the modular architecture of the User-side Passenger Demand Prediction System, showcasing the interaction between the Client Interface, Django-based Server Backend, and Database Layer (SQLite/MySQL). The Client Side comprises user-accessible pages including the LoginPage, RegisterPage, Dashboard, and PredictDemandPage, which initiate HTTP requests to the backend server. These requests are routed through urls.py, then processed by views.py, and rendered via appropriate HTML Templates. The Server Side manages logic, request handling, and model interaction through Django's MVC components: views.py, models.py, and templates. The Database Layer stores user credentials (ClientRegister_Model), predefined route data (route), and predictions (prediction_bus_boarding). The system ensures seamless communication across these layers for user registration, login, demand prediction, and

data visualization. The architecture supports modular, scalable, and secure processing using Django's ORM and routing system.

Step 1: User Authentication: The user accesses the system through the LoginPage or RegisterPage. On registration or login, the client issues a POST request to the server where the views.py controller verifies or stores the user's information in the ClientRegister_Model table via the Django model layer.

Step 2: URL Routing and View Handling: All client-side page interactions are routed via urls.py, which maps URLs to their corresponding view functions in views.py. For instance, the /login, /register, /dashboard, and /predict endpoints are processed by appropriate view logic.

Step 3: Dashboard Access: Once logged in, the user is redirected to the Dashboard via a GET /profile request. The views.py fetches user-specific data from the ClientRegister_Model and renders the corresponding dashboard through the defined HTML Templates.

Step 4: Demand Prediction Request: The user navigates to the PredictDemandPage and submits demand-related inputs (e.g., stop ID, route ID, boarding time) via a POST /predict request. This is handled by views.py, which loads the pretrained machine learning models (e.g., SVM, DNN), applies the vectorizer on the input, and performs prediction.

Step 5: Prediction Storage and Response: The prediction output is stored in the prediction_bus_boarding table, maintaining a record of all predicted outcomes for later analysis. The result is simultaneously returned to the frontend and displayed on the dashboard.

Step 6: Template Rendering and Visualization: All responses — such as prediction results, charts, or tables — are dynamically rendered using Django HTML Templates. These templates are updated based on server responses, ensuring real-time interactivity for the user.

Step 7: ORM-based Model-DB Communication: Django's models.py defines the schema for all involved tables (ClientRegister_Model, route, prediction_bus_boarding). The ORM handles communication between these model objects and the database backend, ensuring smooth query execution and data manipulation without raw SQL.

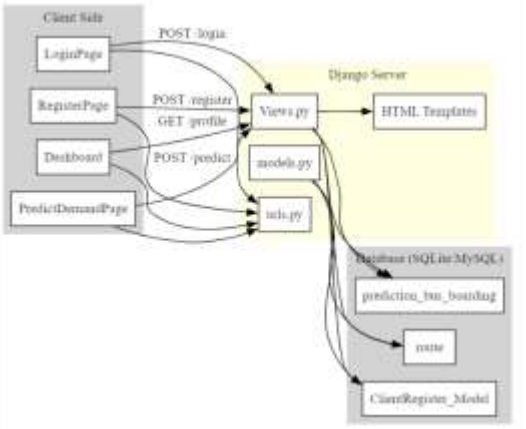

Fig. 2: Proposed System Architecture for USER module.

DNN Classifier

The DNN method is an advanced neural network architecture designed to detect complex, non-linear patterns within high-dimensional datasets, such as those used in internet bus Transport demand detection. In this specific application, bus Transport applications often contain subtle correlations between features that may not be captured by traditional models. The DNN architecture excels in learning such intricate relationships, making it suitable for distinguishing between genuine and demand applications. Its ability to automatically learn hierarchical feature representations through multiple layers gives it a significant advantage in demand detection scenarios where demand behaviors evolve and vary widely. Additionally, DNN models are highly scalable and adaptable, making them ideal for real-time, large-scale financial applications.

Step 1: Input Layer Initialization: The process begins with the input layer, where each node

corresponds to a feature in the preprocessed bus Transport application dataset. These features may include numerical or encoded categorical values such as transaction amount, account type, origin and destination identifiers, and more. This layer serves as the entry point for data into the neural network.

Step 2: Dense Layer Construction: After the input layer, multiple Dense Layers are added to the network. Each Dense Layer consists of several neurons that apply weighted transformations to the inputs followed by a non-linear activation function, such as ReLU. These layers are responsible for capturing complex interactions between features, gradually learning higher-level abstractions and patterns that help differentiate between demand and legitimate applications.

Step 3: Activation Function Application: Activation functions are applied within each Dense Layer to introduce non-linearity into the model, allowing it to learn from complex data. ReLU (Rectified Linear Unit) is commonly used due to its efficiency and effectiveness in avoiding vanishing gradient issues. This step ensures the network can capture intricate variations in data, crucial for demand detection.

Step 4: Dropout and Regularization: To prevent overfitting and improve generalization, dropout layers may be introduced between Dense Layers. Dropout randomly disables a fraction of neurons during training, ensuring that the model does not rely too heavily on specific pathways. Additionally, techniques like L2 regularization can be applied to penalize large weights and promote simpler, more robust models.

Step 4: Output Layer and Classification: The final layer in the DNN model is the output layer, which typically uses a sigmoid or softmax activation function depending on whether binary or multi-class classification is required. In demand detection, a sigmoid function is often used to output a probability score indicating the likelihood of a transaction being demand.

Step 5: Backpropagration: The DNN is trained using backpropagation and an optimization algorithm like Adam or SGD. During training, the model adjusts weights based on a loss function such as binary cross-entropy. The optimization aims to minimize the difference between predicted and actual labels, improving the model's ability to detect demand with high accuracy.

Step 6: Model Evaluation: Once trained, the DNN model is evaluated using metrics such as accuracy, precision, recall, F1-score. These metrics help assess the model's performance on unseen data. After validation, the trained DNN can be deployed into a real-time demand detection system to continuously analyze incoming bus Transport applications.

Advantages

Advanced neural networks can capture complex, non-linear relationships in the data, allowing them to detect subtle patterns indicative of demand that may not be apparent with simpler models. These architectures can automatically learn relevant features from raw data, reducing the need for manual feature engineering and potentially uncovering hidden indicators of demand behavior. Neural networks can handle large volumes of data efficiently, making them suitable for processing the vast amounts of information typically encountered in internet bus Transport applications. Neural networks can adapt to changing patterns of demand over time, making them robust against evolving tactics used by demandsters.

4. RESULTS AND DISCUSSION

Figure 3 displays a table of predicted hourly boarding demand, as implemented in the View_Predicted_Hourly_Boarding_Demand_Type view. The table includes columns for Fid, TripID, RouteID, StopID, StopName, WeekBeginning, NumberOfBoardings, and Prediction. Sample records include entries like

Fid: 39.159.31.120-4.141.105.192-32060-43-75, TripID: TID_48351, RouteID: RID_686, StopID: SID_75943, StopName: Masab Tank, WeekBeginning: 08-09-2022 00:00, NumberOfBoardings: 30, Prediction: More Demand, and others with predictions of More Demand or Less Demand. The table shows a mix of 11 More Demand and 26 Less Demand predictions, reflecting the system's ability to classify bus stops based on demand.

Fig. 3: Prediction of Hourly Demand From Test Data.

Fig. 4: Predicting Hourly Boarding Demand Found Ratio Details.

Figure 4 presents a table of demand prediction ratios, as implemented in the View_Predicted_Hourly_Boarding_Demand_Type_Ratio view. The table lists Hourly Boarding Demand Type and Ratio, with values: Less Demand: 75.60975609756098% and More Demand: 24.390243902439025%. These ratios are calculated from the prediction_bus_boarding model, indicating that 75.61% of predictions are for low demand, while 24.39% are for high demand, providing insights into demand distribution.

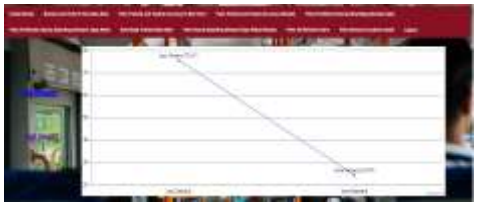

Fig. 5: Predicting Hourly Boarding Demand Found Ratio Graph.

Figure 5 shows a graphical representation of the demand ratios from Figure 9.8, likely generated by the charts view. The graph visualizes Less Demand (75.60975609756098%) and More Demand (24.390243902439025%), possibly as a bar or pie chart using Chart.js or Seaborn. This visualization aids service providers in understanding the proportion of high and low demand predictions across bus stops.

Fig. 6: Predicting Hourly Boarding Demand Found Ratio Graph.

Figure 6 appears to duplicate the functionality of Figure 9.9, showing another graph of the demand ratios (Less Demand: 75.60975609756098%, More Demand: 24.390243902439025%). The repetition suggests either an error in the figure numbering or an alternative visualization style (e.g., different chart type or template) for the same data, reinforcing the system's focus on demand distribution analysis.

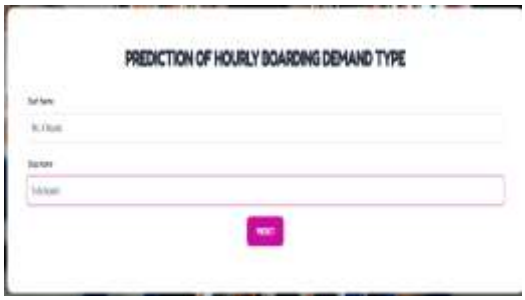

Fig. 7: Input to Prediction of Hourly Boarding Demand Type

Figure 7 illustrates the input form for predicting hourly boarding demand, as part of the Predict_Hourly_Boarding_Demand_Type view. The form includes fields for selecting Start and Stop Names from the route model, enabling remote users to choose a specific bus route for

prediction. This interface facilitates user interaction with the prediction system, leveraging the pre-trained ensemble model.

Fig. 8: Prediction of Hourly Boarding Demand Type as More Demand.

Figure 8 shows the output of a demand prediction, displaying "More Demand" for a selected route, as implemented in the Predict_Hourly_Boarding_Demand_Type view. The prediction is based on the ensemble model's output (1 for "More Demand") for the selected route's Fid, with a 5-minute cooldown enforced to prevent overuse. This result informs users of high demand at the chosen bus stop, aiding in travel planning.

5. CONCLUSION

The developed Django-based web application demonstrates a robust and scalable approach to predicting and analyzing hourly bus boarding demand, significantly contributing to smarter urban transportation management. By leveraging powerful machine learning models such as SVM and DNN, the system accurately classifies demand patterns, enabling data-driven decision-making for transit authorities. Its well-structured backend supports seamless data processing, model training, and real-time predictions, while user-friendly interfaces facilitate easy access to analytics, visualization, and downloadable reports. Admin features further enhance control and transparency within the system. The modular design and persistent model storage ensure long-term maintainability, efficient performance, and adaptability to future expansions. Overall, the application stands as a practical and intelligent tool for optimizing public transit operations and addressing the limitations of traditional demand estimation methods.

REFERENCES

[1] Borndörfer, R.; Grötschel, M.; Pfetsch, M.E. A Path-Based Model for Line Planning in Public Transport A Path-Based Model for Line Planning in Public Transport *. Technical Report. 2005. Available online: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=http://webdoc.sub.gwdg.de/ebook/serien/ah/reports/ZIBreport/ZR-05-18.pdf&ved=2ahUKEwiQrJ6R6pyNAxVjVmwGHYLRJroQFnoECBwQAQ&usg=AOvVaw1KSs6HedcL39KmDr41Voqj (accessed on 5 May 2025).

[2] Zhou, X.; Wei, G.; Zhang, Y.; Wang, Q.; Guo, H. Optimizing Multi-Vehicle Demand-Responsive Bus Dispatching: A Real-Time Reservation-Based Approach. Sustainability 2023, 15, 5909.

[3] Pei, M.; Lin, P.; Du, J.; Li, X.; Chen, Z. Vehicle dispatching in modular transit networks: A mixed-integer nonlinear programming model. Transp. Res. Part E Logist. Transp. Rev. 2021, 147, 102240.

[4] Guan, D.; Wu, X.; Wang, K.; Zhao, J. Vehicle Dispatch and Route Optimization Algorithm for Demand-Responsive Transit. Processes 2022, 10, 2651.

[5] van Oudheusden, D.L.; Zhu, W. Trip frequency scheduling for bus route management in Bangkok. Eur. J. Oper. Res. 1995, 83, 439–451.

[6] Gkiotsalitis, K.; van Berkum, E.C. An exact method for the bus dispatching problem in rolling horizons. Transp.

Res. Part C Emerg. Technol. 2020, 110, 143–165.

[7] Gkiotsalitis, K. A model for the periodic optimization of bus dispatching times. Appl. Math. Model. 2020, 82, 785–801.

[8] Chen, Z.; Li, X.; Zhou, X. Operational design for shuttle systems with modular vehicles under oversaturated traffic: Discrete modeling method. Transp. Res. Part B Methodol. 2019, 122, 1–19.

[9] Gkiotsalitis, K.; Cats, O. Reliable frequency determination: Incorporating information on service uncertainty when setting dispatching headways. Transp. Res. Part C Emerg. Technol. 2018, 88, 187–207.

[10] Hadas, Y.; Shnaiderman, M. Public-transit frequency setting using minimum-cost approach with stochastic demand and travel time. Transp. Res. Part B Methodol. 2012, 46, 1068–1084.

[11] Eranki, A. A Model to Create Bus Timetables to Attain Maximum Synchronization Considering Waiting Times at Transfer Stops. Master's Thesis, Department of Industrial and Management Systems Engineering University of South Florida, Tampa, FL, USA, 2004. Available online: http://scholarcommons.usf.edu/etd/1025 (accessed on 15 February 2025).