Adaptive Feature Fusion Networks for Origin-Destination Passenger Flow Prediction in Metro Systems

Ms. Sadia Kausar[1], Md Zuber Ali[2], Abdul Rahman Anas[3], Amaan Ahmed4, Ahmed Waliuddin Quadri[5]
[1]*Assistant Professor Department of CSE-AIML, Lords Institute of Engineering and Technology, Hyderabad, India*
[2,3,4,5]*BTech Students Department of CSE-AIML, Lords Institute of Engineering and Technology, Hyderabad, India*

Abstract—Accurate prediction of Origin-Destination (OD) passenger flow is a critical component of intelligent metro transportation systems, enabling efficient scheduling, congestion mitigation, and improved passenger experience. Unlike traditional station-level inflow and outflow prediction, OD prediction provides a fine-grained understanding of passenger movement across the entire metro network. However, this task is inherently challenging due to complex spatial dependencies between stations, highly dynamic temporal patterns, and the presence of sparse and incomplete OD matrices caused by delayed trip completion.

In this paper, we propose an advanced deep learning framework termed Adaptive Feature Fusion Network (AFFN) to address these challenges. The proposed model integrates heterogeneous data sources and captures multi-dimensional dependencies through a unified architecture. Specifically, an Enhanced Multi-Graph Convolution Gated Recurrent Unit (EMGC-GRU) is designed to model spatial relationships by leveraging multiple knowledge-based graphs and automatically learned attention-based graphs to capture hidden correlations among stations. To incorporate temporal dynamics, the GRU structure is embedded with graph convolution operations, enabling simultaneous learning of spatial-temporal patterns.

I. INTRODUCTION

Metro systems have become one of the most essential components of modern urban transportation Furthermore, an external factor-aware attention mechanism is introduced to model the influence of contextual variables such as weather conditions, holidays, and special events on passenger flow. This module adaptively weights periodic patterns, allowing the model to adjust predictions dynamically under varying conditions. To mitigate the issue of sparsity and incompleteness in OD matrices, a multi-task learning strategy is employed, where inflow and outflow (IO) prediction is jointly learned alongside OD prediction, improving overall model robustness and accuracy.

Additionally, an extension of the proposed model using EMGC-LSTM is developed to enhance long-term dependency learning capabilities. Experimental evaluations conducted on real-world metro datasets demonstrate that the proposed AFFN and its extension significantly outperform traditional machine learning and existing deep learning models in terms of RMSE and MAP metrics. The results confirm that adaptive feature fusion combined with multi-task learning provides a powerful and especially in densely populated metropolitan regions. With rapid urbanization and continuous population growth, the demand for efficient, reliable, and scalable public transportation has increased significantly. Metro networks, known for their high capacity, speed, and efficiency, are widely adopted across major cities worldwide. In cities such as Tokyo, New York, and Hong Kong, metro systems account for a substantial proportion of daily commuting, often exceeding 80% of total passenger movement. As a result, effective management and optimization of metro operations have become critical for ensuring smooth mobility, reducing congestion, and improving passenger satisfaction.

A key requirement for efficient metro system management is the accurate prediction of passenger demand. Traditionally, most research efforts have focused on station-level inflow and outflow (IO) prediction, which estimates the number of passengers entering and exiting individual stations. While such

approaches provide useful insights for localized decision-making, they fail to capture the complete travel behavior of passengers across the network. In contrast, Origin-Destination (OD) passenger flow prediction, which estimates the number of passengers traveling between each pair of stations, provides a more comprehensive understanding of passenger mobility patterns. This fine-grained information is crucial for advanced applications such as dynamic timetable scheduling, route planning, congestion management, and emergency response strategies.

Despite its significance, OD passenger flow prediction remains a highly challenging problem due to several inherent complexities in metro systems. Passenger flow exhibits strong spatial dependencies, where the relationship between stations is influenced not only by physical connectivity but also by functional similarities, surrounding urban structures, and hidden correlations that are not directly observable. Furthermore, metro passenger demand is highly dynamic over time, with significant variations during peak hours, weekdays, weekends, and seasonal changes. These temporal patterns often involve both short-term fluctuations and long-term dependencies, making them difficult to model using conventional techniques. In addition to spatial and temporal complexities, external factors such as weather conditions, public holidays, and special events further influence passenger behavior, introducing additional variability into the system.

Another major challenge arises from the nature of OD data itself. In real-world metro systems, OD matrices are typically sparse, as only a limited number of station pairs account for the majority of passenger trips, while many pairs have minimal or no interactions. Moreover, OD data is often incomplete in real-time scenarios because passenger destinations are only known after trip completion, leading to missing or delayed information during prediction. These challenges make it difficult for traditional statistical and machine learning models to achieve high prediction accuracy.

In recent years, deep learning techniques have shown great potential in modeling complex spatial-temporal data. Approaches based on recurrent neural networks (RNNs), long short-term memory (LSTM), and graph convolutional networks (GCNs) have been proposed to capture temporal dynamics and spatial relationships in transportation systems. However, existing methods often rely on single-source feature representations or fail to effectively integrate multiple heterogeneous data sources. As a result, they are limited in their ability to capture the full complexity of metro passenger flow patterns.

To overcome these limitations, this paper proposes an Adaptive Feature Fusion Network (AFFN) for OD passenger flow prediction in metro systems. The proposed model is designed to adaptively integrate spatial dependencies, temporal dynamics, and external influencing factors within a unified deep learning framework. By leveraging advanced architectures such as multi-graph convolution and recurrent neural networks, the model captures both explicit and hidden correlations among metro stations while simultaneously learning temporal patterns of passenger flow. Furthermore, a multi-task learning strategy is introduced to jointly predict OD flow and station-level inflow and outflow, thereby addressing the challenges of data sparsity and incompleteness. The proposed approach provides a robust and scalable solution for accurate OD prediction, contributing to the development of intelligent and efficient metro transportation systems.

1.1 Objective of the Project

The primary objective of this project is to develop an accurate and efficient model for predicting Origin-Destination (OD) passenger flow in metro systems by addressing the limitations of existing prediction approaches. Unlike traditional methods that focus only on station-level inflow and outflow, this project aims to provide a comprehensive understanding of passenger movement across the entire metro network by estimating travel demand between every pair of stations. Achieving this requires the ability to model complex spatial and temporal dependencies that exist in real-world transportation systems. Therefore, the project is designed to capture spatial relationships among metro stations using multiple knowledge-based graphs, while simultaneously learning temporal patterns such as daily, weekly, and seasonal variations in passenger flow. Another important objective is to incorporate the influence of external factors such as weather conditions, public holidays, and special events, which significantly impact passenger behavior but are often overlooked or poorly integrated in existing models.

II. LITERATURE SURVEY

2.1 Urban Rail Transit Development and Trends

Urban rail transit systems have witnessed rapid growth in recent years, particularly in developing countries such as China, where large-scale metro infrastructure development has significantly transformed urban mobility. The expansion of metro networks has improved accessibility, reduced traffic congestion, and enhanced the overall efficiency of public transportation systems. Studies analyzing national policies and strategic plans highlight that factors such as population density, urbanization, and economic development play a crucial role in shaping metro system expansion. Additionally, the increasing demand for sustainable and energy-efficient transportation has further accelerated investments in metro systems. Understanding these development trends is essential for designing predictive models that can accommodate growing passenger demand and evolving transportation patterns.

2.2 Timetable Optimization Based on Passenger Demand

Efficient timetable planning is a critical aspect of metro operations, as it directly impacts passenger waiting time, service quality, and energy consumption. Recent research has focused on developing optimization models that consider time-dependent passenger flow, enabling dynamic adjustment of train schedules based on demand variations. These models aim to minimize both passenger waiting time and energy consumption by incorporating constraints such as train capacity and operational limitations. By utilizing real-world data, such as smart card transactions, these approaches provide more realistic and adaptive scheduling solutions. However, their effectiveness largely depends on the accuracy of passenger flow prediction, highlighting the importance of reliable OD prediction models.

2.3 Energy-Efficient Train Scheduling and Rescheduling

In high-density metro systems, unexpected disturbances such as delays or equipment failures can disrupt normal operations and lead to passenger congestion. To address these issues, energy-efficient train scheduling and rescheduling models have been proposed. These models aim to optimize multiple objectives, including minimizing train delays, reducing passenger congestion, and lowering energy consumption. By incorporating dynamic passenger flow and automatic train operation (ATO) profiles, these approaches can adjust train schedules in real time to improve system performance. Despite their advantages, such models require accurate and timely passenger flow predictions, particularly OD flow, to make effective decisions under uncertain conditions.

2.4 Skip-Stop Operation Optimization

Skip-stop operation is a strategy used in metro systems to improve travel efficiency by allowing trains to skip certain stations, thereby reducing travel time for long-distance passengers. Recent studies have proposed flexible skip-stop schemes that adapt to varying passenger demand patterns across different stations and time periods. These approaches use optimization techniques such as genetic algorithms to determine the optimal stopping pattern that minimizes overall passenger travel time. By leveraging smart card data, these models can capture spatial and temporal variations in demand. However, the success of skip-stop strategies heavily depends on accurate OD flow estimation, as incorrect predictions may lead to passenger inconvenience or increased congestion at skipped stations.

2.5 Deep Learning Approaches for Passenger Flow Prediction

With the advancement of artificial intelligence, deep learning models have been increasingly applied to metro passenger flow prediction. Frameworks such as Deep Passenger Flow (DeepPF) integrate multiple data sources, including temporal trends, spatial features, and external factors, into a unified architecture. These models are capable of capturing complex nonlinear relationships and have demonstrated superior performance compared to traditional statistical methods. Additionally, graph-based deep learning models, such as spatio-temporal graph convolution networks, further enhance prediction accuracy by modeling the metro system as a network of interconnected stations. Despite these advancements, many existing models still face challenges in effectively fusing heterogeneous features and handling sparse OD data, which motivates the development of more advanced approaches such as the Adaptive Feature Fusion Network (AFFN).

III. SYSTEM ANALYSIS

3.1 Existing System

The existing system for OD passenger flow estimation mainly includes three approaches:

1. Direct Estimation: Uses surveys to collect passenger trip data, but it is time-consuming and not suitable for real-time prediction.
2. Dis-Aggregated Estimation: Uses models based on calibration and validation (RP/SP methods). It can estimate current and future demand but requires complex modeling.
3. Aggregated Estimation: Uses traffic data to estimate OD flow by minimizing the difference between observed and predicted values. However, accuracy is affected due to difficulty in estimating assignment matrices.

Disadvantages:

- Does not use Adaptive Feature Fusion Network (AFFN).
- Lacks integration of multiple graph-based learning methods for better accuracy.

3.2 Proposed System

The proposed system introduces an Adaptive Feature Fusion Network (AFFN) to improve OD passenger flow prediction.

- It combines spatial dependencies (between stations) and temporal patterns using multiple graphs and attention mechanisms.
- Uses EMGC-GRU to capture spatial and temporal relationships effectively.
- Incorporates external factors (like weather, holidays) using attention mechanisms.
- Extends to multi-task learning by predicting inflow and outflow (IO) to improve accuracy.

Advantages:

- Captures complex spatial correlations using multi-graph learning.
- Improves prediction using external factor-based attention.
- Multi-task learning enhances overall accuracy.
- Achieves better performance compared to existing methods

Extension:

- GRU is replaced with LSTM to further improve prediction accuracy.

Modules:

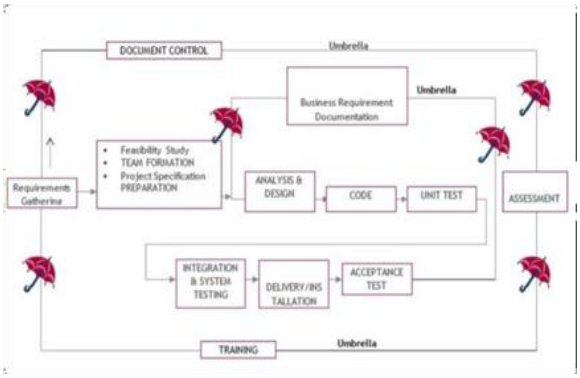
- User Login
- Dataset Upload & Processing
- Existing MLP Model
- Proposed AFFN (GRU)
- Extended AFFN (LSTM)
- Performance Graph

3.3 Process Model — SDLC (Umbrella Model)

SDLC stands for Software Development Life Cycle. It is a standard which is used by the software industry to develop good software. The Umbrella Model is adopted for this project as it supports continuous maintenance with no defined endpoint.

Stages in SDLC:

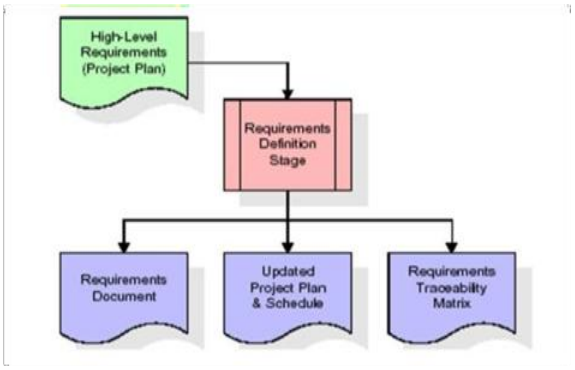
1. Requirements Gathering
2. Analysis
3. Designing
4. Coding (Development)
5. Testing
6. Maintenance



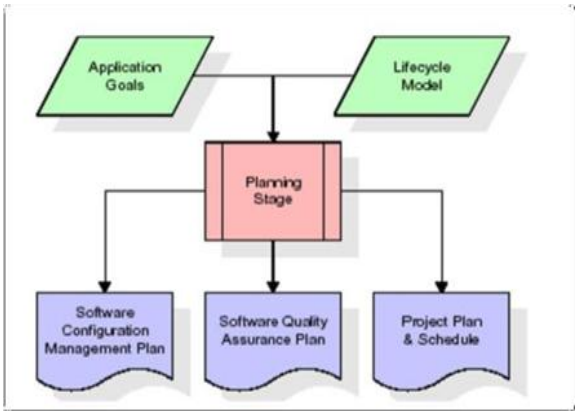
Requirements Gathering Stage:

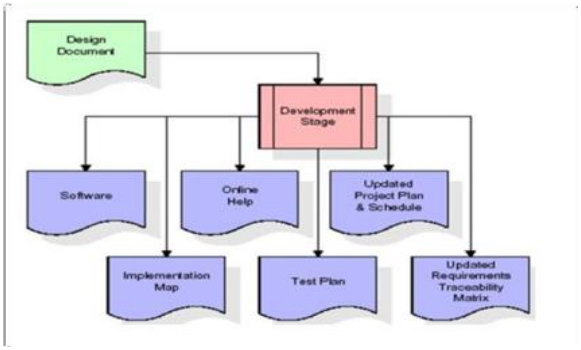
The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a

requirement title and textual description.
These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. The RTM shows that the product components developed during each stage of the SDLC are formally connected to the components developedinpriorstages.

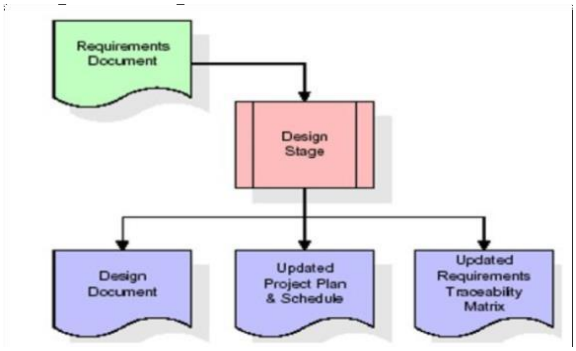


Analysis Stage:
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches. The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals.



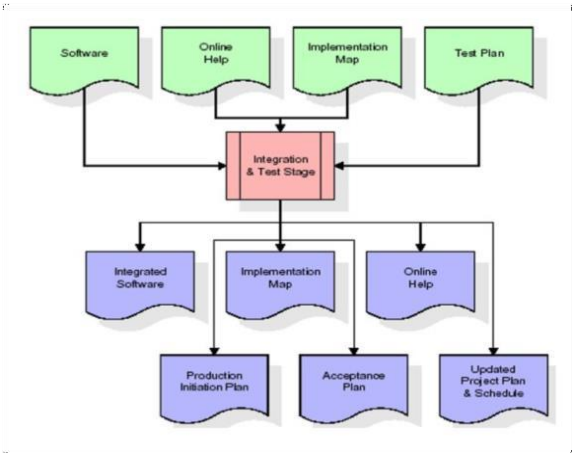


Designing Stage
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



Development (Coding) Stage:
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

Integration & Test Stage:
During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles.



Maintenance:
The outer rectangle represents maintenance of a project. The maintenance team will start with requirement study, understanding of documentation, after which employees will be assigned work and they will undergo training on their particular assigned category. For this life cycle there is no end it will be continued so on like an umbrella (no ending point to umbrella sticks).

3.4 Software Requirement Specification (SRS)
3.4.1 Overall Description
A Software Requirements Specification (SRS) is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).
Projects are subject to three sorts of requirements: Business requirements describe in business terms what must be delivered or accomplished to
provide value. Product requirements describe properties of a system or product. Process requirements describe activities performed by the developing organization.

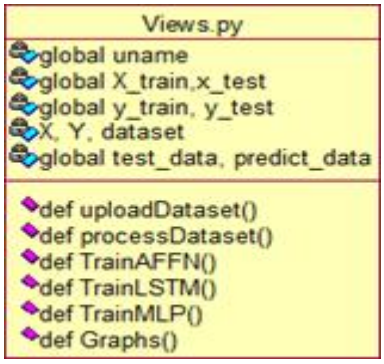
3.4.2 Feasibility Study
Economic Feasibility: A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible as it does not require any additional hardware or software.
Operational Feasibility: Proposed projects are beneficial only if they can be turned out into information systems that will meet the organization's operating requirements. This system is targeted to be in accordance with the above-mentioned issues. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.
Technical Feasibility: The current system developed is technically feasible. It is a user interface for audit workflow. Thus, it provides easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Therefore, it provides the technical guarantee of accuracy, reliability and security.

3.5 External Interface Requirements
User Interface: The user interface of this system is a user-friendly Python Graphical User
Interface (GUI).
Hardware Interfaces: The interaction between the user and the console is achieved through Python capabilities.
Software Interfaces: The required software is Python 3.7.0.

IV. SYSTEM DESIGN

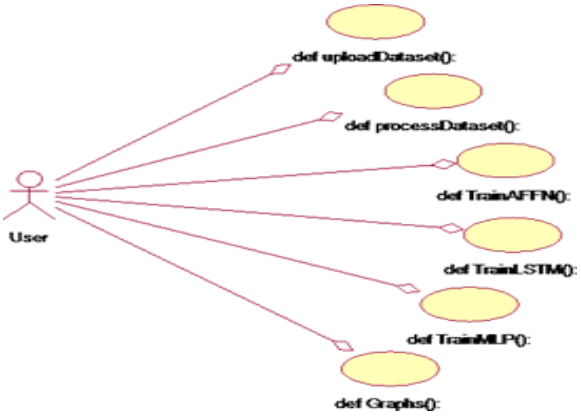
4.1 Class Diagram
The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the

models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects and interactions in the application as well as the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts: the upper part holds the name of the class; the middle part contains the attributes of the class; and the bottom part gives the methods or operations the class can take or undertake.



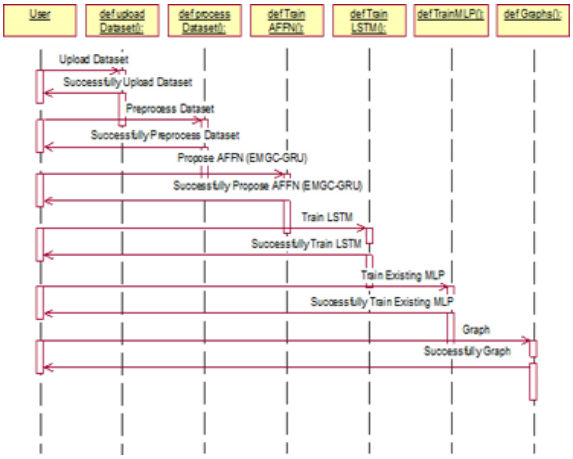
4.2 UseCaseDiagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicts the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams Aswell.



4.3 Sequence Diagram

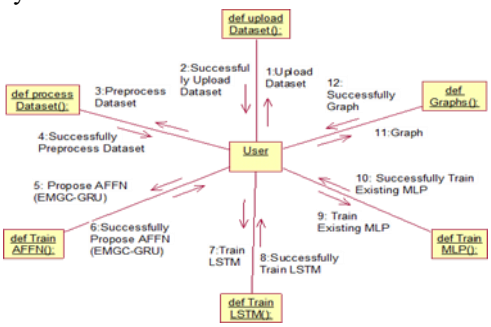
A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.4 Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing

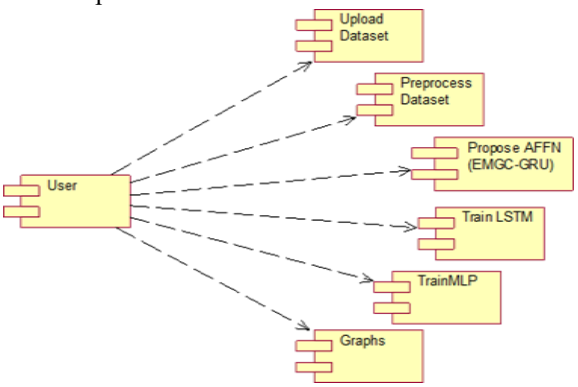
both the static structure and dynamic behaviour of a system.



4.5 Component Diagram

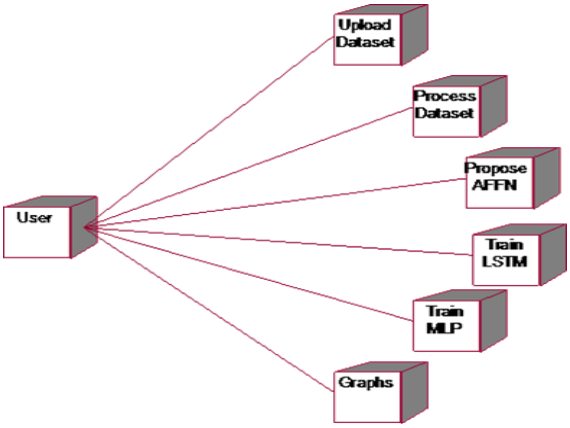
In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service

consumer — service provider relationship between the two components.



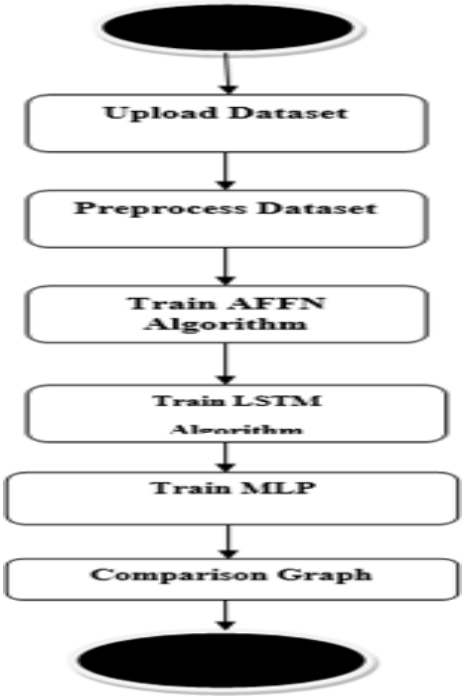
4.6 Deployment Diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



4.7 Activity Diagram

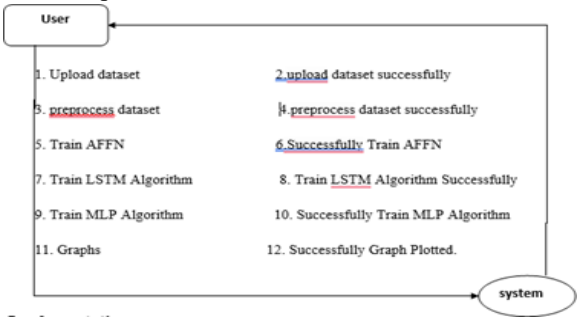
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.



4.8 Data Flow Diagram

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required.

As the name suggests, a Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. A DFD is a model for constructing and analyzing information processes it illustrates the flow of information in a process depending upon the inputs and outputs. A DFD demonstrates business or technical processes with the support of outside data saved, plus the data flowing from the process to another, and the end results.



V. IMPLEMENTATION

5.1 Python

Python is a general-purpose language. It has a wide range of applications from Web development (like Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user interfaces (Pygame, Panda3D). The syntax of the language is clean and the length of the code is relatively short. It is fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991. In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to the design of a new language which was later named Python — adopted from the comedy series "Monty Python's Flying Circus".

Key Features of Python:

- Simple and elegant syntax — easier to learn and read compared to languages like C++ or Java.
- Free and open-source — freely use and distribute Python, even for commercial use.
- Portability — Python programs can move from one platform to another and run without any changes on Windows, Mac OS X, and Linux.
- Extensible and Embeddable — can combine pieces of C/C++ or other languages with Python code for high performance.
- High-level, interpreted language — no need to worry about memory management or garbage collection.
- Large standard libraries — extensive libraries that make a programmer's life much easier.
- Object-oriented — everything in Python is an object, aiding in solving complex problems intuitively.

5.2 Installation

Python can be installed by downloading it from the official website and selecting the appropriate version for the operating system. After installation, required libraries and dependencies must be installed to run the project successfully. The installation process is simple and allows users to quickly set up a working development environment for executing Python programs and applications.

5.3 Python Features

Python offers several useful features such as easy syntax, interpreted execution, and portability across different systems. It is open-source and supports dynamic typing, which makes coding flexible and efficient. Python also provides extensive libraries that help in developing applications quickly. Its simplicity and versatility make it suitable for beginners as well as advanced developers working on complex systems.

5.4 Python GUI (Tkinter)

Tkinter is a standard Python library used for creating graphical user interfaces. It provides various tools and widgets to design interactive desktop applications easily. With Tkinter, developers can create windows, buttons, labels, and other interface components, making it simple to build user-friendly applications. It is widely used due to its simplicity and integration with Python’s standard library. .

5.6 Libraries

Python provides a wide range of libraries that simplify development by offering pre-written code for various tasks. Libraries such as NumPy and Pandas are used for data processing,

Matplotlib for visualization, and Scikit-learn, TensorFlow, and Keras for machine learning and deep learning applications. These libraries enhance performance, reduce development time, and make Python a powerful tool for data analysis and intelligent systems.

Algorithm:

The system uses deep learning algorithms such as LSTM and GRU for predicting passenger flow. LSTM is a type of recurrent neural network that is capable of learning long-term dependencies in sequential data using memory cells and gates like input, forget, and output gates. This helps in capturing temporal patterns effectively. GRU is a simplified version of LSTM that uses fewer gates, namely reset and update gates, to

control the flow of information, making it computationally efficient while still maintaining good performance. Both algorithms are well-suited for time-series prediction tasks and help improve the accuracy of passenger flow prediction by capturing complex temporal relationships in the data.

VI. TESTING

6.1 Implementation and Testing

Implementation is one of the most important tasks in the project. It is the phase in which one has to be cautious because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving a successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has been verified that these programs link together in the way specified in the program specification.

6.2 Testing Strategy

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system properly function as a unit. The test data should be chosen such that it passes through all possible conditions. Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commences.

Module Testing:

To locate errors, each module is tested individually. This enables us to detect errors and correct them without affecting any other modules. All the modules are individually tested from bottom up, starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately with different inputs and its approximate execution time is verified.

Integration Testing:

After module testing, the integration testing is applied. When linking the modules there may be chances for errors to occur; these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

Acceptance Testing:

When the user finds no major problems with accuracy, the system passes through a final acceptance test. This test confirms that the system meets the original goals, objectives and requirements established during analysis. Acceptance tests place the final verification on the shoulders of users and management it is finally acceptable and ready for operation.

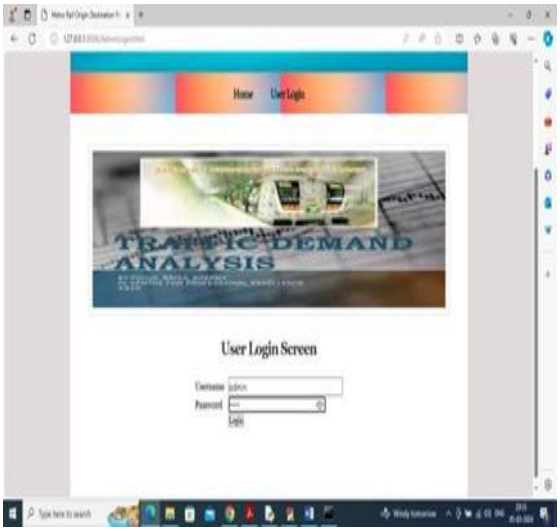
Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Upload Dataset	Verify Dataset is uploaded or not	If Dataset is not uploaded	we cannot do any further operations	we can do further operations	High	High
02	Preprocess Dataset	Verify Preprocess Dataset or not	If Dataset may not Preprocess	we cannot do any further operations	we can do further operations	High	High
03	Run ResNet50 Algorithm	Verify ResNet50 Algorithm runned or not	If ResNet50 Algorithm is not run	we cannot do any further operations	we can do further operations	High	High
04	Run Lenet Algorithm	Verify Lenet Algorithm runned or not	If Lenet Algorithm not Run	We cannot do further operations	We can do further Operations	High	High
05	Run Extension Lenet algorithm	Verify Extension Lenet Algorithm runned or not	If Extension Lenet algorithm not run	We cannot do further operations	We can do further Operations	High	High
06	Comparison Graph	Verify Comparison Graph obtained or not	If Comparison graph is not obtained	We cannot do further operations	We can do further Operations	High	High

VII. SCREENSHOTS

We have coded this DJANGO web framework so double click on run.bat file to start web server and get below page

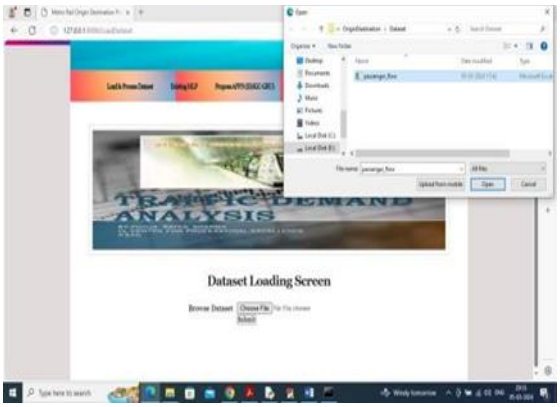
In above screen python web server started and now open browser and enter URL as http://127.0.0.1:8000/index.html and press enter key to get below page

In above screen click on 'User Login' link to get below page



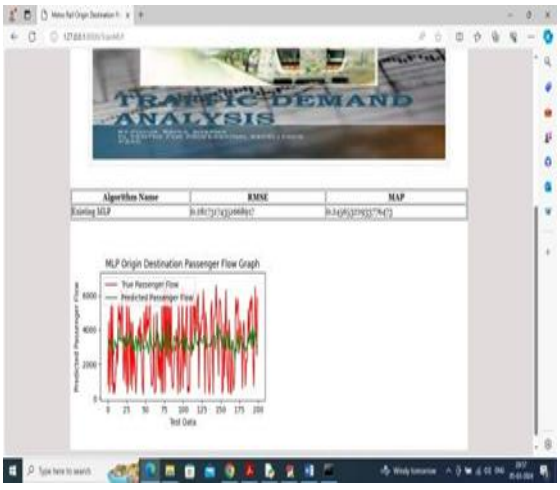
In above screen user is login and after login will get below page

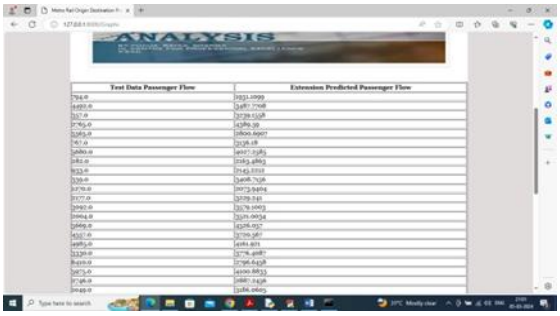
In above screen click on 'Load Process Dataset' link to get below page



In above screen selecting and uploading dataset and then click on 'Open' button to load dataset and get below page

In above screen dataset loaded and now click on 'Train Existing MLP' link to train existing algorithm and get below page



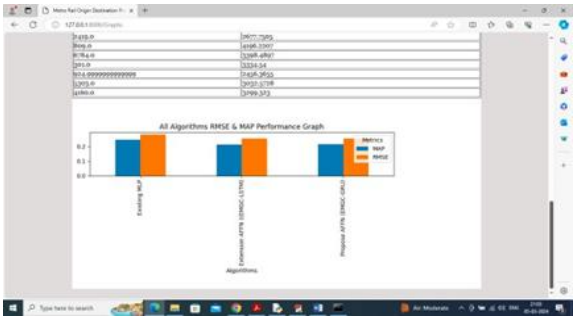


In above screen MLP existing algorithm training completed and can see RMSE and MAP values and in prediction graph x-axis represents Test data number and y-axis represents passenger flow and then red line represents True passenger flow and green line represents MLP predicted passenger flow and can see huge gap between both lines so we can say MLP is not accurate and now click on 'Propose AFFN (EMGC-GRU)' link to train propose algorithm and get below page

In above screen can see propose algorithm MAP and RMSE values which is lesser than existing algorithm and can see green and red line is overlapping with little gaps and now click on 'Extension AFFN (EMGC-LSTM)' link to train extension LSTM algorithm and get below page

In above screen in all 3 algorithms can see extension AFNN with LSTM got less MAP and RMSE values and can see little less gap in green and red lines and now click on 'Comparison Graph' link to get below graph

In above screen first column contains True passenger flow and second column contains extension passenger flow and can see some predictions are very close and now scroll down above page to view below graph



In above graph x-axis represents algorithm names and y-axis represents MAP and RMSE value in different color bars and in all algorithms, extension got less MAP and RMSE values so extension is better than all algorithms.

VIII. CONCLUSION

We proposed an Adaptive Feature Fusion Network (AFFN) to predict origin-destination passenger flow in a citywide metro system. To exhaustively capture the complex spatial and temporal dependencies in OD flows, we first developed an enhanced multi-graph convolution-gated recurrent unit (EMGC-GRU) that fuses the predefined correlations modelled by multiple knowledge-based graphs and the auto learned attention-based hidden correlations between stations within GRUs. An external factor-based attention module is then developed to accurately capture the periodic pattern by integrating the periodic data flow and external factors. To further improve prediction accuracy, we also proposed an asymmetric multi-task framework to predict OD flow and IO flow mutually. Evaluation results show that our proposed methods outperform the state-of-the-art spatial-temporal prediction techniques in terms of various prediction errors on two real-world metro trip datasets. Future works include 1) extending the one-step prediction model to a multi-step prediction model, 2) predicting more fine-grained passenger flow by fusing more detailed local trip information such as passenger movements and waiting time within the stations, collected from surveillance cameras or other sensors, 3) studying how our proposed model performs in more complex metro systems, such as those containing circular lines and multi-line shared track structures,

and 4) improving the prediction accuracy by fusing other non-metro trips such as bus and taxi trips.

REFERENCES

[1] X. Bao, "Urban rail transit present situation and future development trends in China: Overall analysis based on national policies and strategic plans in 2016–2020," *Urban Rail Transit*, vol. 4, no. 1, pp. 1–12, Mar. 2018.

[2] K. Alikhan, C. V. Narasimhulu, K. N. Reddy, and R. Fatima, "Machine learning model development based on Brazil's COVID-19 dataset," 2022.

[3] Z. Hou, H. Dong, S. Gao, G. Nicholson, L. Chen, and C. Roberts, "Energy-saving metro train timetable rescheduling model considering ATO profiles and dynamic passenger flow," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2774–2785, Jul. 2019.

[4] P. Zhang, Z. Sun, and X. Liu, "Optimized skip-stop metro line operation using smart card data," *J. Adv. Transp.*, vol. 2017, pp. 1–17, Jan. 2017.

[5] Y. Mei, W. Gu, M. Cassidy, and W. Fan, "Planning skip-stop transit service under heterogeneous demands," *Transp. Res. B, Methodol.*, vol. 150, pp. 503–523, Aug. 2021.

[6] Y. Liu, Z. Liu, and R. Jia, "DeepPF: A deep learning-based architecture for metro passenger flow prediction," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 18–34, Apr. 2019.

[7] J. Ye, J. Zhao, K. Ye, and C. Xu, "Multi-STGCNet: A graph convolution-based spatial–temporal framework for subway passenger flow forecasting," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.

[8] Y. Lu, H. Ding, S. Ji, N. Sze, and Z. He, "Dual attentive graph neural network for metro passenger flow prediction," *Neural Comput. Appl.*, vol. 33, pp. 13417–13431, Aug. 2021.

[9] Gao, L. Zheng, Z. Wang, X. Luo, C. Xie, and Y. Luo, "Attention-based short-term metro passenger flow prediction," in *Proc. Int. Conf. Knowl. Sci., Eng. Manage. *, New York, NY, USA: Springer, 2021, pp. 598–609.

[10] J. Zhang, F. Chen, Z. Cui, Y. Guo, and Y. Zhu, "Deep learning architecture for short-term passenger flow forecasting in urban rail transit," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7004–7014, Jul. 2021.

[11] M. Botte, C. Di Salvo, C. Caropreso, B. Montella, and L. D'Acierno, "Defining economic and environmental feasibility thresholds in the case of rail signalling systems based on satellite technology," in *Proc. IEEE 16th Int. Conf. Environ. Electr. Eng. (EEEIC)*, Jun. 2016, pp. 1–5.

[12] L. D'Acierno and M. Botte, "A passenger-oriented optimization model for implementing energy-saving strategies in railway contexts," *Energies*, vol. 11, no. 11, p. 2946, Oct. 2018.

[13] L. D'Acierno, M. Gallo, B. Montella, and A. Placido, "The definition of a model framework for managing rail systems in the case of breakdowns," in *Proc. IEEE 16th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 1059–1064.

[14] L. D'Acierno, A. Placido, M. Botte, and B. Montella, "A methodological approach for managing rail disruptions with different perspectives," *Int. J. Math. Models Methods Appl. Sci.*, vol. 10, pp. 80–86, Jan. 2016.

[15] F. Toqué, E. Come, M. K. El Mahrsi, and L. Oukhellou, "Forecasting dynamic public transport origin-destination matrices with long short-term memory recurrent neural networks," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1071–1076.

[16] L. Liu, J. Chen, H. Wu, J. Zhen, G. Li, and L. Lin, "Physical-virtual collaboration modeling for intra- and inter-station metro ridership prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 3377–3391, Apr. 2020.