C2SMART
CONNECTED CITIES WITH SMART TRANSPORTATION
A USDOT University Transportation Center

New York University
Rutgers University
University of Washington
University of Texas at El Paso
The City College of New York

An Artificial Intelligence Platform for Network-wide Congestion Detection and Prediction using Multi-Source Data

June 2019

C2SMART Center is a USDOT Tier 1 University Transportation Center taking on some of today's most pressing urban mobility challenges. Using cities as living laboratories, the center examines transportation problems and field tests novel solutions that draw on unprecedented recent advances in communication and smart technologies. Its research activities are focused on three key areas: Urban Mobility and Connected Citizens; Urban Analytics for Smart Cities; and Resilient, Secure and Smart Transportation Infrastructure.

Some of the key areas C2SMART is focusing on include:

Disruptive Technologies

We are developing innovative solutions that focus on emerging disruptive technologies and their impacts on transportation systems. Our aim is to accelerate technology transfer from the research phase to the real world.

Unconventional Big Data Applications

C2SMART is working to make it possible to safely share data from field tests and non-traditional sensing technologies so that decision-makers can address a wide range of urban mobility problems with the best information available to them.

Impactful Engagement

The center aims to overcome institutional barriers to innovation and hear and meet the needs of city and state stakeholders, including government agencies, policy makers, the private sector, non-profit organizations, and entrepreneurs.

Forward-thinking Training and Development

As an academic institution, we are dedicated to training the workforce of tomorrow to deal with new mobility problems in ways that are not covered in existing transportation curricula.

Led by the New York University Tandon School of Engineering, C2SMART is a consortium of five leading research universities, including Rutgers University, University of Washington, the University of Texas at El Paso, and The City College of New York.

c2smart.engineering.nyu.edu

An Artificial Intelligence Platform for Network-wide Congestion Detection and Prediction using Multi-Source Data

Yinhai Wang
University of Washington

Xuegang (Jeff) Ban
University of Washington

Zhiyong Cui
University of Washington

Meixin Zhu
University of Washington

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

Acknowledgements

The project team appreciates the financial and administrative support from the C2SMART UTC. The team also thanks WSDOT for providing the research datasets. The team is grateful to the helpful guidance to conduct the data management plan and the insightful comments about the project via webinars with C2SMART partners. The UW STAR Lab members, especially Mr. Mingjian Fu, also offered much assistance to the development of the transportation AI platform, which is greatly appreciated.

Executive Summary

The advancement of new smart traffic sensing, mobile communication, and artificial intelligence technologies has greatly stimulated the growth of transportation data. A great deal of generated transportation data is playing an important role in modern smart transportation and smart city research and applications. The increase of computation power enabled by advanced hardware and the rise of artificial intelligence (AI) technologies, especially in the deep learning field, provides great opportunities to comprehensively utilize the transportation big data. When applying AI in the transportation area, transportation domain knowledge is beneficial for designing AI models and solving transportation problems in a smarter way. However, because most AI algorithms were not originally designed for transportation problems, using big data and AI technologies to solve transportation problems is facing challenges. Since key hyper-parameters are missed in some proposed AI models and the software and hardware adopted by various studies are different, many proposed AI-based methods can hardly be accurately re-implemented. Further, in most of the AI-based transportation research studies, there is no uniform dataset to evaluate the proposed models. Thus, to overcome the challenges mentioned earlier, this project seeks to build a transportation AI platform with widely accepted datasets, provide well-established models, and use uniform training and testing procedures to assist the evaluation of emerging novel methodologies. Traffic forecasting involving high-dimensional spatiotemporal data is a good applicable scenario to utilize novel deep learning models to solve complicated transportation problems. Thus, in this project, the prototype platform mainly focuses on solving the traffic prediction problem. More specifically, the major contributions of this project include:

- Develop a prototypical artificial intelligence platform for solving challenging transportation problems, which require large-volume high-dimensional transportation data and complex models. This AI platform is capable of providing standardized datasets and novel deep learning-based models for specific problems.
- Design a novel architecture for the transportation AI platform to enhance the efficiency of the transportation data processing, management, and communication and increase the computational power of the platform.
- Design a data storage and management schema to manage multiple network-wide traffic data sets for supporting the traffic prediction task and simplify the whole training and testing process.
- Develop multiple deep learning-based models for solving the network-wide traffic prediction problem, which can be the baseline models to evaluate novel deep learning-based models.

The developed transportation AI platform is capable of evaluating the traffic prediction performance of various implemented models by comparing and visualizing the prediction results tested on multiple real-world network-wide traffic state data sets. Future work will focus on increasing the computation capability of the platform and broadening the topics that the platform can deal with.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

1.1.1. Transportation Data Science

The advancement of new smart traffic sensing, mobile communication, and artificial intelligence technologies has stimulated significant growth in the volume and variety of transportation data. A huge volume of newly generated transportation data is playing an important role in modern smart transportation and smart city research and applications. Although transportation data has great potential to enhance smart transportation applications in the development of smart cities, we are still facing challenges regarding how to fully and properly use these various massive transportation data sets. In recent years, new transportation sensing technologies have constantly emerged, which provide more options to transportation practitioners and researchers to collect necessary transportation data. The transportation data not only can be collected by traditional traffic sensors, but it can also be generated by transportation-related applications and tools, such as social media, cell phones, mobile apps, electric vehicles, cyclists, pedestrians, etc. Thus, due to the diversity and variety of transportation data, both transportation practitioners and researchers are facing substantial opportunities and challenges.

Before the "big data" term was widely used, most of the previous transportation applications and studies were designed and conducted based on real data. However, real transportation problems are normally intricate and related to many unexpected influential factors. The sizes of the datasets used in previous studies are usually too small to reflect the real-world complexity of these problems. Meanwhile, many methodologies in the transportation field are built based on complicated mathematical models without comprehensive validation from real transportation datasets. Thus, biased models may be generated in the process of understanding the core of transportation problems. Since the term "big data" has become more commonly used, transportation data science, or the use of immense data sets to investigate transportation issues, has become a more prevalent approach to smart transportation research and development. The newly generated transportation data inherently has the core "5v" properties of big data, i.e. volume, velocity, variety, value, and veracity [1]. Thus, it is critical to find proper and practicable ways to comprehensively utilize it.

In recent years, the increased computation power enabled by advanced hardware and the rise of artificial intelligence (AI) technologies, especially in the deep learning field, have provided great opportunities to comprehensively utilize transportation big data. AI technologies have been widely adopted for many applications requiring large datasets in multiple research and industrial fields, such as computer vision, natural language processing, and robotics. The complex deep neural network models with powerful non-linear fitting capabilities can provide great power for dealing with complicated transportation problems that cannot be solved by classical methods. However, the deep learning-based models are normally highly

flexible, and thus, the designs of these models need to be customized depending on specific problems and datasets. For a specific problem, the design of a model structure will directly affect the model performance. In addition, although transportation is a proper scientific domain for developing and applying novel deep learning models, most of the existing deep learning models are not originally designed for transportation problems. Hence, developing novel or customized deep learning models for classical transportation problems is another challenge for transportation practitioners and researchers.

To efficiently overcome these challenges, developing multiple deep learning models and comparing their performance by testing on existing standard datasets can stimulate the emergence of new methodologies. With this idea, this project aims to build a platform with standard procedures to train and test different deep learning models to solve various transportation-related problems. The testing results can be easily compared to assist in selecting the more effective models for further studies or implementations.

1.1.2. Transportation AI Applications

Artificial intelligence (AI) refers to a broad field in computer science that enables machines to think and act like human brains [2]. With increasing populations, vehicles, and mobility demands, improving the safety, efficiency, and sustainability of the transportation system is becoming an urgent task for transportation practitioners and researchers. With the recent development of accelerated computation power, big data, and machine learning algorithms, various AI methods have achieved state-of-art performance in speech recognition, visual object recognition, object detection, and many other domains. When applying AI in the transportation domain, transportation domain knowledge is beneficial for designing AI models and solving transportation problems in a smarter way. Because traffic congestion detection and prediction requires taking network traffic states into consideration, traffic prediction using a huge volume of historical traffic state data is a great field for applying novel AI algorithms. In this project, we focus on using AI technologies to solve the traffic prediction problem, including developing and evaluating AI-based models, via a data-driven transportation AI platform.

Since urban traffic flow is complex and constantly changing, it is difficult for travelers to acquire information describing current and estimated future traffic conditions for various roadway facilities. As a result, congestion prediction was proposed to support transportation agencies and help them establish effective traffic management measures, as well as aid road users in their adoption of smarter trip strategies, including route and departure time selection [3]. Ultimately, there are two major challenges in urban traffic congestion detection and prediction: (1) How to estimate and predict traffic state in large-scale urban areas? (2) How to improve the accuracy, instantaneous nature, and stability of traffic congestion detection and prediction?

With the advancement of data collection technologies, transportation data has become more and more ubiquitous. This triggered a series of data-driven research projects to investigate transportation

phenomena. Some recent studies have proposed data-driven methods for traffic congestion detection and prediction. Several traffic congestion prediction methods have also been developed, such as adaptive data-driven real-time congestion prediction [3], traffic flow prediction using floating car trajectory data [4], Bayesian network analysis [5], deep learning theory [6], data mining based approaches (integration of K-means clustering, decision trees, and neural networks) [7], hierarchical fuzzy rule-based systems optimized with genetic algorithms [8], etc. These existing studies have made significant contributions to the development of methodologies and technologies for traffic congestion detection and prediction, but with the development of Intelligent Transportation Systems (ITS) and Internet of Things (IoT) technologies, new challenges and opportunities are continuously emerging with higher requirements for metrics such as detection and prediction accuracy, real-time results, and stability.

Recently, artificial intelligence (AI) has been considered one of the most promising techniques to tackle tremendously high-dimensional data analysis tasks. AI technologies have been applied for transportation analysis applications such as traffic signal control, autonomous driving, pedestrian crossing detection, travel time prediction, short term traffic volume prediction, and car ownership determinants [9]. Although the applications of AI technologies are still in the early stage in the transportation area, deep learning-based traffic prediction is becoming a fairly popular research field. In this project, multiple novel deep learning-based traffic prediction models are implemented and evaluated to enhance the development of new algorithms.

1.1.3. Transportation AI Platforms

Even though transportation data has been broadly collected and archived, data accessibility and usability are unsatisfactory [10]. AI-based traffic prediction requiring immense historical traffic state data is facing several challenges, including 1) there are no uniform datasets to evaluate the new proposed models and 2) most of the proposed models can hardly be re-implemented because key hyper-parameters are missing and different software and hardware are used by various studies. To solve these problems, this project seeks to build a transportation AI platform with standard datasets and provide well-established models to assist in evaluating emerging novel methods.

Since various transportation datasets are stored separately and managed independently by different agencies, extensive communication, data formatting, and data integration efforts are required to make the data accessible and interpretable to the users. Therefore, a large number of transportation data storing, analysis, and visualization platforms and advanced traveler information systems have been developed in attempts to overcome such barriers. The PI's research team has already established an online transportation platform, named the Digital Roadway Interactive Visualization and Evaluation Network (DRIVE Net) [10][11], whose development is funded by Washington State Department of Transportation (WSDOT). DRIVE Net can be used for sharing, integration, visualization, and analysis of transportation-related data. This project aims to extend the functions of DRIVE Net by developing an AI-

based platform for network-wide congestion detection and prediction using multi-source data. The designed AI platform incorporates novel deep learning models to support network-wide analysis for identifying solutions for traffic forecasting. Compared to existing transportation data analysis platforms, the transportation AI platform has superior computational capability and flexibility. The new AI platform will provide more convenient and efficient traffic analysis tools for transportation agencies, researchers, and practitioners.

1.2. Objective

Artificial intelligence and immense transportation data offer the potential to significantly improve the efficiency and robustness of solving modern transportation problems. Traffic forecasting requiring high-dimensional spatiotemporal data is an appropriate scenario to utilize novel deep learning models. The goal of this project is to develop an AI-based transportation platform to stimulate and enhance the design of novel transportation-oriented deep learning algorithms. The transportation AI platform can provide standard network-wide traffic state datasets and implement existing state-of-the-art algorithms as the baselines to evaluate the novel algorithms.

In particular, we aim to achieve the following research objectives:

1. Develop an artificial intelligence platform for solving challenging transportation problems that require large-volume high-dimensional transportation data and complex models. This AI platform is capable of providing standardized datasets and novel deep learning-based models for specific problems. In this project, the prototype platform mainly focuses on solving the traffic prediction problem.
2. Design a novel architecture for the transportation AI platform to enhance the efficiency of transportation data processing, management, and communication and increase the computational power of the platform.
3. Design a data storage and management schema to manage multiple network-wide traffic data sets for supporting the traffic prediction task and simplifying the model's training and testing process.
4. Develop multiple deep learning-based models for solving the network-wide traffic prediction problem, which can be the baseline models to evaluate novel deep learning-based models.
5. Develop a transportation AI platform capable of evaluating the traffic prediction performance of various implemented models by comparing and visualizing the prediction results tested on multiple real-world network-wide traffic state data sets.

2. Literature Review

Data science is a set of fundamental principles that support and guide the principled extraction of information and knowledge from data [12]. The core task of data science is to extract knowledge from data via technologies that incorporate these principles. Accordingly, transportation data science can be

realized by applying the fundamental principles of data science in the transportation field. Specifically, transportation data science can be defined as the computationally intensive investigation of transportation issues using immense data sets. Artificial intelligence, especially deep learning methods, applied to the analysis of emerging transportation datasets, has brought new power to the transportation research field. To adequately bring the power of AI to the transportation fields, this project aims to build a transportation AI platform to stimulate and enhance the design of novel transportation-oriented deep learning algorithms. In this section, a number of AI-based transportation applications and platforms are introduced. As a typical complicated transportation problem, traffic prediction studies that can comprehensively incorporate deep learning methods are also introduced.

2.1.Artificial Intelligence Applied in Transportation

Artificial intelligence (AI) has the potential to solve problems that are hard for traditional methods to address, and various AI methods have achieved state-of-the-art performances in speech recognition, visual object recognition, object detection and many other domains [13]. Some AI-based methods even surpass human-level performance on some specific problems [14]. With increasing population, vehicles, and mobility demands, improving the safety, efficiency, and sustainability of the transportation system remains a challenge, and traditional methods may not be able to fully address it. To overcome these issues, an increasing amount of studies have been conducted to apply AI-based methods to solve complicated transportation problems including traffic signal control, traffic prediction, microscopic traffic modeling, and autonomous driving. In this section, a brief review of recent studies that utilize AI-based methods is presented.

For traffic signal control, reinforcement learning (RL), as a significant branch of deep learning methods, has been adequately studied and widely applied in recent years. Abdulhai et al. [15] proposed a Q-learning based traffic signal control method for an isolated traffic intersection with variable traffic demands. Arel et al. [16] proposed a traffic signal control policy based on multi-agent RL. A five-intersection traffic network was studied and each intersection was controlled by an RL agent. By combining deep neural network and RL, Li et al. [17] used a deep RL method for traffic signal control and claimed that it outperforms conventional traffic control methods. Leveraging large-scale real-world traffic data from surveillance cameras, Wei et al. [18] built a deep Q-learning model for signal controlling with 24 traffic intersections.

Microscopic traffic models including car-following and lane-changing models are the fundamental parts of transportation research. Recently, many emerging deep learning-based technologies have been applied to microscopic traffic modeling. Wang et al. [19] proposed a data-driven car-following model based on the gated recurrent unit (GRU) neural networks, followed by Zhou et al. [20] who used a general recurrent neural network (RNN) for car following modeling. Zhu et al. [21] built a deep RL based car-following model using the deep deterministic policy gradient (DDPG) algorithm. Further, the convolution neural network

(CNN) was also used for lane change intention prediction [22]. In addition, a deep RL model was proposed for high-level lane-changing decision making [23].

For autonomous driving perception, several CNN-based deep learning methods, such as YOLO [24], Faster R-CNN [25], and Mask R-CNN [26], have been the most widely used for object detection and recognition in 2D camera images. For 3D object detection with Lidar data, Chen et al. [27] proposed a multi-view 3D network that predicts orientated 3D bounding boxes base on Lidar point cloud and RGB images. Zhou and Tuzel [28] developed a generic 3D object detection network called VoxelNet that combines feature extraction and bounding box prediction into a single stage network. For autonomous driving decision making, CNN and RL approaches have been studied. Bojarski et al. [29] developed an end-to-end learning approach that uses CNN to predict steering commands directly from front camera image pixels. By integrating a fully-convolutional network (FCN) and a long short-term memory (LSTM) network, predicting discrete and continuous driving behaviors from a large-scale driving video dataset is fulfilled [30]. Moreover, Kendall et al. [31] demonstrated that using deep RL methods for autonomous driving in real-world situations is possible, where the vehicle can learn to drive on real roads in a single day.

2.2. Existing Transportation Data/AI Platforms

Well-designed platforms or systems are capable of properly utilizing the existing immense transportation data sets and AI methods. In this section, a brief review of existing transportation data/AI platforms for typical transportation problems, including traffic signal control, traffic congestion detection, and autonomous driving, is presented.

For traffic signal control, IBM has been granted a patent - titled " Cognitive traffic signal control " - for a real-time traffic management system powered by AI [32]. The patent describes a traffic management system where a computer receives a streaming video for one or more paths of traffic and pedestrians at an intersection. Based on the analysis of the real-time flow of traffic, the computer processor(s) would then determine the best way to manage the traffic signals. For real-time traffic signal control, an innovative approach, Surtrac [33], is proposed by combining artificial intelligence methods with traffic theory. The Surtrac is capable of optimizing traffic signals to control the traffic flow on both urban grids and corridors, leading to less waiting time, reduced congestion, shorter trips, and less pollution. Further, the DeepDrive platform[34] is developed to provide adaptive traffic signal control based on deep reinforcement learning.

For traffic congestion detection and traffic prediction, PTV Optima [35] can generate traffic prediction information for up to 60 minutes in the future. Traffic congestion can be detected by the speed and traffic flow detected in the field or calculated from roadway traffic states data (e.g., floating car and license plate identification data). The detected traffic congestion has spatiotemporal impacts on the surrounding traffic. PTV Optima's model-based approach enables the analysis based on special features, such as

unpredictable events and traffic accidents. The Miovision TrafficLink platform [36] is developed to assist traffic engineers to create more responsive and efficient traffic networks. TIMON [37] is a European research project, whose main objective is to provide real-time services through a web-based platform and a mobile application for drivers. To provide these services, one of the core technologies developed inside TIMON is the design and development of AI techniques for traffic prediction and route planning. Microsoft research team pioneered the use of machine learning methods to build predictive models for traffic [38]. The developed models can infer and predict traffic flow at different time periods in the future based on the analysis of large amounts of data over months and years. For fulfilling the predictive insights and proactive traffic management optimization, Waycare [39] is shaping the future of city mobility by enabling cities to take full control of their roads by harnessing in-vehicle information and municipal traffic data. Aimsun [40] is a leader in traffic prediction software and services. Aimsun has fully integrated software packages to complement the ITS portfolio by simulating future traffic flows to aid offline strategic transportation planning and real-time mobility management. In addition, a transportation data storage, management, and visualization platform, named Digital Roadway Interactive Visualization and Evaluation Network (DRIVE Net), was developed to enable large-scale online data sharing, visualization, modeling and analysis [10]. In addition to those mentioned above, The UrbanLogiq platform [41] aggregates diverse data sets such as traffic counts, weather, infrastructure, mobile, and accidents so that cites can understand movement and congestion. Moreover, companies providing mobility services and map-based navigation services, such as Alibaba, Baidu, and Didi Chuxing, have all come up with artificial intelligence solutions that bring data from government and other partners to develop a city traffic management powered by AI and cloud technology [42]. For example, the Didi Chuxing company announced the full opening of AI technology, services, computing power and diversified accumulated solutions by releasing an open platform “Qunan” [43].

In the autonomous driving field, there are a variety of technical paths for autopilot, such as camera-based solutions (Tesla, AutoX), LIDAR-based solutions (Waymo, Baidu), and multi-sensor deep integration solutions (Drive.ai, Zoox). Drive.ai [44] uses artificial intelligence to create self-driving transportation solutions that improve the state of mobility today. NVIDIA also invests heavily in the NVIDIA DRIVE PX autopilot development platform [45]. Combined with deep learning, sensor fusion, and vision technologies, the environmental changes around the vehicle can be recognized in real time and the vehicle can accurately locate itself on a high-definition map or even plan a safe route ahead. Tesla is developing a new generation of autonomous driving hardware that includes an artificial intelligence processor-Autopilot [46]. Waymo [47] announced that the company began testing autonomous vehicles without a safety driver in the driver's seat and carried passengers in Phoenix. Baidu Apollo system is the world's first and most comprehensive intelligent driving commercial solution [48]. Comma.ai [49] has launched an open source project called Openpilot, which includes a complete set of accessibility programs for unmanned cars.

2.3. Existing Deep Learning-based Traffic Forecasting

To demonstrate the computation capability of the transportation AI platform, a training and testing procedure for traffic prediction models is developed. Numerous traffic state/congestion prediction methods have also been developed such as adaptive data-driven real-time congestion prediction [3], traffic flow prediction using floating car trajectory data [4], Bayesian network analysis [5], deep learning theory [6], data mining based approaches (integration of K-means clustering, decision trees, and neural networks) [7], hierarchical fuzzy rule-based systems optimized with genetic algorithms [8], etc.

However, to summarize the previous research on traffic prediction, those existing models are roughly classified into two categories: statistical methods and machine learning models [50][51][52][53]. Most of the statistical methods for traffic forecasting were developed to deal with traffic roadways with less complex network structures. Meanwhile, the transportation datasets used in those statistical methods were relatively small in size. Thus, the capability of such statistical methods to handle high-dimensional and dynamic time series data is quite limited. With the development of data mining and knowledge discovery, much recent work on this topic focuses on machine learning methods for traffic forecasting.

Because machine learning methods are normally capable of capturing complex non-linear relationships in high-dimensional data, many machine learning-based methods, like support vector regression (SVR) [54], tend to outperform the statistical methods, such as autoregressive integrated moving average (ARIMA) [55] and its many variants [56]. However, the potential of artificial intelligence approaches to handle complex traffic forecasting problems was not fully investigated until recent years. Following early works [50][57] applying NNs to the traffic prediction problem, deep learning models have shown their superior abilities to capture nonlinear spatiotemporal effects for traffic forecasting [58].

Deep learning based-models turn out to be very effective for solving complex traffic forecasting problems. Since a preliminary study [59] utilized the feed-forward NN to estimate vehicle travel time, many NN-based models, including fuzzy NN [60], recurrent NN [57], convolution NN [53][61], deep belief networks [62][63], auto-encoders [64][65], generative adversarial networks [66][67], and combinations of these models have been applied to forecast traffic states. Because traffic state datasets are mostly made up of spatiotemporal data, the recurrent NN with the capability of capturing temporal dependencies and its variants, such as LSTM [68] and GRU [69], were widely used as a component of a traffic forecasting model to forecast traffic speed [56], travel time [70], and traffic flow [71]. These existing studies have made significant contributions to the development of the methodologies and technologies for traffic congestion detection and prediction, but with the development of Intelligent Transportation Systems (ITS), new challenges and opportunities are continuously emerging with higher requirements for metrics such as detection and prediction accuracy, real-time results, and stability.

3. Architecture

The artificial intelligence-based platform is designed to solve multiple cutting-edge transportation problems, and thus, it should have the ability to support hosting multiple types of models and datasets to solve multiple tasks. In this project, a task incorporated by the transportation AI platform refers to a specific transportation problem, such as traffic forecasting, data imputation, and vehicle detection, whose performance can be quantitatively evaluated by well-established metrics. For example, given a standard dataset, the traffic forecasting problems can be evaluated by accuracy or robustness-related metrics. Although this project mainly focuses on the network-wide traffic forecasting problem, the flexible storage and management of models and dataset are taken into consideration when the architecture of the transportation AI platform is designed and developed.

3.1. Platform Architecture

The transportation AI platform is built based on transportation-related datasets, and thus there should be a data management system to store, query, process, and manage all the datasets to make the modeling process efficient. Because the artificial intelligence methods are mostly neural network-based models and the training and testing of neural networks are quite time-consuming, a computer or a cluster, or even a cloud with the power of conducting multiple training and testing tasks, should be incorporated as a main component of the transportation AI platform. In addition, to let the users conveniently and interactively manipulate the models and datasets on the platform, the platform should also have a user-friendly interface. Thus, to fulfill the aforementioned requirements, the transportation AI platform mainly contains three main components, i.e. a data warehouse, a web server, and a computation center, as shown in Figure 1. All three components are connected and the communications between those components are mainly data transmission. The data warehouse hosting multiple types of databases can manage and provide datasets for solving transportation problems. The computation center uses computers or the cloud which contains multiple graphics processing units (GPUs) capable of efficiently training and testing deep learning models. The web server hosts a website that allows users to access the

developed transportation AI platform remotely and help manage user accounts and training and testing tasks.

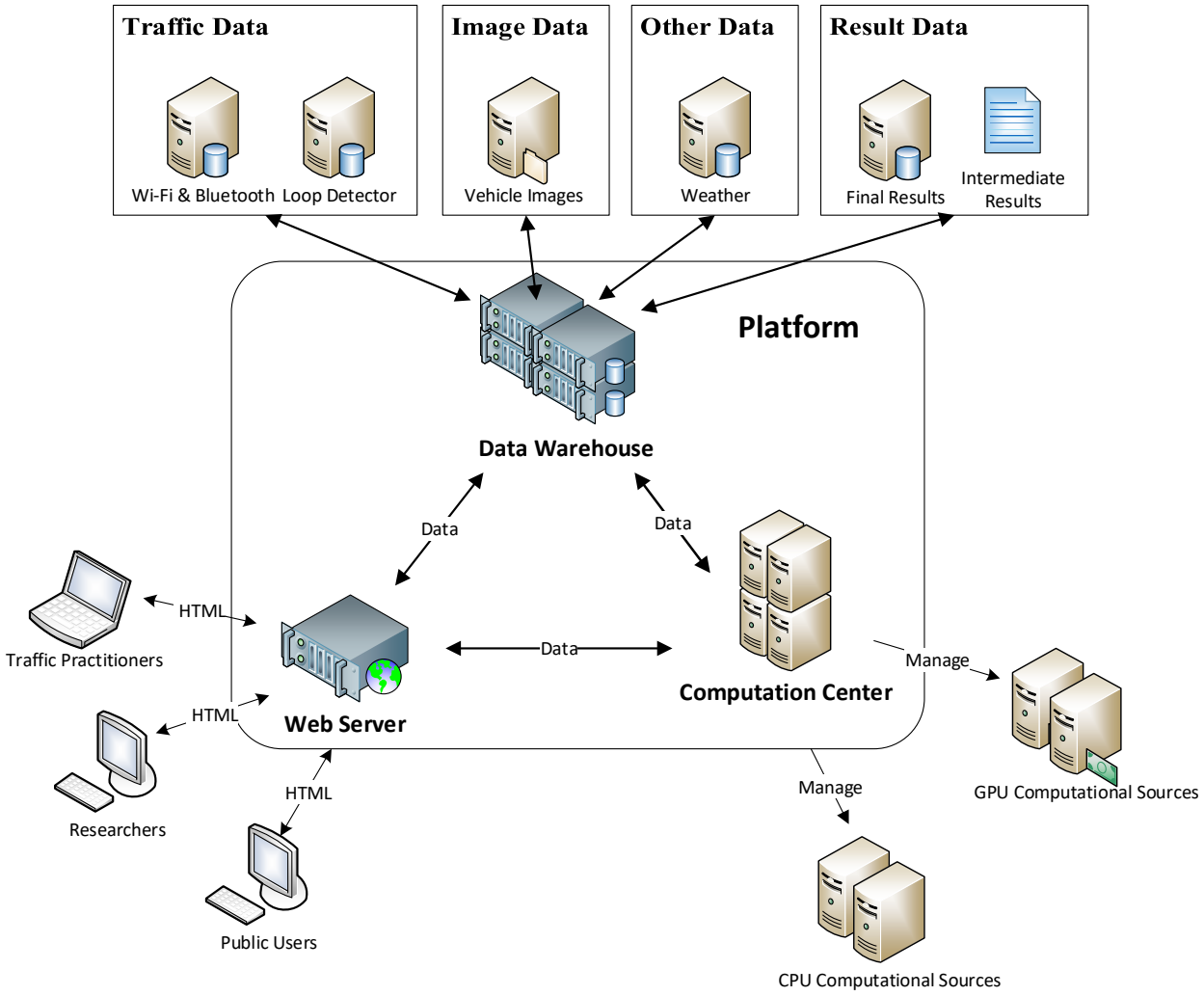

Figure 1 : Architecture of the Transportation AI Platform

A detailed description of all three components is included in the following subsections. The data transmission between those components are also introduced.

3.2. Data Management and Database Design

The data warehouse is capable of hosting multiple types of data, as shown in Figure 1, including tabular data, image data, and structured data. The traffic state data, mainly collected from loop detector sensors and other types of traffic sensors, are mostly stored in relational databases in the tabular format. The

image or video data normally should be traffic monitoring data for solving problems, such as traffic flow counting or traffic detection. The data describing the traffic network's physical or topological structures are normally stored using various types of databases or files, depending on the types of tasks that need to be solved. Basically, the aforementioned data is used for training and testing models to solve specific transportation problems. However, when a model is trained or tested on the platform, many types of metadata and result data are generated and should also be stored.

For a specific task, the data used for the training and testing process should be identical when different users attempt to adjust hyper-parameters or conduct the training/testing process multiple times. Hence, in this project, the formats of the datasets used for training/testing models are fixed and those datasets are stored as files that can be read using the same data loading procedure. As the training/testing processes are conducted in the computation center, to reduce data communication and efficiently load data to the models, all the well-processed training and testing data are stored as files in the computation center. However, since different users may conduct different tasks and the results of these tasks will be reused and visualized by the transportation AI platform, the query process of these result data should be convenient and flexible. Hence, the task information and task result data are stored in the database in the data warehouse side.

Figure 2 shows the database schema of the task-related tables on the transportation AI platform. The tables are as followings:

- Users: stores the information of each user of the platform. The primary key is UserID (id).
- Goals: stores all the specific task (goal), including traffic prediction, vehicle detection, etc. The primary key is goalID.
- Datasets: stores the detailed information of all datasets and the path to locate the dataset files. The primary is datasetID.
- Models: stores the detailed information of all models. The primary key is modelID.
- TrainingTask: stores the detailed information of historical training tasks, consisting of userID, goalID, datasetID, modelID, and some other parameters. The primary key is TrainingTaskID.
- Training Result: stores the results of each historical training tasks, consisting of training loss, validation loss, and timestamp. The primary key is the combination of TrainingTaskID and Epoch.
- GPU: stores the detailed information of GPUs in the computation center. The primary key is GPUID.

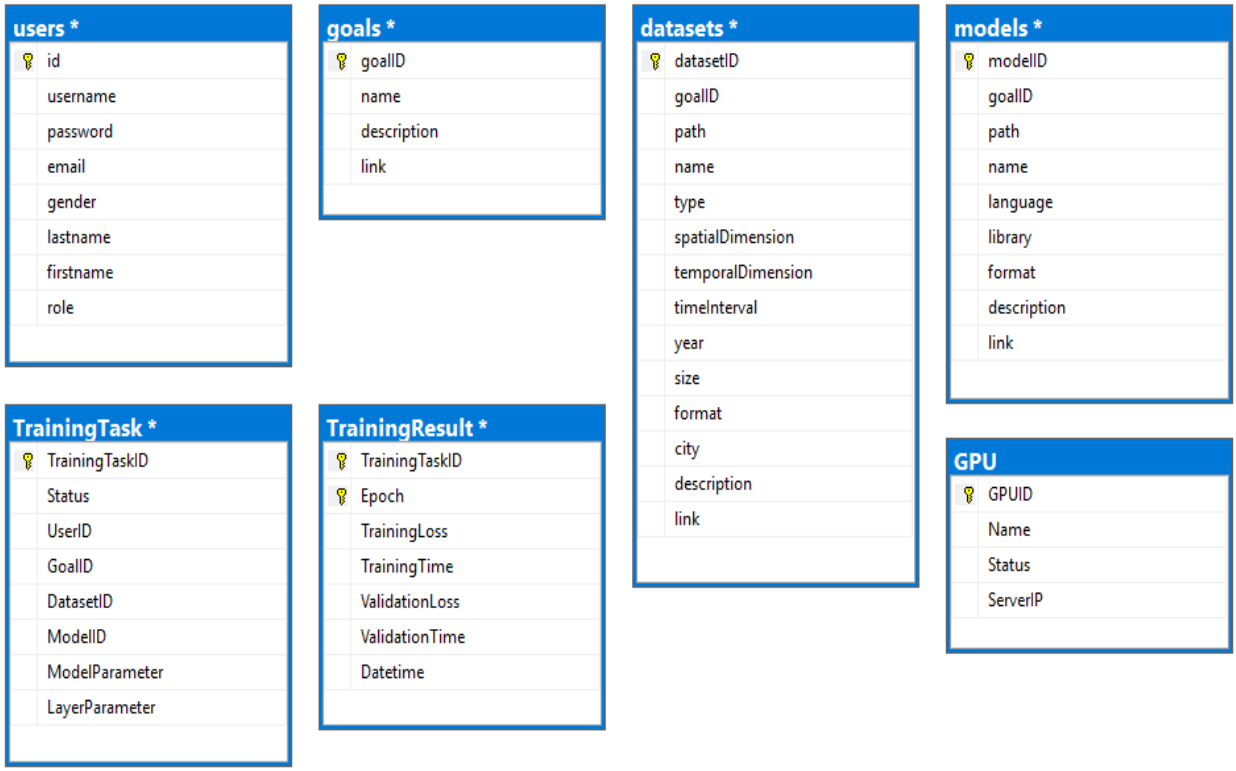

Figure 2 : Database design of main tables which are used for storing, training, and testing process information

Given the main detailed information of these tables, the relationship between them, i.e. the database schema, is briefly shown in Figure 3. The TrainingTask is the core entity that connects the User, Goal, Model, and Dataset entities. The intermediate training results are stored in the TrainingResult table. The trained model is saved as a file name by the TrainingTaskID and stored in the computation center. With the help of the stored paths in the dataset table, the datasets can be easily located in the computation center.

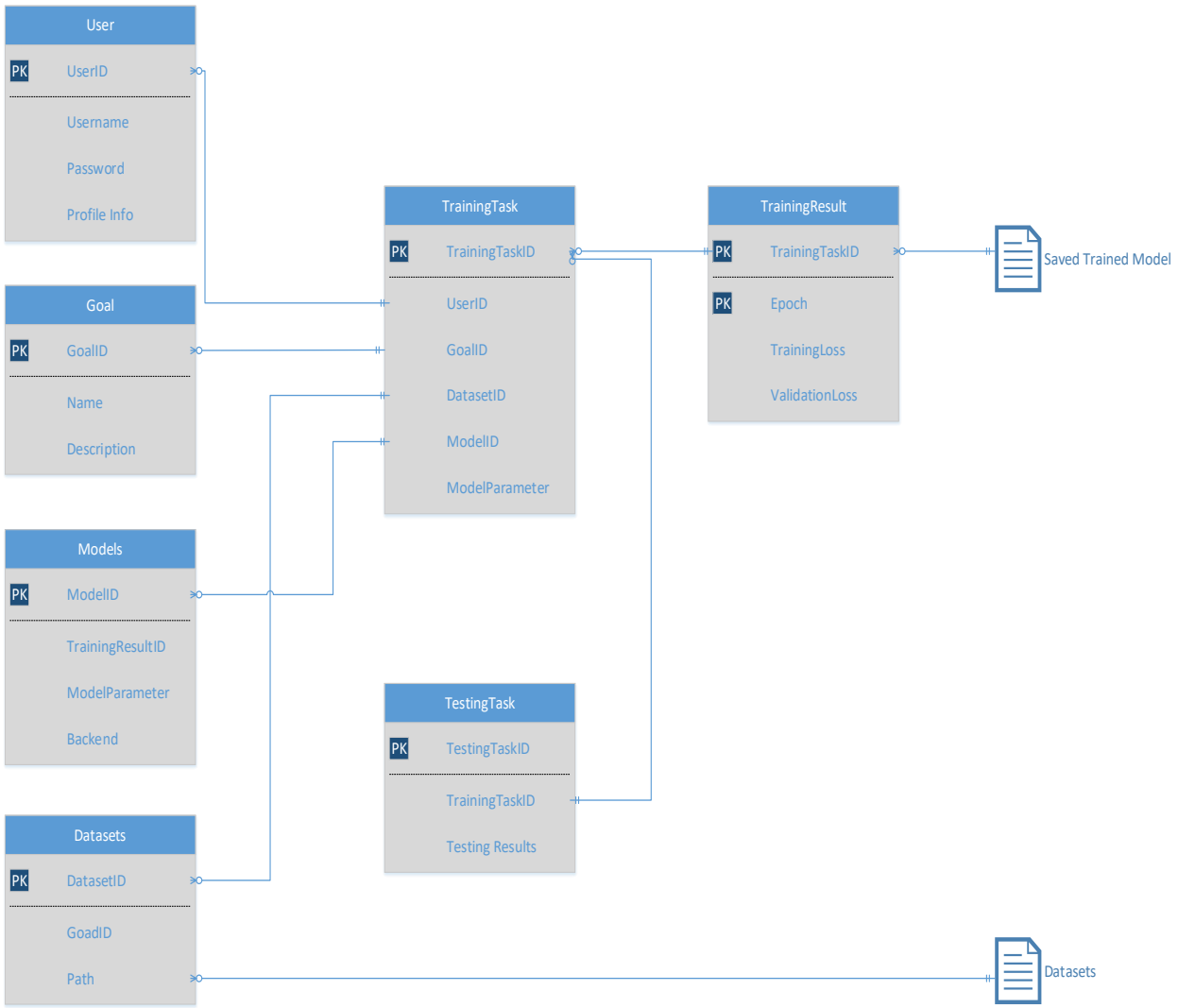

Figure 3 : A brief introduction of the database schema for the transportation AI platform tasks

3.3.Computation Center

The computation center has the ability to receive task requests from the web server and execute the specific task. The task results will be sent to the data warehouse to be permanently stored. Since multiple users may use the platform at the same time, multiple task requests may arrive at the computation center at the same time or in a short period of time. It is necessary for the computation center to be able to manage all the received tasks and execute those tasks in a reasonable order based on the amount of computation resources the computation center has. To reasonably manage all the received tasks, the computation center is built based on several important components, including a task manager, a thread manager, a dataset pool, a model pool, and a computation resource pool, as shown in Figure 4.

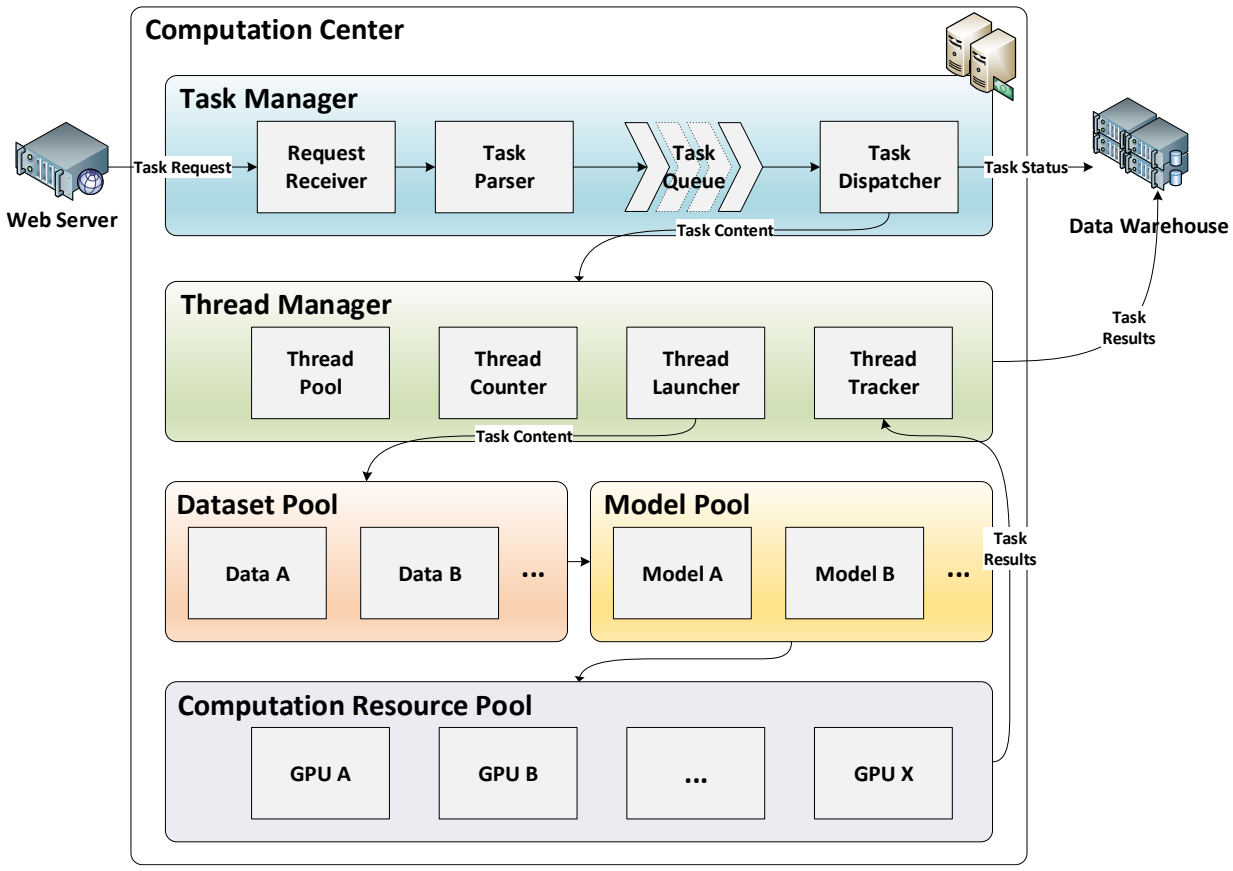

Figure 4 : Structure of the computation center

The task manager is the core component of the computation center, which consists of a request receiver, a task parser, a task queue, and a task dispatcher. The request receiver acts as a server that monitors all the task requests via the http request technologies. When a task request is received, the encoded context of the task request will be decoded by the task parser. After the task content is successfully decoded without errors, the task request will be input into a task queue, which will attempt to first launch the earliest-arrived task request. The main reason to add a task queue in the task manager is that when the number of requested tasks is larger than the amount of available GPUs on the platform, the task requests that arrived last need to be stored in the task queue. When a GPU is available to be used to train or test deep learning models, the first task will be removed from the task queue and passed to the task dispatcher. The task dispatcher will input the task content to the thread manager to execute the task by creating a new thread.

The thread manager manages the threads to optimize the distribution of computation resources among the tasks. The thread manager contains a thread pool, thread counter, thread launcher, and a thread tracker. The thread manager has a maximum amount of threads in the thread pool and the thread counter is used to count how many threads are operating. The maximum number of threads is based on the

amount of available computation resources. After a thread is created, the thread launcher will start to load the correct dataset and the correct model from the dataset pool and the model pool, respectively, to execute the task. When the thread is created, the thread launcher will also assign a GPU from the computation resource pool to the task. The dataset will be loaded to the memory and the model will be initialized in the assigned GPU to start the training process. At the same time, the thread tracker will monitor the GPU's status to help dispatch tasks.

While the task is executing, the intermediate results and final outputs will be sent back to and stored in the data warehouse. In the end, via the thread manager and the task manager, the status of the computation resource pool will be updated to assist with the arrangement of upcoming tasks.

3.4. Web Server

The web server is the interface that connects the platform and users. Most web applications are designed based on the Model-View-Controller (MVC) framework. The architecture of the transportation AI platform is also similar to the MVC framework which contains the view, controller and models. However, since the transportation AI platform has the computation center, the web server can be simplified as the computation center is responsible for the modeling and computation jobs. Thus, the web server mainly serves for the view and controller parts.

The UI of the transportation AI platform is designed to be simple and clear. Thus, the whole UI of the platform is designed as a system administration console providing all the required functions directly on the web page. The logical level of the transportation AI platform is controlled based on the procedure for efficiently and easily creating a new task on the platform. The procedure for creating new tasks is introduced in detail in the following section.

3.5. New Task Procedure

Based on the introduced data warehouse, computation center, and web server, the procedure for starting a new task is introduced in this section. The transportation AI platform mainly provides two types of tasks. One is starting a new training task and the other is checking the historical training results. Figure 5 displays the creating new task procedures. When the users, including traffic practitioners, researcher and public users, create a new task, the web server will let the users select one of the task types supported by the transportation AI platform. Based on the selected task, the platform UI will let the users select the dataset hosted in the computation center. Although the datasets are stored in the computation center to reduce the data communication between the data warehouse and the computation center, the descriptions of those datasets are stored in the database. Thus, during the data selection process, the data selection module needs to request data from the data warehouse. Then, the users will come to the model selection and parameter configuration module, during which this module still needs to query descriptive data from

the database. After that, the information for creating a new task is completed and will be sent to the computation center to start the new task.

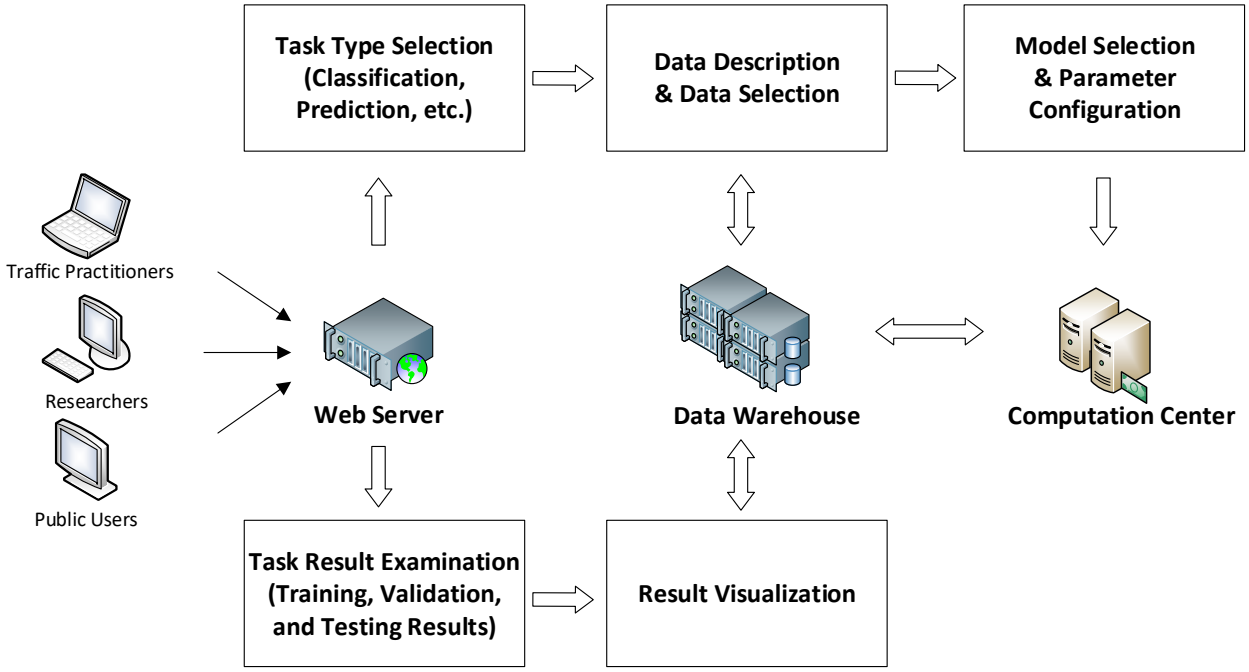

Figure 5 : Creating new task procedures

The other type of task is historical or real-time task result examination. During this process, the web server does not need to request any data from the computation center because all the intermediate and final results are stored in the data warehouse. The task result examination process mainly provides result visualization functions, especially for the training and validation loss. Since the training and validation loss of each epoch can intuitively show how well a model is trained, the visualization of the training and validation loss data is critical for the transportation AI platform. The details of the implementation technologies and tools are introduced in the platform development section.

4. Methodology

The transportation AI platform can be used to address various transportation problems with the help of the novel deep learning models. Since this project mainly focuses on the traffic forecasting problems, this section mainly introduces the novel deep learning-based traffic forecasting algorithms.

4.1.Notions

To define the traffic forecasting problems, several ideas are introduced in this subsection. Traffic states have many properties, including traffic speed, travel time, and traffic flow. In this section, the traffic speed is used as an example of the traffic states to introduce the deep learning algorithms used in this project.

For a single location with traffic sensors, the collected traffic states can be represented by sequences and those sequences with n historical time steps can be represented by a vector,

$$X_T = [x_1, x_2, \dots, x_t, \dots, x_T] \tag{1}$$

For a long roadway or a network-wide traffic network with multiple traffic sensing areas, the traffic state data is normally represented by two-dimensional (2D) spatial-temporal data. A traffic network that contains P sensing areas/locations, with the traffic state data at time T using n historical time frames (steps) can be characterized as a matrix:

$$X_T^P = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^P \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{T-1}^1 & x_T^1 \\ x_1^2 & x_2^2 & \ddots & x_{T-1}^2 & x_T^2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_1^P & x_2^P & \cdots & x_{T-1}^P & x_T^P \end{bmatrix} \tag{2}$$

where each element x_t^p represents the speed of the t-th time frame at the p-th location. To reflect the temporal attributes of the speed data and simplify the expressions of the equations in the following subsections, the speed matrix is represented by a vector, $X_T = [x_1, x_2, \dots, x_T] \in \mathbb{R}^{P \times T}$, in which each element is a vector of the P locations' speed values.

Traffic forecasting refers to predicting future traffic states, such as traffic speed, travel time, or volume, given previously observed traffic states from a roadway network. Specifically, the traffic forecasting problem aims to learn a function $F(\cdot)$ to map T time steps of historical traffic states, i.e. $\boldsymbol{X}_T^P$, map to the traffic states in the subsequent one or more time steps. In this project, the function attempts to forecast the traffic states in the subsequent one step, i.e. x_{T+1}, and the formulation of $F(\cdot)$ is defined as

$$F\left(\left[x_{T-n}, x_{T-(n-1)}, \dots, x_{T-2}, x_{T-1}\right]\right) = x_{T+1} \tag{4}$$

4.2.Recurrent Neural Network

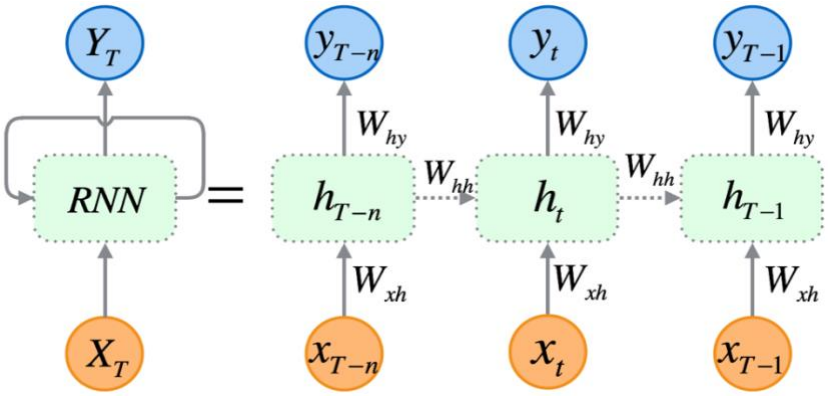

Figure 6 : Standard RNN architecture and an unfolded structure with T time steps[1]

Recurrent neural network is a type of powerful deep neural network using an internal memory with loops to deal with sequence data. The architecture of RNNs is illustrated in Figure 6.The hidden layer in a RNN receives the input vector $\boldsymbol{X}_T^P$ and generates the output vector $\boldsymbol{Y}_T$. The unfolded structure of RNNs, shown in the right part of Figure 6, illustrates the calculation process that, at each time iteration t, the hidden layer maintains a hidden state h_t and updates it based on the layer input x_t. The previous hidden state h_{t-1} is also used during the update process using the following equation:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{5}$$

where W_{xh} is the weight matrix from the input layer to the hidden layer, W_{hh} is the weight matrix for the hidden states, and b_h is the bias vector of the hidden layer. The σ_h is the activation function to generate the hidden state, which is normally a sigmoid function. Then, the network output at time t can be characterized as:

$$y_t = \sigma_y(W_{hy}h_t + b_y) \tag{6}$$

where W_{hy} is the weight matrix from the hidden layer to the output layer, b_y is the bias vector of the output layer and σ_y is the activation function of the output layer. By applying Equation (5) and Equation (6), the parameters in the weight matrices and bias vectors is trained and updated iteratively via the back-propagation (BP) method. In each time step t, the hidden layer will generate a value y_t and the last output y_T is the desired predicted speed in the next time step, namely $\hat{x}_{T+1} = y_T$.

[1] Image source [93]

Although RNNs exhibit superior capabilities for modeling non-linear sequence data, regular RNNs suffer from the vanishing or blowing-up gradient during the BP process. Thus, it is difficult for RNNs to capture long-term dependencies and RNNs is not very effective for learning sequence data with long time lags.

4.3. Long Short-Term Memory Network

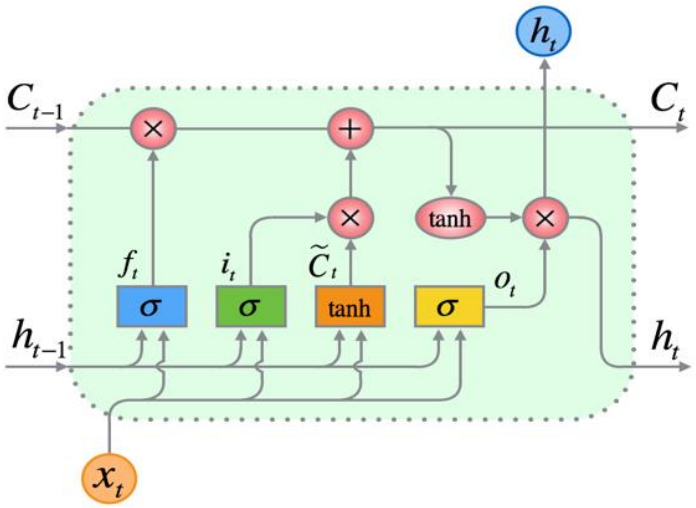

Figure 7 : LSTM architecture. The pink circles are arithmetic operators and the colored rectangles are the gates in LSTM.[2]

To handle the aforementioned gradient vanishing or blowing-up problems of RNNs, several sophisticated recurrent architectures, like LSTM architecture [72] and Gated Recurrent Unit (GRU) architecture [73], are proposed. The LSTM neural network works well on sequence-based tasks with long-term dependencies, but GRU, which is a simplified LSTM architecture, is widely used in the domain of machine translation and natural language processing. Although there have been a variety of typical LSTM variants proposed in recent years, a large-scale analysis of LSTM variants shows that the LSTM variants can significantly improve the performance upon the standard LSTM architecture[74]. Thus, the LSTM neural network is selected as an optional traffic prediction method in the transportation AI platform in this project and introduced in this section.

The main difference between standard LSTM architecture and RNN architecture is in the hidden layer. The hidden layer of LSTM, i.e. the LSTM cell, is shown in Figure 7. Similar to RNNs, at time t, the LSTM cell takes x_t as the input and generates h_t as the output. However, LSTM also maintains another hidden state, which is called cell state and denoted as C_t. Hence, the cell state C_{t-1} at the previous time step $t-1$ is

[2] Image source [93]

also input to the LSTM cell, just like the previous hidden state h_{t-1}. Due to the gated structure, LSTM can handle long-term dependencies to allow comprehensive information to go through the LSTM cell along loop structure. There are three gates in an LSTM cell, including an input gate, a forget gate, and an output gate. The gated structure, especially the forget gate, helps LSTM effectively deal with sequential data learning problems [74] in a scalable way. At time t, the input gate i_t, the forget gate f_t, and the output gate o_t, and the input cell state $\tilde{\mathrm{C}}_\mathrm{t}$ are represented by the colorful boxes in the LSTM cell in Figure 7 and calculated using the following equations:

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{7}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{9}$$

$$\tilde{C}_t = tanh(W_C x_t + U_C h_{t-1} + b_C) \tag{10}$$

where W_f, W_i, W_o, and W_C are the weight matrices of the three gates and the input cell state in the LSTM cell and the U_f, U_i, U_o, and U_C are the weight matrices connecting the previous cell output state to the three gates and the input cell state. The b_f, b_i, b_o, and b_C are four bias vectors. The σ_g is the gate activation function, which is normally the sigmoid function, and the $\tanh$ is the hyperbolic tangent function. Based on the results of four above equations, at each time iteration t, the cell output state, C_t, and the layer output, h_t, can be calculated as follows:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{11}$$

$$h_t = o_t \odot \tanh(C_t) \tag{12}$$

Where $\odot$ is an element-wise multiplication operator. The final output of an LSTM layer should be a vector of all the outputs, represented by $\boldsymbol{Y}_T = [h_1, \ldots, h_T]$. If the traffic prediction problem targets to forecast the traffic state at the next time step $\hat{x}_{T+1}$, only the last element of the output vector h_T the desired predicted value, i.e. $\hat{x}_{T+1} = h_T$.

4.4.Gated Recurrent Unit Network

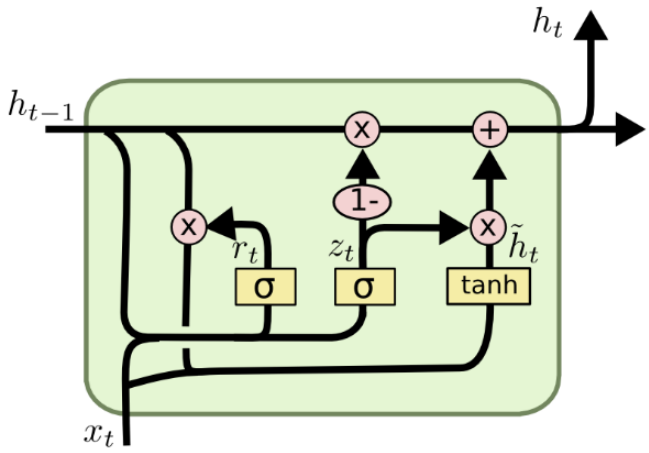

Figure 8 : GRU structure.[3]

A gated recurrent unit (GRU) was proposed by Cho et al. [75] to make each recurrent unit adaptively capture dependencies of different time scales. Similar to the LSTM cell, the GRU has gated units that modulate the flow of information inside the gated units. However, the GRU does not have separate memory cells. The structure of the GRU is illustrated in Figure 8. The GRU contains two gates, an update gate z_t and a reset gate r_t, which can be described as following:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{13}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{14}$$

where W_z and W_r are the input weight matrices and U_z and U_r are the hidden state weight matrices. A candidate hidden state $\tilde{h}_t$ is defined to update the hidden state, which is described as follows:

$$\tilde{h}_t = \tanh\big(W x_t + U(r_t \odot h_{t-1})\big) \tag{15}$$

When r_t is close to zero, i.e. the reset gate is off, the reset gate effectively makes the unit act as if it is reading the first symbol of an input sequence, allowing it to forget the previously computed state [73]. Then, the hidden state h_t is updated based on the update gate z_t, defined as follows:

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t \tag{16}$$

[3] Image Source: https://towardsdatascience.com/what-is-a-recurrent-nns-and-gated-recurrent-unit-grus-ea71d2a05a69

The hidden state h_t of the GRU at time t is a linear interpolation between the previous hidden state h_{t-1} and the candidate hidden state $\tilde{h}_t$. Similar to LSTM, when taking the X_T as input, the last output of GRU h_T is the desired predicted value.

4.5.Graph Wavelet Gated Recurrent Network

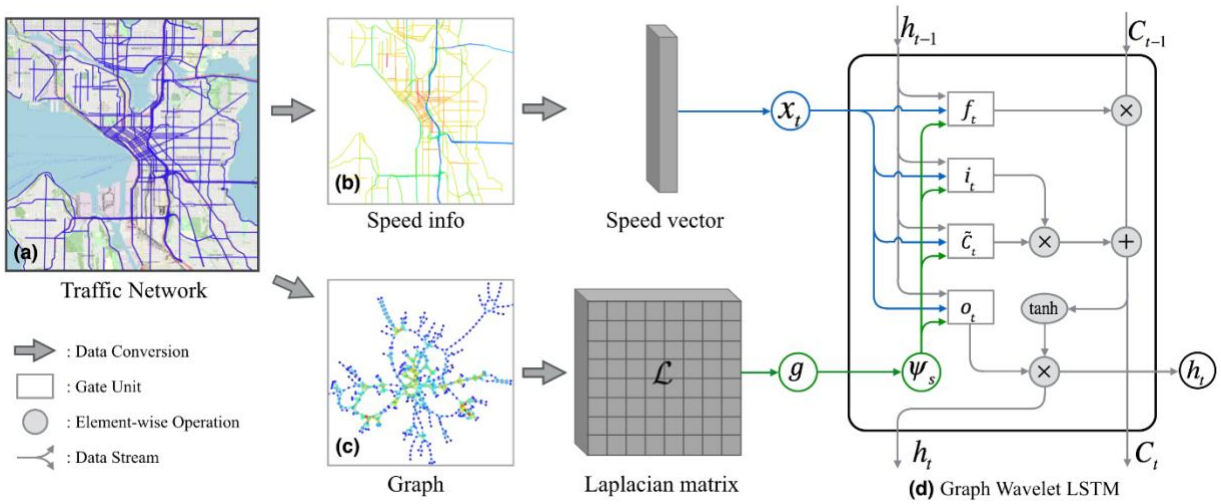

Figure 9 : Demonstration of the graph wavelet gated recurrent network. (a) Urban traffic network in downtown Seattle. (b) Speed information of roadway segments illustrated by various colors. (c) Graph structure converted from the traffic network. (d) Structure of a graph wavelet LSTM unit at time t, in which g is the kernel function and Ψ_s is the graph wavelet matrix.

To extract the spatial-temporal features from the traffic network, a more efficient way that considers the roadway network as a graph is proposed. The roadway network-based graph consists of vertices and edges representing sensing locations and connecting links, respectively. The graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ consisting of N vertices $v_j \in \mathcal{V}$ and their linking edges $(v_i, v_j) \in \mathcal{E}$. The adjacency matrix $A \in \mathbb{R}^{N\times N}$ describes the connectedness of vertices, in which element $A_{i,j} = A_{j,i} = 1$ if vertices i and j are connected, otherwise $A_{i,j} = 0$ ($A_{i,i} = 0$). The degree matrix of the graph $D \in \mathbb{R}^{N\times N}$, which is a diagonal matrix used to describe how many edges are attached to each vertex , is defined as $D_{i,i} = \sum_{j=1}^{N} A_{i,j}$. The connectivity of the graph vertices can also be encoded by the graph Laplacian matrix $\mathcal{L}$, which is the basis of spectral graph analysis. The combinatorial form of the graph Laplacian matrix can be defined as $\mathcal{L} = D - A$ and the normalized form is defined as $\mathcal{L} = I_N - D^{-1/2}AD^{-1/2}$, where $I_N \in \mathbb{R}^{N\times N}$ is the identity matrix. Because $\mathcal{L}$ is a symmetric positive semidefinite matrix, the eigen-decomposition of $\mathcal{L}$ can be described as $\mathcal{L} = U\Lambda U^T$, in which $U \in \mathbb{R}^{N\times N}$ is the eigenvector matrix. The columns U is a set of

eigenvectors that $U = [u_0, u_1, \dots, u_{N-1}]$. The corresponding eigenvalues can be represented by $\lambda_0, \lambda_1, \dots, \lambda_{N-1}$ such that $\mathcal{L}u_i = \lambda_i u_i$. Hence, the diagonal eigenvalue matrix is denoted as $\Lambda = \text{diag}(\lambda_0, \lambda_1, \dots, \lambda_{N-1}) \in \mathbb{R}^{N \times N}$.

Based on the graph wavelet theory [76], the graph wavelet coefficients of all the vertices can be defined as

$$\Psi_s = U G_s U^T \tag{17}$$

where U is the matrix formed by Laplacian eigenvectors and $G_s = \text{diag}\big(g(s\lambda_0), \dots, g(s\lambda_{N-1})\big)$ is a diagonal kernel matrix. $g(\cdot)$ is a kernel functions and s is a scaling parameter that can be assigned as any positive real value.

To capture the complex spatial-temporal dependencies in network-wide traffic data, we learn the traffic network as a graph and propose a graph wavelet gated recurrent (GWGR) neural network. The GWGR is similar to LSTM in that it has several gate units to filter out or add information to the cell state. However, the gate units in GWGR are defined based on the graph wavelet matrix Ψ_s. The framework of the proposed model is shown in Figure 9 (d). Figure 9 (a) shows the traffic network. At time t, the roadway on the traffic network has different speed values, which is shown in Figure 9 (b). The speed values of all the roadways are converted into a vector x_t as the input of the model. The roadway network structure is converted into a graph, as shown in Figure 9 (c). The graph wavelet coefficient is fixed for all time steps and it is designed based on the Laplacian matrix and a kernel function.

In the proposed GWGR model, the graph wavelet coefficient matrix Ψ_s is defined in Equation (17). The heat kernel $g(s\lambda_i) = e^{-s\lambda_i}$ is adopted to generate the graph wavelet coefficient. Then, the Ψ_s^{-1} is easily obtained by replacing $g(s\lambda_i)$ with $g(-s\lambda_i)$ [76]. The basic structure of the GWGR can be defined by the following equations:

$$f_t = \sigma_g\big(\Psi_s \Lambda_f^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_f^h \Psi_s^{-1} h_{t-1} + b_f\big) \tag{18}$$
$$i_t = \sigma_g\big(\Psi_s \Lambda_i^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_i^h \Psi_s^{-1} h_{t-1} + b_i\big) \tag{19}$$
$$o_t = \sigma_g\big(\Psi_s \Lambda_o^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_o^h \Psi_s^{-1} h_{t-1} + b_o\big) \tag{20}$$
$$\tilde{C}_t = \tanh\big(\Psi_s \Lambda_C^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_C^h \Psi_s^{-1} h_{t-1} + b_C\big) \tag{21}$$

where f_t, i_t, o_t and $\tilde{C}_t \in \mathbb{R}^N$ are the outputs of the forget gate, input gate, output gate and the input memory cell. Λ_f^x, Λ_i^x, Λ_o^x, and $\Lambda_C^x \in \mathbb{R}^{N \times N}$ are diagonal weight matrices that filter the input x_t to the three gates and the memory cell with the help of graph wavelet matrix. Similarly, Λ_f^h, Λ_i^h, Λ_o^h, and $\Lambda_C^h \in \mathbb{R}^{N \times N}$ are also diagonal weight matrices for the hidden state h_t. b_f, b_i, b_o, and $b_C \in \mathbb{R}^N$ are four bias weight vectors. The σ_g is the sigmoid activation function and tanh is the hyperbolic tangent function. Then, the cell state C_t and the hidden state h_t at time t are calculated as following

$$C_t = \mathrm{f}_t \odot C_{t-1} + \mathrm{i}_t \odot \tilde{C}_t \quad (22)$$

$$h_t = \mathrm{o}_t \odot \tanh(C_t) \quad (23)$$

The $h_t \in \mathbb{R}^N$ is also the output of the GWGR unit at time t. Given the input sequence $X_T = [x_1, x_1, \dots, x_T] \in \mathbb{R}^{T \times N}$, the predicted value of the future step is $\hat{x}_{T+1} = h_T$. If we only need to predict traffic data for one future step, the loss function of the model can be defined as

$$Loss = \mathrm{Loss}(\hat{x}_{T+1} - x_{T+1}) = \mathrm{Loss}(h_T - x_{T+1}) \quad (24)$$

where $\mathrm{Loss}(\cdot)$ is the loss function, normally adopting the mean square error function of the traffic prediction problem.

5. Dataset

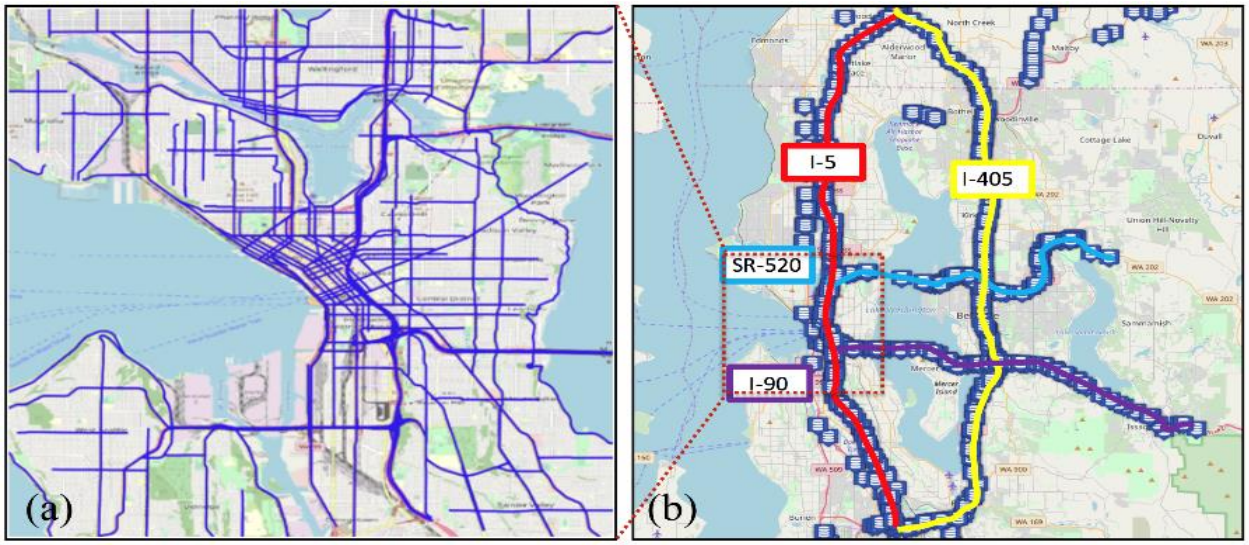

Figure 10 : (a) Urban traffic dataset covering the downtown Seattle urban corridors. (b) Freeway traffic data covering freeway system in Seattle area.

5.1. Datasets for Traffic Prediction

The artificial intelligence platform has the potential to provide datasets and deep learning-based models to multiple transportation related problems. In this project, the main target is to develop the prototype platform and implement the traffic prediction functionalities by providing network-wide traffic datasets and traffic prediction models. Hence, the datasets used in this project are the network-wide traffic datasets, including the inductive loop detector dataset and the National Performance Management Research Data Set (NPMRDS) dataset. The artificial intelligence platform is also scalable, with the potential to host more types of traffic datasets for fulfilling the traffic prediction problem and other problems. In this section, the two datasets for the traffic forecasting problem are introduced.

NPMRDS data: This dataset provided by WSDOT originates from the Federal Highway Administration (FHWA) 's National Performance Management Research Data Set [77]. The NPMRDS data contains the average travel times on the National Highway System for use in its performance measures and management activities. This data set is also available to State Departments of Transportation (DOTs) and Metropolitan Planning Organizations to use for assisting their performance management activities. From 2014 to 2016, the NPMRDS data provider is the Here Company. From 2017, the IRNIX travel time data is selected as the NPMRDS data for the USDOT and state DOTs to measure freeway roadway performance. In this project, the INRIX data in 2012 is used as one of the data sets. The INRIX data contains the travel times on roadway segments collected by and integrated from the probe vehicle data and traffic sensing data.

This data set contains the speed data of roadway links in the Seattle downtown area, which is mostly collected by probe vehicles. In this area, the road network is very complex in that it contains principal arterials, minor arterials, one-way streets, freeways, ramps, express lanes, etc. This dataset covers the year 2012 and the time interval is 5-minute. The roadway network contains more than 1000 roadway links, but we select the largest connected roadway network containing 745 segments in the experiment, i.e. $N = 745$, as shown in Figure 10 (a). For confidentiality reasons, this dataset is not allowed to be published at this stage.

Loop detector dataset [78]: The second data set is the loop detector data set which is collected by inductive loop detectors deployed on the freeway system. In this data set, the loop detector data covers four connected freeways in the Greater Seattle areas, including I-5, I-90, I-405, and SR-520, as shown in Figure 10 (b). The raw data contains three basic traffic flow characteristics, including traffic speed, volume, and density. The time interval in the raw data is 20 seconds. To solve the missing data problem, a flexible and robust data imputation method is applied [79]. After the dataset was comprehensively checked and cleaned [80], only the high-quality speed information from 2015 is used in this project. 323 traffic sensing locations are selected and integrated to form the loop detector data set. The time interval is integrated into 5-minute intervals. In this way, the loop detector speed data set is well-formatted without missing values. This dataset is published at https://doi.org/10.5281/zenodo.3258904, according to the guidelines in the C2SMART data management plan.

5.2. Data Formatting

For a specific task on the transportation AI platform, no matter which model and which dataset are used, the data input to the models should be in the same formats for establishing an automatic task executing process on the platform. Thus, although two datasets are used for the traffic prediction task, the two datasets are uniformly formatted to support the following modeling process. To demonstrate how the dataset are processed and formatted, the loop detector data is taken as an example in this section.

The loop detector data has 323 sensing locations/areas. The dataset covers the whole year 2015 and its time interval is 5 minutes. That means the dataset has 365 (days) * 24 (hours) * 12 (5-minutes) = 105120 time intervals in total. Thus, the loop detector dataset can be represented by a 2D matrix. Figure 11 shows the left top corner of the 2D matrix with two axes, where the horizontal axis demonstrates the sensor locations and the vertical axis shows the timestamps. The values in the spatiotemporal matrix are the speed values and their unit is miles per hour (mph).

ID stamp	d005es15036	d005es15125	d005es15214	d005es15280	d005es15315	d005es15348	d005es15410
2015-01-01 00:00:00	61.939138	64.280883	62.077397	60.786423	63.120675	64.448315	63.411123
2015-01-01 00:05:00	59.232527	65.082450	64.808345	65.853953	59.206229	62.496716	65.992183
2015-01-01 00:10:00	61.991801	65.309123	64.803916	64.266082	62.239202	63.816610	60.196829
2015-01-01 00:15:00	62.480655	65.191651	67.206597	63.988427	65.808507	64.757556	62.011448

Figure 11 : An example of the loop detector data matrix.

Normally, the parameters in weight matrices in deep learning models are randomly initialized within the range of [-1, 1]. It is easier to train and test the deep learning models by normalizing the input data. Thus, the speed values in the 2D matrix are normalized to the range of [0,1] and the normalization is conducted using the following equation:

$$X = \frac{X - min(X)}{max(X) - min(X)} \tag{25}$$

Based on the traffic forecasting problem formulated by Equation (4), the input data should be a sequence X_T and the label should be Y_T. In this project, the default length of historical speed data T is set as 10 and the $Y_T = X_{T+1}$. Then, the loop detector data set covering a period of a year with the 5-minute interval can generate 104120 – T pairs of (X_T, Y_T) data samples.

Finally, to train and evaluate machine learning models, the data set is normally separated into three subsets, i.e. the training set, the validation set, and the testing set. In this project, all the samples extracted from the dataset are first randomized and then separated into three subsets according to the proportion of training: validation: testing = 7:2:1.

6. Platform Development

As described in Section 3, the transportation AI platform has three main components, the data warehouse, the web server, and the computation center. To build the platform, all three components need to be comprehensively designed and established using multiple technologies and tools. Meanwhile, timely and efficient data communication between the three components is also critical to the transportation AI platform. In this section, the key technologies and tools used in the platform development process are introduced.

6.1. Key Technologies in Platform Components

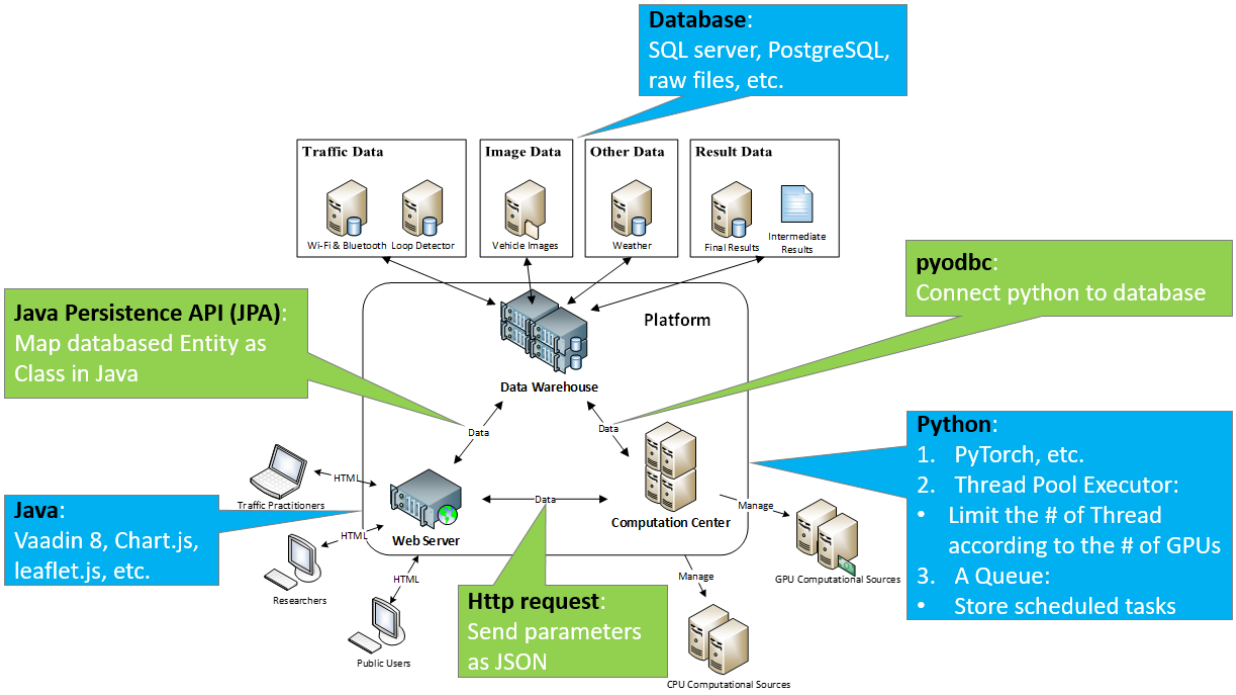

Figure 12 : Key technologies in the transportation AI platform architecture

The transportation AI platform has three components, but the core of the platform is data. Firstly, the platform users need to manipulate the platform to select data and model to fulfill the desired tasks and check the visualized and tabularized result data via the user interfaces. Secondly, the computation center needs to acquire the user-specified model parameters and import the existing datasets to train and test the deep learning-based models. Thirdly, the data warehouse, which contains multiple types of data, needs to provide timely responses to the data requests from the web server and the computation center and store the newly generated data in an efficient way. Figure 12 briefly shows the key technologies that are used to develop the transportation AI platform and complete all the project requirements.

6.1.1. Web Server

The Vaadin Framework is used to build the web server for the transportation AI platform. The web server is built using the Java programing language. When a public user attempts to access the transportation AI platform, the request will first be sent to the web server and a service/session will be built for the user. Then the user can continue to conduct other activities on the platform based on the response from the web server. Since the Vaadin framework provides multiple user-friendly interface plug-in tools to enhance the visualization performance, several Vaadin plug-in tools, such as Chart.js and Leaflet.js wrappers, are used in this project to visualize data sets and task results.

- Vaadin Framework [81]: The web server is developed based on the Vaadin 8 framework, which allows developers to build the entire web server and the user interface using only the Java programing language. Normally, the user interface of a web site is build based on multiple programing languages, including HTML, JavaScript, CSS, etc., but the Vaadin framework is a Java UI framework and library that simplifies the web application development. The web development code is written in Java and executed on the server's JVM, while the UI is rendered as HTML5 in the browser. The framework also automates all the communication between the browser and the server, and it provides various web components to speed up the development of web applications. Thus, the Vaadin framework is particularly suitable for prototype development. Thus, in this project, the Vaadin framework is chosen as the base framework to develop the transportation AI platform.
- Chart.js [82]: Chart.js is a JavaScript library that provides various plotting functions to generate multiple types of charts, including bar chart, line chart, pie chart, etc. The Vaadin framework is compatible with chart.js and provides a plug-in tool that can be integrated into the Vaadin framework to plot charts. In this project, the charts used to visualize the training and testing results are plotted by Chart.js.
- Leaflet.js [83]: Leaflet.js is the leading open-source JavaScript library for mobile-friendly interactive maps. Leaflet.js has good usability and efficient performance. It is also very simple to use to visualize geospatial data on maps. The Vaadin framework supports Leaflet.js based plug-in tools. In this project, Leaflet.js based plug-in tools are used to visualize geospatial related data.

6.1.2. Data Warehouse

The data warehouse mainly consists of different types of databases. As the user account information and the task information are the main data that need to be permanently stored, they are recorded by the SQL Server, which is a relational database. Other information, such as geospatial data in different data sets, is stored in databases with other specialties.

- SQL Server [84]: SQL Server is a relational database management system developed by Microsoft. It supports the standard SQL (Structured Query Language) language and other advanced database functions. Since the user and task information are relational data and SQL Server is a leading relational database management system, it is used to host all the user information, task information, and intermediate result data.
- PostgreSQL [85]: PostgreSQL is a powerful, open-source object-relational database system that is similar to SQL Server. However, PostgreSQL has many unique features aimed to help developers build applications, protect data integrity, and manage datasets. PostgreSQL is also extensible in that it supports the PostGIS [86], a spatial database extender for PostgreSQL. PostGIS supports the execution of location queries for geographic objects in SQL. In this project, the geospatial data in different datasets are all stored in the PostgreSQL, which can be used for data visualization.

6.1.3. Computation Center

The computation center behaves as the brain of the transportation AI platform. It is critical for the platform because it is the place where all the model training and testing are carried out. The computation center is built using the Python programing language. When a user of the platform has configured a task, including the goal, the dataset, the model, and the related parameters, the information about the new task is transmitted to the computation center. The computation center uses a Flask server to manage all the requests from the web server and control the task execution process. The data hosted in the computation center are processed by the Python packages, including Pandas and NumPy, and the deep learning models are developed and trained using the PyTorch package.

- Flask [87]: Flask is a micro web framework written in Python. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, and various open authentication technologies. In this project, Flask acts as a server that runs all the time to receive real-time requests from the web server and dispatches the requests to different threads to execute the training or testing jobs on different GPUs.
- Pandas [88] & NumPy [89]: Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. NumPy is the fundamental package for scientific computing with Python. In this project, both Pandas and NumPy are used to process data sets before the data sets are input to the deep learning models.
- PyTorch [90]: PyTorch is a Python machine learning package based on Torch. It has two main features, i.e. tensor computation with strong GPU acceleration and automatic differentiation for building and training neural networks. PyTorch and TensorFlow are the two most popular Python deep learning packages that have been widely used in industry and academia. Because of its

flexibility, PyTorch is used in this project to build all the traffic prediction models and conduct the training and testing tasks.

6.2.Key Technologies between Platform Components

To ensure the platform components can be efficiently and effectively connected, several technologies and tools are used to fulfill the data communication requirements.

6.2.1. Communication between Web Server and Data Warehouse

Since the transportation AI platform has multiple users, the user information needs to be easily stored and queried. Similarly, the platform's task information also needs to be stored in an efficient way. Because the information structures of the users and tasks are fixed, – it is better to utilize a uniform method to ensure timely persistence of the data into the database. Thus, the Java Persistence API (JPA) is adopted to do the data persistence. JPA has a clear mapping mechanism between the Java class and a database table, which means the Java program does not need to write simple SQL query code. With the help of JPA, all the data communication between the web server and the data warehouse is fulfilled without writing SQL code.

- Java Persistence API [91]: JPA is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. Now, it is considered the standard industry approach for Object to Relational Mapping (ORM) in the Java Industry.

6.2.2. Communication between Web Server and Computation Center

Since the web server is programed using Java and the computation center is programed using Python, these two platform components need to have a bridge connecting them with each other and supporting the data commination. As mentioned in Section 6.1, the computation center uses a Flask server to monitor the request from the web server. Hence, sending data using the http request to the Flask server from the web server is good option. In this project, in the web server side, the task information configured by the users are encoded as a JSON object and sent to the Flask server. In the computation center side, the encoded information is decoded into a Python dictionary object for further processing. In this way, the data from the web server can be successfully sent to the computation center.

However, in the opposite direction, the data from the computation center cannot be sent to the web server using the http request method. Considering that the intermediate and final results of each task are stored in the data warehouse, we solve the aforementioned problem with the help of the data warehouse, which means the web server query data is sent from the data warehouse instead of the computation center.

6.2.3. Communication between Data Warehouse and Computation Center

The data generated during the training or testing process are stored in the data warehouse via the database connection. The computation center uses the pyodbc package to build the connection to the SQL Server database. The data storage SQL code is directly written in the python environment.

6.3. Software and Hardware

Given the key technologies and tools in the platform, the transportation AI platform is developed and tested on a tower computer with 32GB memory and two GeGorce GTX 1080 GPUs. The version of the software and tools are listed as following:

- Database
  - SQL Server 2016
  - PostgreSQL 9.6
  - PostGIS 2.5
- Web server
  - Vaadin 8
  - Chart.js: 1.3.0
  - JPA (Spring boot): 1.5.3
- Python packages
  - PyTorch: 1.0.1
  - Pandas: 0.24.2
  - NumPy: 1.16.2
  - Flask: 1.0.2
  - pyodbc: 4.0.26

6.4. Platform Demonstration

Based on the designed architecture and the introduced key technologies, the core functions of the transportation AI platform are developed in this project. In this section, three main functions are demonstrated by showing the functions' user interface (UI). It should be noted that the transportation AI platform is designed as a prototype and the UI will be adjusted or redesigned with the future development process.

6.4.1. Platform Dashboard

The transportation AI platform has the ability to control the user identity and execute tasks launched by different users. Thus, a login UI is needed. Figure 13 shows the login UI of the platform. The user account is stored in the database. Only registered users with a username and a password can login to the platform.

The platform also has a dashboard to display the current status of the platform, including the running tasks, the historical tasks, and the status of the computation center. Figure 14 shows the UI of the dashboard. The left section of the UI is the menu of the platform and the right section is the functional panel that displays the platform information. The top part of the panel shows the existing tasks launched by the users and the left bottom part of the panel shows the tasks of other users. The right bottom part of the panel shows the status of the GPUs in the computation center. The user can also check the status of the existing task by visualizing the training and validation loss, as shown in Figure 15.

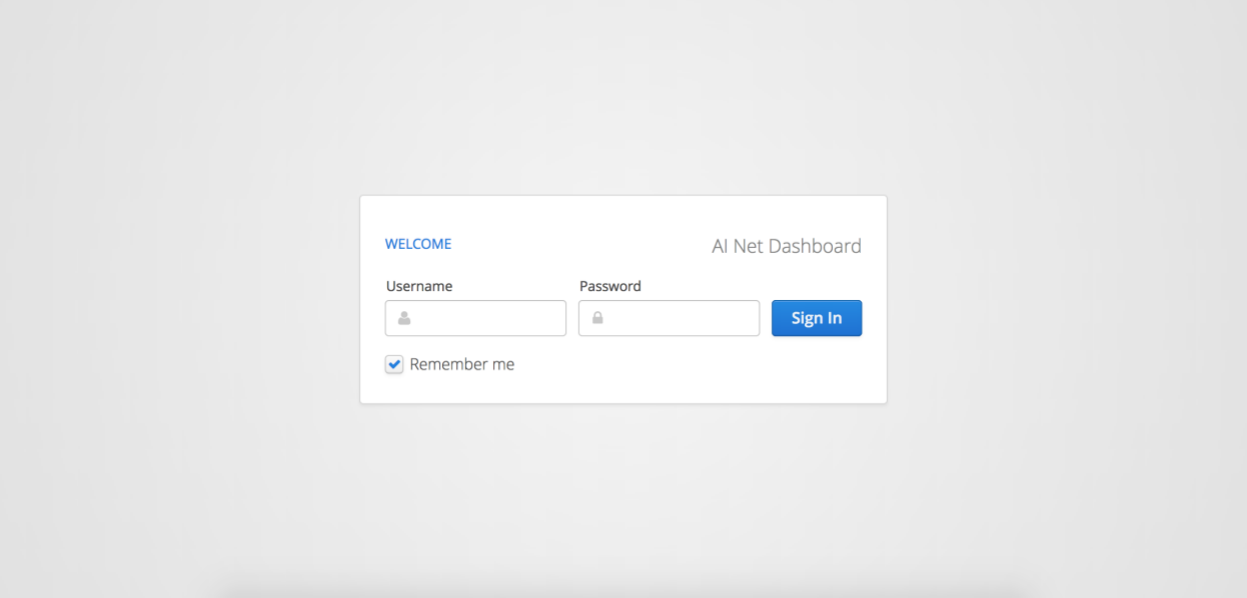

Figure 13 : Platform login interface

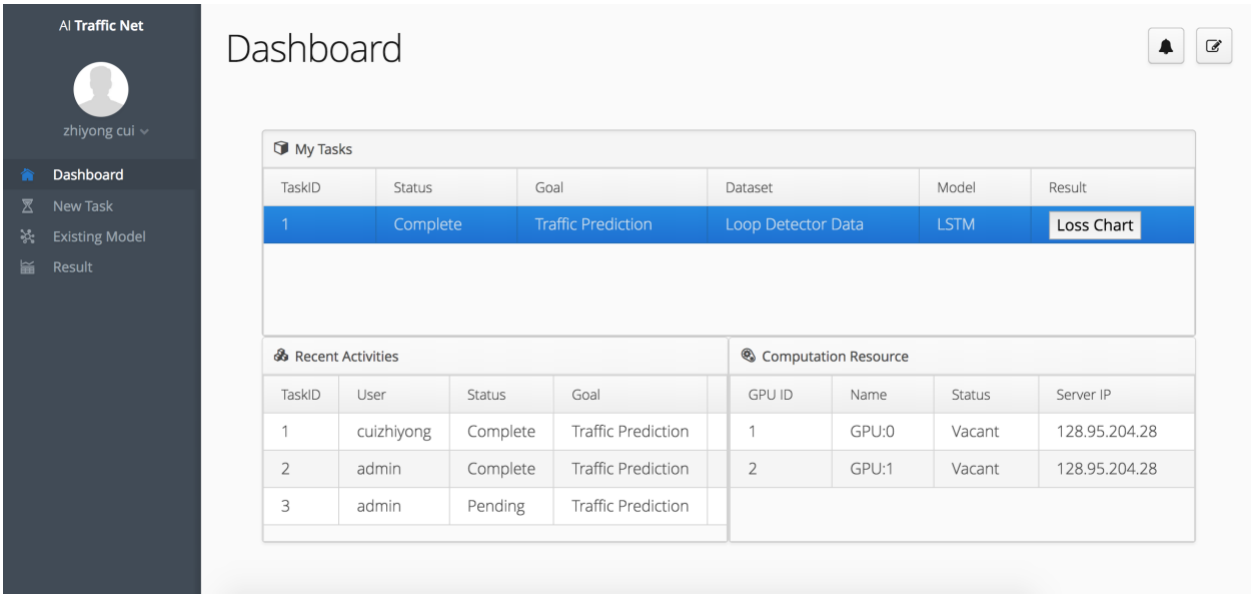

Figure 14 : Platform dashboard

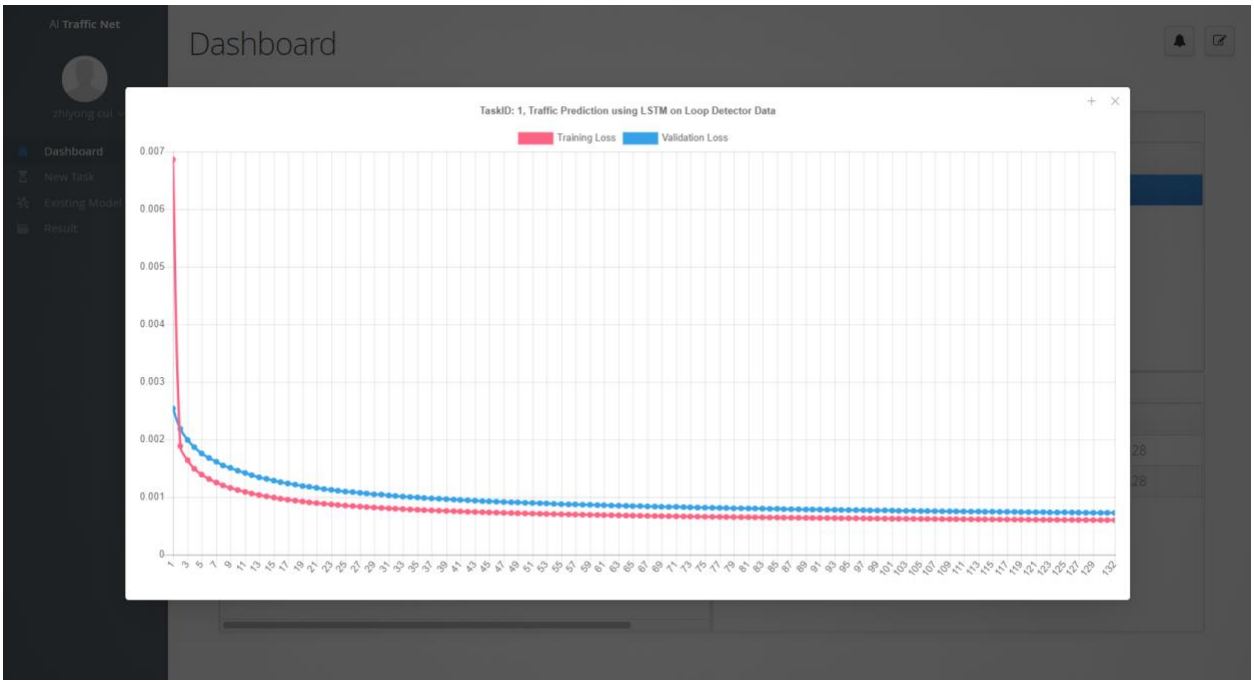

Figure 15 : Overview of the result of a training task on the platform dashboard

6.4.2. Creating New Task

The second tab of the platform menu is “New Task”, as shown in Figure 16. In this section, the process of creating a new task is demonstrated. Basically, the creation of a new task consists of three steps, including

task goal selection, dataset selection, and model selection, which are displayed by Figure 16, Figure 17, and Figure 19, respectively.

The task goal selection is the first step to determine the category of the task to be launched, such as traffic prediction and vehicle detection. After the task goal is selected, the data set selection step will be displayed on the functional panel of the platform and the data sets shown on the panel are the ones related to the selected task goal. Users can also check the detailed information for each dataset by clicking the “Info” button. Then, a description panel will be displayed, as shown in Figure 18. After the data set is selected and the “Next Step” button is clicked, the model selection panel will be displayed, as shown in Figure 19. The models are implemented using PyTorch and can be shared via the GitHub. The detailed information about the models, including the programing language, the library, and the brief introduction, is also displayed in the model selection table. By click on the table, the desired models can be selected. Then, if the “Start Training” button is clicked, the configuration parameters of the new task will be sent to the computation center to start the training process.

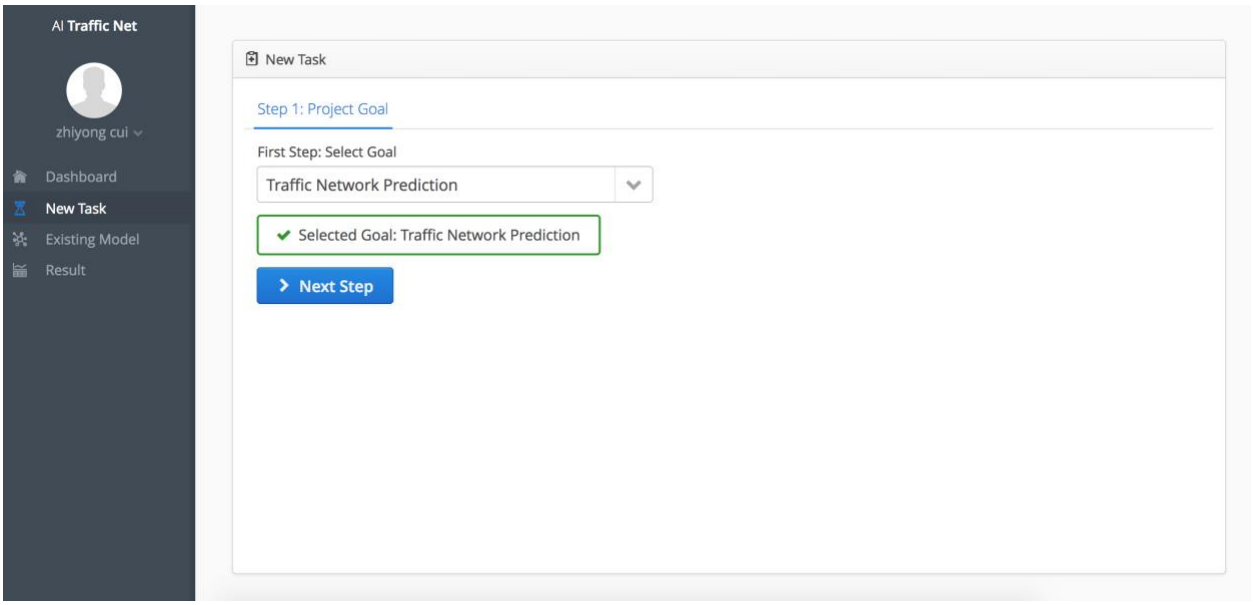

Figure 16 : Creating a new task and select the task goal

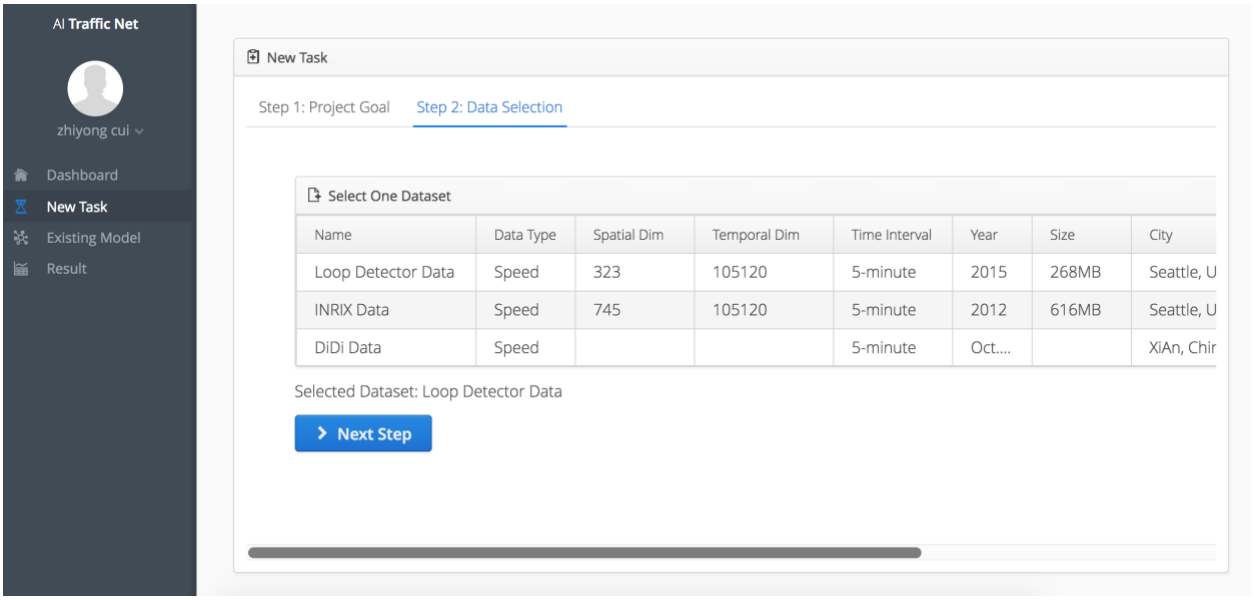

Figure 17 : Creating a new task and selecting the dataset

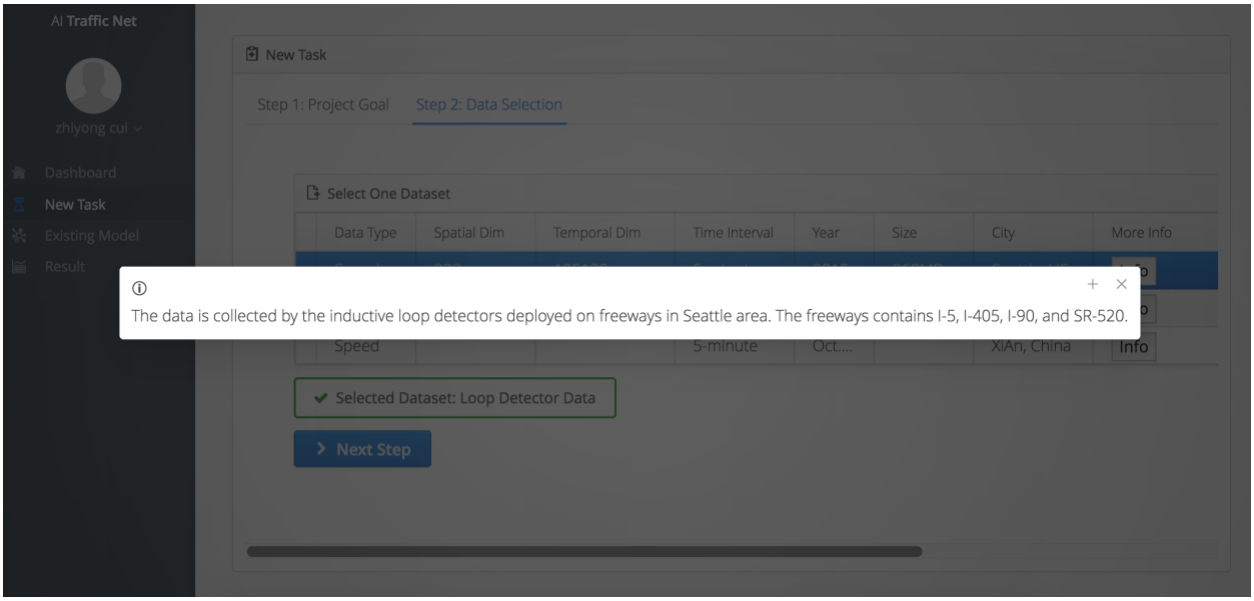

Figure 18 : Brief introduction of the dataset

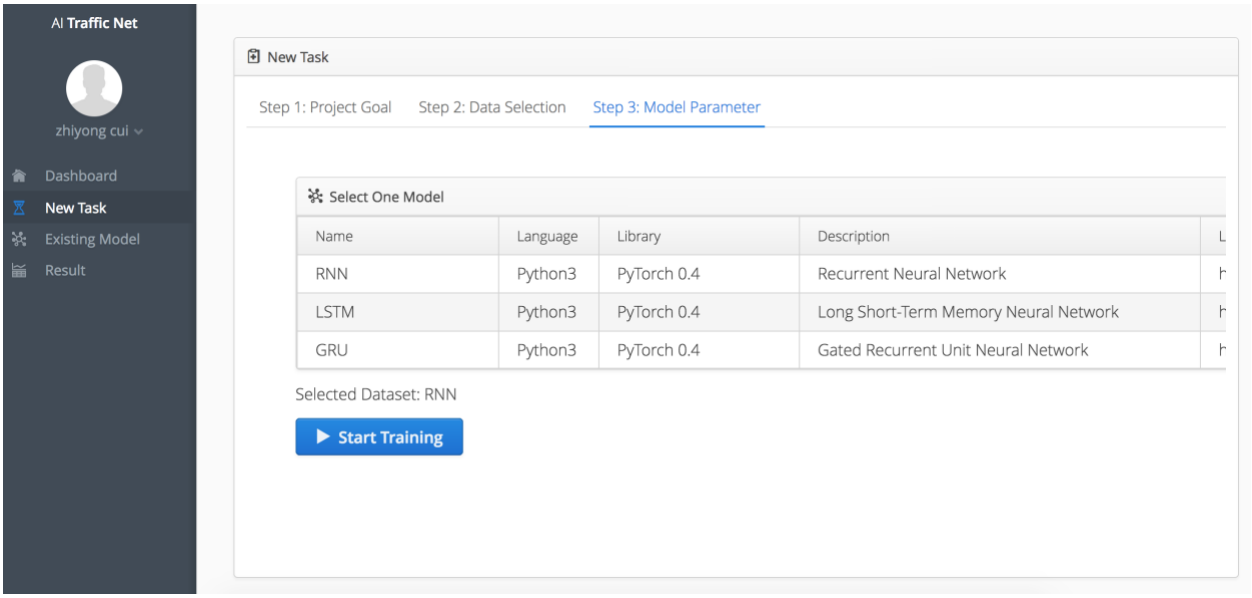

Figure 19 : Creating a new task and selecting a model

6.4.3. Exhibit Existing Models

The third section in the platform menu is the “Existing Model” tab. This section shows the existing models that have been trained by the user. Figure 20 shows the existing model panel with the detailed information of the trained model, including taskID, status, goal, dataset, model, and result. At the current stage, the result can show the training and validation loss of the trained model, as shown in Figure 21. The prediction accuracy and other model evaluation metrics will also be able to be displayed on the platform in the future.

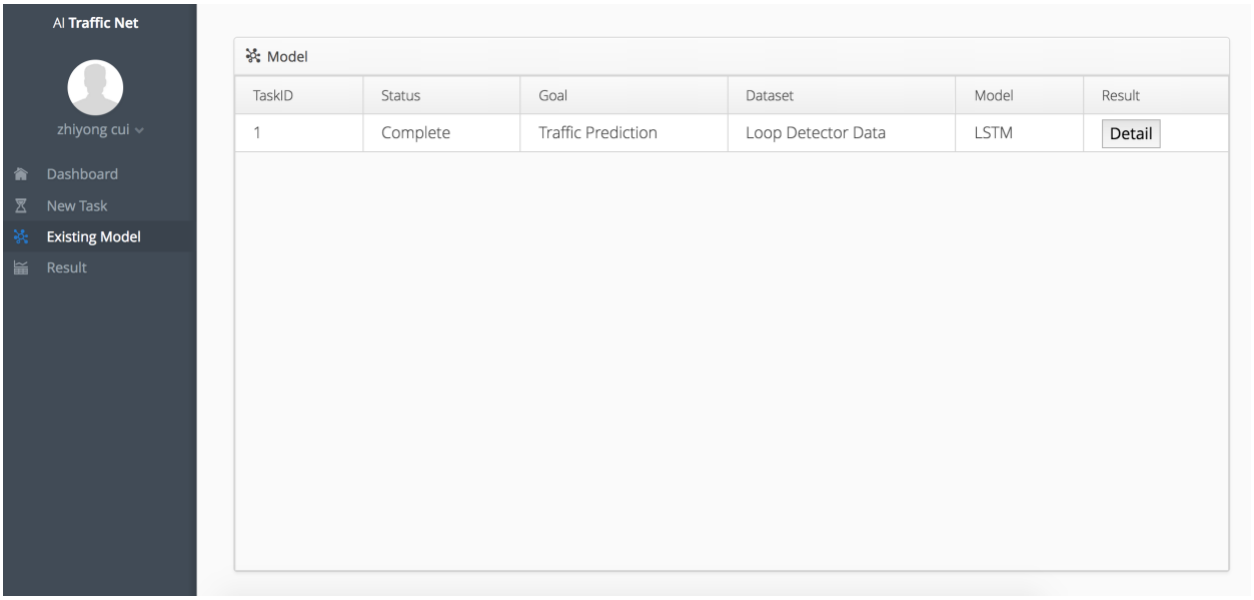

Figure 20 : Display the existing models

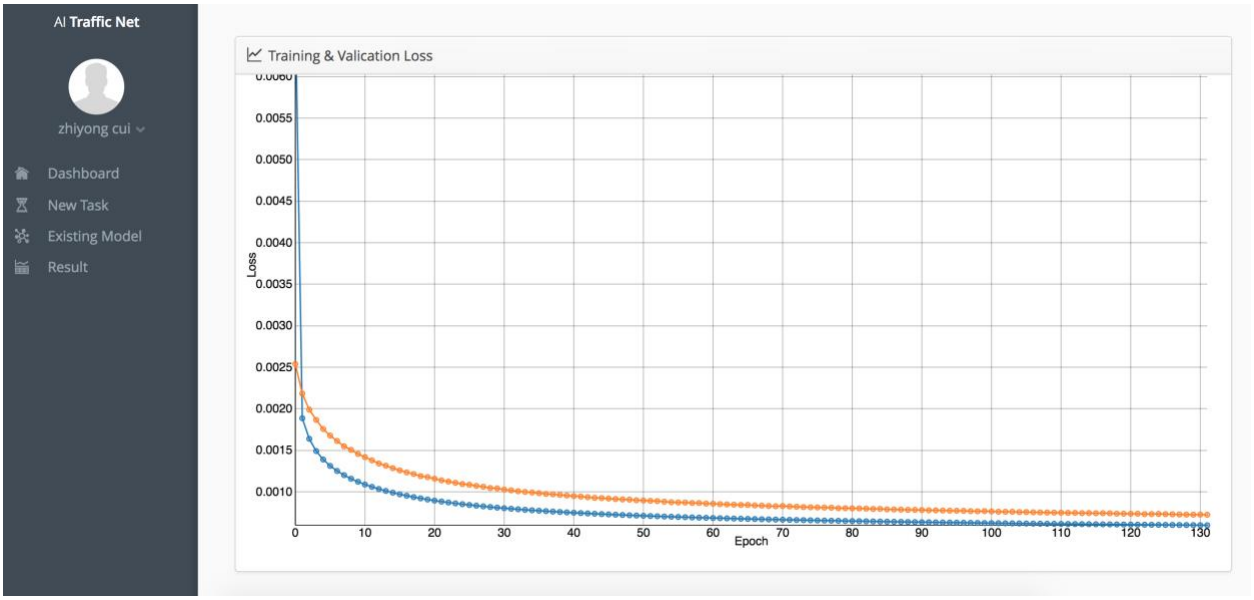

Figure 21 : Display the training and validation loss of the existing models

7. Traffic Prediction Task Performance Measurement

7.1.Performance Measurement Metrics

7.1.1. Hyper-parameters

For the traffic prediction task, several hyper-parameters need to be set before starting to train the model. The spatial dimension P is set according to the selected datasets mentioned in Section 5. The temporal dimension of the input sequence is set as 10, i.e. $T = 10$. As introduced in the data formatting section, the samples generated from a dataset are randomized and divided into training, validation, and testing sets with a ratio of 7:2:1. The batch size is set as 40 and the training loss is based on mean square error (MSE). Since the RMSProp [92] works well for RNNs, it is used as the gradient descent optimizer for the traffic prediction tasks. In RMSProp, the alpha (smoothing constant) is set as 0.99 and the epsilon (the term added to the denominator to improve numerical stability) is set as 10^{-8}. In addition, during the training process, the early stopping strategy is applied to the validation set to avoid overfitting. The training task will stop if the validation error value (MSE) cannot drop 0.00001 within 10 patience steps.

7.1.2. Evaluation metrics

The performances of all the traffic prediction models can be evaluated by several metrics, including mean absolute error (MAE), mean absolute percentage error (MAPE), standard deviation (STD), and Root Mean Squared Error (RMSE).:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{26}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_T - \hat{y}_T}{Y_T}\right| * 100\% \tag{27}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_T - \hat{y}_T)^2} \tag{28}$$

In this study, the samples of the input are traffic time series data with 10 time steps. The output/label is the next subsequent data of the input sequence. The performance of the proposed and the compared models are evaluated by three commonly used metrics in traffic forecasting, including 1) Mean Absolute Error (MAE), 2) Mean Absolute Percentage Error (MAPE), and 3) Root Mean Squared Error (RMSE).

7.2.Prediction Performance

7.2.1. Prediction Accuracy

In this section, the traffic prediction accuracy of different models applied on different datasets is presented. Table 1 shows the prediction accuracy on the loop detector dataset. The GRU does not perform as well as LSTM and GWGR. The GWGR method outperforms other models with all three metrics. The reason might be that GRU has no cell state to store historical information in its gate units compared to LSTM-based models, including GWGR. Both LSTM and GWGR work well and they have similar performance. However, compared with other models, the GWGR model is the state-of-the-art model that is implemented on the transportation AI platform because the GWGR can capture graph-based features while accommodating the physical specialties of traffic networks.

Model	MAE (mph)	MAPE	RMSE
GRU	4.58	10.34%	0.37
LSTM	2.70	6.83%	0.18
GWGR	2.48	5.44%	0.11

Table 1 : Traffic prediction accuracy on the loop detector dataset

Table 2 shows the prediction accuracy on the NPMRDS dataset. The GWGR still performs best among the three models in terms of the three performance metrics. It should be noted that, for the NPMRDS data, during the nighttime or off-peak hours when there are no observed speed values on specific roads, the missing speed values are comprehensively imputed by the data provider. Thus, there are few variations at the non-peak hours in the NPMRDS data. Further, the speed values in the NPMRDS data are all integers. Therefore, the calculated errors of the NPMRDS data is less than that of the loop detector data and the evaluated performance on NPMRDS data is inflated.

The transportation AI platform can provide the prediction accuracy on the user interface. This section only shows the model results of the traffic prediction-related models. In the future, when other types of tasks are implemented on the transportation AI platform, more model results will be presented and compared on the platform.

Model	MAE (mph)	MAPE	RMSE
GRU	2.12	8.59%	0.36
LSTM	1.14	3.88%	0.09
GWGR	0.93	2.67%	0.07

Table 2 : Traffic prediction accuracy on the NPMRDS dataset

7.2.2. Training and Validation Loss

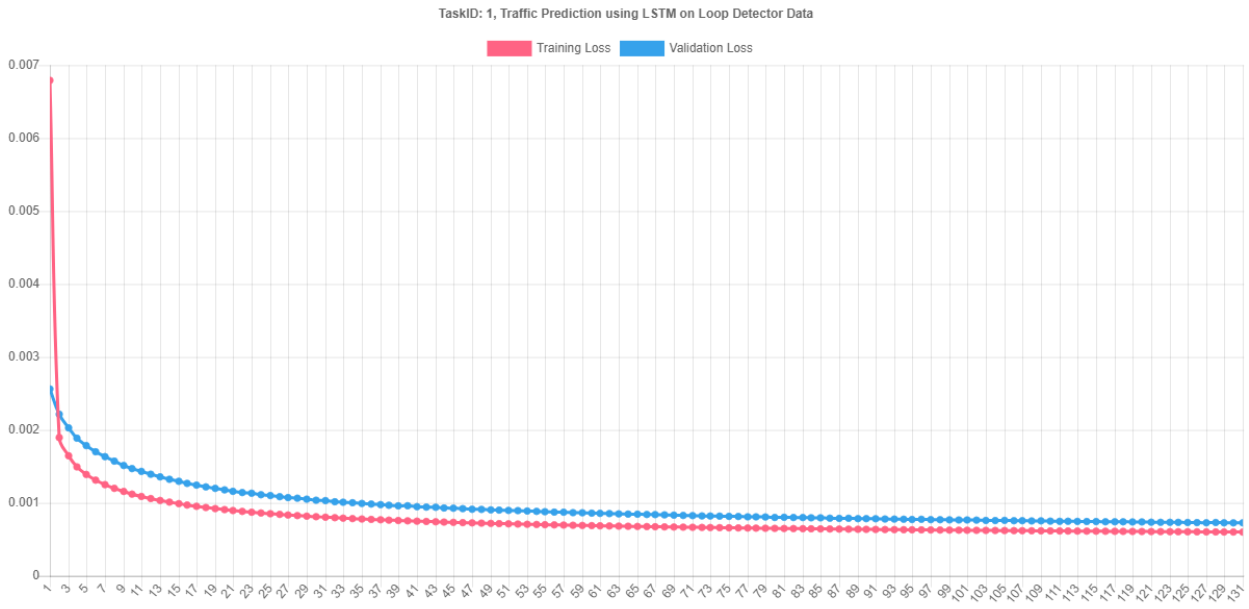

Figure 22 : Training and validation loss of the traffic prediction task using the LSTM model based on the loop detector dataset.

The transportation AI platform also records the training and validation loss during the training process. The training and validation loss is critical to determine whether a model is well trained. If the model is correctly designed and the training parameters are properly set, both training loss and validation loss will

converge after a certain number of steps. Normally, the validation loss will be a little bit larger than the training loss since the back-propagation process is conducted only based on the training set. Figure 22 and Figure 23 visualize the training and validation loss of the traffic prediction task tested on the loop detector dataset and the NPMRDS (INRIX) dataset, respectively. In the future, the transportation AI platform will provide more visualization functions, such as training time visualization and map-based prediction results visualization, to let the user quickly understand the performance of the tested models.

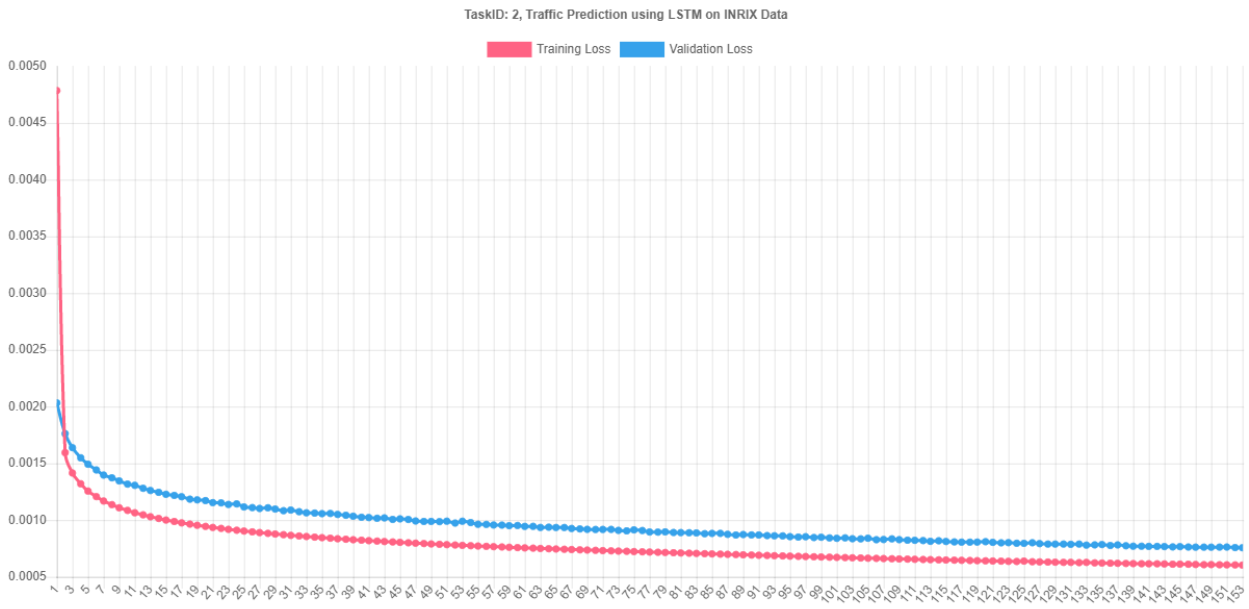

Figure 23 : Training and validation loss of the traffic prediction task using the LSTM model based on the NPMRDS dataset.

8. Conclusion

The emergence of novel traffic sensing, data communication, and artificial intelligence technologies has greatly stimulated the growth of transportation data. To overcome the challenges brought on by immense transportation data, this project aims to utilize artificial intelligence technologies to derive knowledge from a huge volume of transportation data by building a transportation AI platform. Traffic forecasting involving high-dimensional spatiotemporal data is a good scenario to utilize novel deep learning models.

In this project, the research team developed a prototype artificial intelligence platform for solving challenging transportation problems using large-volume high-dimensional transportation data and complex models. This AI platform is capable of providing standardized datasets and novel deep learning-based models for solving specific pre-defined transportation problems. For a specific problem, the platform offers standardized training and testing procedures to assist in the evaluation of emerging novel methodologies.

The contributions of this project can be summarized as follows:

- A transportation AI platform incorporating large-volume high-dimensional transportation data and complex models is developed for solving challenging transportation problems. This AI platform is capable of providing standardized datasets and novel deep learning-based models for specific problems.
- A novel architecture is designed for the transportation AI platform to enhance the efficiency of the transportation data processing, management, and communication and increase the computational power of the platform. The transportation AI platform consists of three main components, i.e. the web server, data warehouse, and computation center. All three components are connected by using novel data communication technologies.
- The web server uses a novel web-application framework to provide user-friendly interfaces and support the interaction between users and the platform.
- A data storage and management schema is designed with the help of the data warehouse and the computation center to manage multiple network-wide traffic data sets for supporting the traffic prediction task and to simplify the whole training and testing process.
- Novel deep learning-based traffic prediction models and baseline model are developed and stored in the computation center of the platform. A novel graph-based deep learning, i.e. the graph wavelet gated recurrent (GWGR) network is proposed to capture the complicated topological structure of the roadway network to improve the traffic prediction performance. Compared with baseline models, including LSTM and GRU, the GWGR shows superior prediction performance.
- New visualization technologies are also incorporated in the transportation AI platform to help users intuitively and efficiently use the platform and deploy novel methodologies.

In summary, the planned tasks of the project are all completed. Further, the transportation AI platform has the potential to be a platform to host standard datasets and models to stimulate the development of novel deep learning models for transportation problems. In the future, the research team plans to develop a web page to demonstrate what kinds of problems the platform can deal with and introduce the existing datasets and models on the platform. In addition, the best performance of each model for each specific task will be displayed on the platform to reduce researchers' duplicated modeling work and stimulate the emergence of novel technologies.

References

[1] Y. Demchenko, C. De Laat, and P. Membrey, "Defining architecture components of the Big Data Ecosystem," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*, 2014, pp. 104–112.

[2] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[3] F. Li, J. Gong, Y. Liang, and J. Zhou, "Real-time congestion prediction for urban arterials using adaptive data-driven methods," *Multimed. Tools Appl.*, vol. 75, no. 24, pp. 17573–17592, 2016.

[4] X. Kong, Z. Xu, G. Shen, J. Wang, Q. Yang, and B. Zhang, "Urban traffic congestion estimation and prediction based on floating car trajectory data," *Futur. Gener. Comput. Syst.*, vol. 61, pp. 97–107, 2016.

[5] J. Kim and G. Wang, "Diagnosis and prediction of traffic congestion on urban road networks using Bayesian networks," *Transp. Res. Rec.*, vol. 2595, no. 1, pp. 108–118, 2016.

[6] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS One*, vol. 10, no. 3, p. e0119044, 2015.

[7] C.-S. Li and M.-C. Chen, "A data mining based approach for travel time prediction in freeway with non-recurrent congestion," *Neurocomputing*, vol. 133, pp. 74–83, 2014.

[8] X. Zhang, E. Onieva, A. Perallos, E. Osaba, and V. C. S. Lee, "Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction," *Transp. Res. Part C Emerg. Technol.*, vol. 43, pp. 127–142, 2014.

[9] S. Kikuchi, "Artificial intelligence in transportation analysis: approaches, methods, and applications," *Transp. Res. Part C*, vol. 5, no. 17, p. 455, 2009.

[10] X. Ma, Y.-J. Wu, and Y. Wang, "DRIVE Net: E-science transportation platform for data sharing, visualization, modeling, and analysis," *Transp. Res. Rec. J. Transp. Res. Board*, no. 2215, pp. 37–49, 2011.

[11] Z. Cui, S. Zhang, K. C. Henrickson, and Y. Wang, "New progress of DRIVE Net: An E-science transportation platform for data sharing, visualization, modeling, and analysis," in *Smart Cities Conference (ISC2), 2016 IEEE International*, 2016, pp. 1–2.

[12] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big data*, vol. 1, no. 1, pp. 51–59, 2013.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[14] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[15] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal

control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, 2003.

[16] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, 2010.

[17] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Autom. Sin.*, vol. 3, no. 3, pp. 247–254, 2016.

[18] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[19] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, "Capturing car-following behaviors by deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 910–920, 2017.

[20] M. Zhou, X. Qu, and X. Li, "A recurrent neural network based microscopic car following model to predict traffic oscillation," *Transp. Res. part C Emerg. Technol.*, vol. 84, pp. 245–264, 2017.

[21] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transp. Res. part C Emerg. Technol.*, vol. 97, pp. 348–368, 2018.

[22] D. Lee, Y. P. Kwon, S. McMains, and J. K. Hedrick, "Convolution neural network-based lane change intention prediction of surrounding vehicles for ACC," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[23] M. Mukadam, A. Cosgun, A. Nakhaei, and K. Fujimura, "Tactical decision making for lane changing with deep reinforcement learning," 2017.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv Prepr. arXiv1804.02767*, 2018.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[27] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[28] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[29] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv Prepr. arXiv1604.07316*, 2016.

[30] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017,

pp. 2174–2182.

[31] A. Kendall *et al.*, "Learning to Drive in a Day," *arXiv Prepr. arXiv1807.00412*, 2018.

[32] "IBM patent describes AI-managed traffic system." [Online]. Available: https://www.smart2zero.com/news/ibm-patent-describes-ai-managed-traffic-system. [Accessed: 27-May-2019].

[33] "Surtrac: Intelligent Traffic Signal Control System." [Online]. Available: https://www.rapidflowtech.com/surtrac. [Accessed: 27-May-2019].

[34] "Berkeley DeepDrive | We seek to merge deep learning with automotive perception and bring computer vision technology to the forefront." [Online]. Available: https://deepdrive.berkeley.edu/project/adaptive-traffic-signal-control-based-deep-reinforcement-learning. [Accessed: 28-May-2019].

[35] "PTV Optima." [Online]. Available: http://vision-traffic.ptvgroup.com/en-us/products/ptv-optima/. [Accessed: 30-May-2019].

[36] "TrafficLink Platform - Miovision." [Online]. Available: https://miovision.com/resources/trafficlink-platform/. [Accessed: 30-May-2019].

[37] "timon-project.eu." [Online]. Available: https://www.timon-project.eu/. [Accessed: 30-May-2019].

[38] S. Amershi *et al.*, "Software Engineering for Machine Learning: A Case Study," *41st ACM/IEEE Int. Conf. Softw. Eng. (ICSE 2019)*, 2019.

[39] "Waycare | AI-driven Mobility Solutions." [Online]. Available: http://waycaretech.com/. [Accessed: 28-May-2019].

[40] "Traffic Prediction | Strategic Management and Coordination | Siemens." [Online]. Available: https://new.siemens.com/global/en/products/mobility/road-solutions/traffic-management/strategic-management-and-coordination/traffic-prediction.html. [Accessed: 29-May-2019].

[41] "Traffic & Transportation Data Analytics Platform for Government | UrbanLogiq." [Online]. Available: https://www.urbanlogiq.com/traffic. [Accessed: 29-May-2019].

[42] "China's biggest ride-hailing platform Didi now wants to help cities solve traffic jams | South China Morning Post." [Online]. Available: https://www.scmp.com/tech/china-tech/article/2154027/chinas-biggest-ride-hailing-platform-wants-crunch-its-data-improve. [Accessed: 30-May-2019].

[43] "DiDi Labs - Intelligent Transportation Technology & Security." [Online]. Available: http://www.didi-labs.com/?jobs=active. [Accessed: 29-May-2019].

[44] "Home » drive.ai » the self-driving car is here." [Online]. Available: https://www.drive.ai/. [Accessed: 30-May-2019].

[45] “Autonomous Car Development Platform | NVIDIA DRIVE AGX.” [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/. [Accessed: 30-May-2019].

[46] “Autopilot | Tesla.” [Online]. Available: https://www.tesla.com/autopilot. [Accessed: 30-May-2019].

[47] “Waymo – Waymo.” [Online]. Available: https://waymo.com/. [Accessed: 30-May-2019].

[48] “Apollo.” [Online]. Available: http://apollo.auto/. [Accessed: 30-May-2019].

[49] “comma.ai.” [Online]. Available: https://comma.ai/. [Accessed: 30-May-2019].

[50] D. Park and L. R. Rilett, “Forecasting freeway link travel times with a multilayer feedforward neural network,” *Comput. Civ. Infrastruct. Eng.*, vol. 14, no. 5, pp. 357–367, 1999.

[51] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, “Short-term traffic forecasting: Where we are and where we’re going,” *Transp. Res. Part C Emerg. Technol.*, vol. 43, pp. 3–19, Jun. 2014.

[52] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” *Transp. Res. Part C Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[53] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction,” *Sensors*, vol. 17, no. 4, p. 818, 2017.

[54] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[55] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said, “Short-term prediction of traffic volume in urban arterials,” *J. Transp. Eng.*, vol. 121, no. 3, pp. 249–254, 1995.

[56] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” *Transp. Res. Part C Emerg. Technol.*, vol. 54, pp. 187–197, 2015.

[57] J. Van Lint, S. Hoogendoorn, and H. Van Zuylen, “Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks,” *Transp. Res. Rec. J. Transp. Res. Board*, no. 1811, pp. 30–39, 2002.

[58] N. G. Polson and V. O. Sokolov, “Deep learning for short-term traffic flow prediction,” *Transp. Res. Part C Emerg. Technol.*, vol. 79, pp. 1–17, 2017.

[59] J. Hua and A. Faghri, “Applcations of Artificial Neural Networks to Intelligent Vehicle-Highway Systems,” *Record (Washington).*, vol. 1453, p. 83, 1994.

[60] H. Yin, S. Wong, J. Xu, and C. K. Wong, “Urban traffic flow prediction using a fuzzy-neural approach,” *Transp. Res. Part C Emerg. Technol.*, vol. 10, no. 2, pp. 85–98, 2002.

[61] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, “Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks,” *Sensors*, vol. 17, no. 7, p. 1501, 2017.

[62] W. Huang, G. Song, H. Hong, and K. Xie, “Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning.,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, 2014.

[63] F. Kong, J. Li, B. Jiang, and H. Song, “Short-term traffic flow prediction in smart multimedia system for Internet of Vehicles based on deep belief network,” *Futur. Gener. Comput. Syst.*, vol. 93, pp. 460–472, 2019.

[64] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, and others, “Traffic flow prediction with big data: A deep learning approach.,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2015.

[65] B. Liao *et al.*, “Deep Sequence Learning with Auxiliary Information for Traffic Prediction,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 537–546.

[66] Y. Liang, Z. Cui, Y. Tian, H. Chen, and Y. Wang, “A Deep Generative Adversarial Architecture for Network-Wide Spatial-Temporal Traffic-State Estimation,” *Transp. Res. Rec. J. Transp. Res. Board*, p. 036119811879873, Oct. 2018.

[67] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, “Pattern Sensitive Prediction of Traffic Flow Based on Generative Adversarial Framework,” *IEEE Trans. Intell. Transp. Syst.*, no. 99, pp. 1–6, 2018.

[68] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[69] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv Prepr. arXiv1406.1078*, 2014.

[70] Y. Duan, Y. Lv, and F.-Y. Wang, “Travel time prediction with LSTM neural network,” in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, 2016, pp. 1053–1058.

[71] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, “LSTM network: a deep learning approach for short-term traffic forecast,” *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, 2017.

[72] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[73] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv Prepr. arXiv1412.3555*, 2014.

[74] K. Greff, R. K. Srivastava, J. Koutn\’\ik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE Trans. neural networks Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, 2017.

[75] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv Prepr. arXiv1409.1259*, 2014.

[76] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, “Graph Wavelet Neural Network,” in *International Conference on Learning Representations*, 2019.

[77] FHWA, “National Performance Management Research Data Set (NPMRDS),” 2019. [Online]. Available: https://ops.fhwa.dot.gov/perf_measurement/index.htm.

[78] Z. Cui, R. Ke, and Y. Wang, “Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction,” in *6th International Workshop on Urban Computing (UrbComp 2017)*, 2017.

[79] K. Henrickson, Y. Zou, and Y. Wang, “Flexible and robust method for missing loop detector data imputation,” *Transp. Res. Rec. J. Transp. Res. Board*, no. 2527, pp. 29–36, 2015.

[80] Y. Wang, R. Ke, W. Zhang, Z. Cui, and K. Henrickson, “Digital Roadway Interactive Visualization and Evaluation Network Applications to WSDOT Operational Data Usage,” *Diss. Univ. Washingt. Seattle, Washingt.*, 2016.

[81] vaadin.com, “Vaadin Framework 8,” 2019. .

[82] Chart.js, “https://www.chartjs.org/,” 2019. .

[83] Leaflet.js, “https://leafletjs.com/,” 2019. .

[84] S. Server, “https://www.microsoft.com/en-us/sql-server/sql-server-2017,” 2019. .

[85] PostgreSQL, “https://www.postgresql.org/,” 2019. .

[86] PostGIS, “https://postgis.net/,” 2019. .

[87] Flask, “http://flask.pocoo.org/,” 2019. .

[88] Pandas, “https://pandas.pydata.org/,” 2019. .

[89] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, p. 22, 2011.

[90] A. Paszke, S. Gross, S. Chintala, and G. Chanan, “Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration,” *PyTorch Tensors Dyn. neural networks Python with strong GPU Accel.*, vol. 6, 2017.

[91] H. Böck, “Java persistence api,” in *The Definitive Guide to NetBeans™ Platform 7*, Springer, 2012, pp. 315–320.

[92] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA Neural networks Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[93] Z. Cui, R. Ke, and Y. Wang, “Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction,” *arXiv Prepr. arXiv1801.02143*, 2018.