TARA-Net: A Fusion Network for Detecting Takeaway Rider Accidents

YIFAN HE, Fudan University, China
ZHAO LI*, Alibaba Group, China
LEI FU, Alibaba Group, China
ANHUI WANG, Alibaba Group, China
PENG ZHANG, Alibaba Group, China
SHUIGENG ZHOU, Fudan University, China
JI ZHANG, The University of Southern Queensland, Australia
TING YU, Zhejiang Lab, China

In the emerging business of food delivery, rider traffic accidents raise financial cost and social traffic burden. Although there has been much effort on traffic accident forecasting using temporal-spatial prediction models, none of existing work studies the problem of detecting the takeaway rider accidents based on food delivery trajectory data. In this paper, we aim to detect whether a takeaway rider meets an accident on a certain time period based on trajectories of food delivery and riders' contextual information. The food delivery data has a heterogeneous information structure and carries contextual information such as weather and delivery history, and trajectory data are collected as a spatial-temporal sequence. In this paper, we propose a **T**ake**A**way **R**ider **A**ccident detection fusion network TARA-Net to jointly model these heterogeneous and spatial-temporal sequence data. We utilize the residual network to extract basic contextual information features and take advantage of transformer encoder to capture trajectory features. These embedding features are concatenated into a pyramidal feed-forward neural network. We jointly train the above three components to combine the benefits of spatial-temporal trajectory data and sparse basic contextual data for early detecting traffic accidents. Furthermore, due to traffic accidents rarely happen in food delivery, we propose a sampling mechanism to alleviate the imbalance of samples when training the model. We evaluate the model on a transportation mode classification data set Geolife and a real-world *Ele.me* data set with over 3 million riders. The experimental results show that the proposed model is superior to the state-of-the-art.

CCS Concepts: • **Computing methodologies** → **Supervised learning by classification**; **Neural networks**; **Learning latent representations**; • **Information systems** → **Spatial-temporal systems**.

*Corresponding author

Authors' addresses: Yifan He, Fudan University, Shanghai, China, yfhe20@fudan.edu.cn; Zhao Li, Alibaba Group, Hangzhou, China, lizhao.lz@alibaba-inc.com; Lei Fu, Alibaba Group, Hangzhou, China, arley.fl@alibaba-inc.com; Anhui Wang, Alibaba Group, Hangzhou, China, anhui.wah@alibaba-inc.com; Peng Zhang, Alibaba Group, Hangzhou, China, hanyi.zp@alibaba-inc.com; Shuigeng Zhou, Fudan University, Shang Hai, China, sgzhou@fudan.edu.cn; Ji Zhang, The University of Southern Queensland, Queensland, Australia, Ji.Zhang@usq.edu.au; Ting Yu, Zhejiang Lab, Hang Zhou, China, yuting@zhejianglab.com.

Additional Key Words and Phrases: traffic accident detection, trajectory data, transformer encoder, residual network, deep learning

ACM Reference Format:
Yifan He, Zhao Li, Lei Fu, Anhui Wang, Peng Zhang, Shuigeng Zhou, Ji Zhang, and Ting Yu. 2020. TARA-Net: A Fusion Network for Detecting Takeaway Rider Accidents. 1, 1 (February 2020), 21 pages. https://doi.org/***********

1 INTRODUCTION

During the recent peak of covid-19 cases in China, food-delivery companies such as *Ele.me* and *Meituan* played a crucial role in helping people who stayed at home by developing contactless food delivery. Contactless food delivery is very different from traditional work that allows Takeaway Riders to choose their own work hours. However, many takeaway riders are under the pressure of heavily burdened delivery tasks in a short amount of time, i.e., two hours during lunchtime. To complete these delivery tasks, they often jump red lights or take a shortcut to avoid late-delivery penalties, which leads to many traffic accidents during the delivery time. In the first half of 2019, the city of Shanghai in China recorded 12 road accidents a week involving food-delivery riders. However, due to subjective or objective reasons, many of the accidents were not reported, which may hurt the riders and the delivery platform. For riders, they may lost the opportunity of assistance from the platform in handling the accident, such as medical reimbursement and subsidies. For platforms, failure to properly handle undetected incidents may affect their reputation.

The purpose of this work is to detect the takeaway rider accidents based on food delivery trajectory data. Generally, a food-delivery company can collect the trajectory data by tracing the GPS signals obtained from a takeaway rider. The takeaway rider trajectory is represented as a set of spatial-temporal trajectory dots, Some hidden characteristics of these dots are useful for accident detection. For example, Fig. 1 shows the trajectory involving a traffic accident. Obviously, the trajectory dots of the rider clustered at the traffic site and the hospital and appear a long-time stay, which means a higher probability that an accident happened.

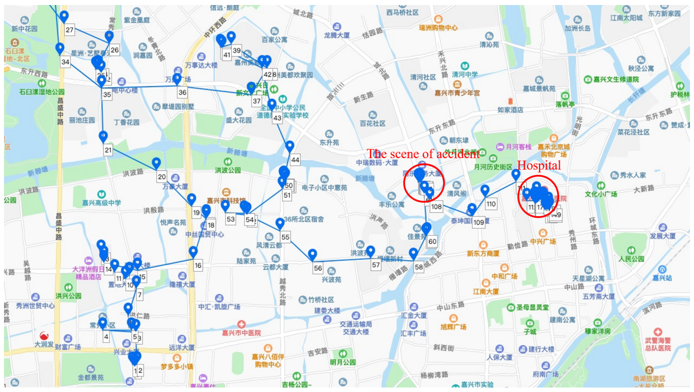


Fig. 1. An example of the trajectory data generated from a takeaway rider. The blue dots are trajectory dots formed as {L,T}, where L represents the Location information of the trajectory dots, T represents the Time information of trajectory dots.

To detect traffic accidents of takeaway riders. We need to analyze their data given in Fig. 1, which is essential to predict the class label (safe/unsafe) based on the collected trajectory data. Analyzing the trajectory data has been widely studied in the vehicle for hire companies such as Uber and Didi. Generally, two types of deep learning models, CNNs and RNNs are used to analyzing trajectory data depending on the representations of trajectories. For example, to predict human trajectory, the work [59] represents trajectory data as sequences of locations, and proposed Collision-Free LSTM by adding the Repulsion pooling layer to classical LSTM, which can share hidden-states of neighboring pedestrians for human trajectory prediction. To predict the destination of taxi trajectories, the work [38] represented trajectory as two-dimensional images and adopted multi-layer CNN to combine multi-scale trajectory patterns for predicting taxi destination. To estimate travel time, the work [53] represented trajectory as the sequence of the image-like matrixes, stacked LSTM on the proposed Geo-Conv layer to capture temporal dependencies for travel time estimation.

In addition to analyzing the trajectory data, we also have rich contextual information data, consisting of historical rider information and current environmental information. These contextual data have different structures and characteristics from the trajectory data, so it is unreasonable to extract representations of them by an identical network. From the perspective of deep neural networks for feature representation of heterogeneous data, the fusion network has been widely used to extract the heterogeneous feature representation for the regression and classification problems. The principle behind is to combine the benefits of different network components, which performs better on a particular type of data. For instance, the Wide and Deep model [6] combines the memorization of the wide network and the generalization of the deep network can emphasize both low and high-order feature interactions. Based on wide and deep, the DeepFM model [17], Deep and Cross [56], XDeepFM [34] are proposed to construct both low and high order intersection features. However, these models focus on the click-through rate of the recommender system, lacking of methodology for traffic accident detection. They cannot be directly used to solve our problem.

Compared with the above work, this research meets the challenges of the sparse, spatial-temporal and imbalance characteristics of data. Specifically, the challenges are listed as follows:

- First, the information of daily takeaway rider trajectory contains both categorical and numerical features, where the category features are often imported into the network as the binarized sparse features with one-hot encoding, which always need further feature engineering.
- Second, trajectory data contains a lot of spatial-temporal information. However, most of the trajectory dots are not informative, and only part of them is useful for accident detection. We need to separately model them.
- Third, traffic accident rarely happened in food delivery, thousands of takeaway orders may just involve one accident, which causes the class imbalance of training data.

To solve the challenges, we present in this paper a TARA-Net method to analyze both basic and trajectory data for accurate accident detection. To solve the first challenge, TARA-Net concatenates category embedding features and numerical features and puts them into the residual neural network to extract comprehensive intersection features. To solve the second challenge, TARA-Net uses a transformer to distribute the attention of trajectory position embedding sequence, which can automatically assign a higher weight to accident-related trajectory data. To solve the third challenge, we propose a sampling mechanism to create balance data distribution for each training batch. The contribution of the paper can be summarized as follows:

- We present a fusion network called TARA-Net, which integrates the architectures of Transformer and Resnet and jointly training network with both sparse embedding and trajectory position embedding. It can be trained end-to-end without any manual feature engineering.
- We randomly select a certain percentage of accident samples and combine them into each training batch, which greatly alleviates the problem of class imbalance.
- We evaluate TARA-Net on real-world *Ele.me* takeaway rider data, which shows excellent performance on the rider traffic accident detection task.

2 RELATED WORK

In this part, we survey the work of traffic accident analysis and trajectory data mining.

2.1 Traffic Accident Analysis

Prior works have made significant advances on the analysis of traffic accidents from various dimensions [11, 40, 55]. The two most important branches are traffic accident forecasting and traffic accident detection. Traffic accident forecasting has become a fundamental challenge in urban sensing. It first uses historical information to predict the future accident risk in a certain area, and then alerts people in the risk area and allocate more assistance to reduce the occurrence of an accident or mitigate the injury of an accident. Prior works can be grouped into two categories: conventional pattern-based methods and deep neural network models. For example, the work [45] uses a support vector machine model with a Gaussian kernel to extract key factors from the collected data set which are responsible for the majority of the accident, and then infers future accidents. The work [42] employs neural network and decision trees to analyze continuous data and categorical data about the accident respectively, and combines them to forecast accident. However, pattern-based methods always assume that traffic accident data is stationary. Hence, many follow-up works are proposed to capture dynamic traffic changes by the deep neural network. For example, the work [43] proposes a deep learning model based on recurrent neural network towards a prediction of traffic accident risk. Many works use multi-modal data for accident risk forecasting, the work [63] proposes GraphCast, a graph neural network framework to accurately forecast the traffic risks in a city by jointly exploring the multi-modal data collected from social media sensing and remote sensing paradigms. Furthermore, some accident forecasting models need to be designed for particular necessities. To solve the problem of spatial heterogeneity of the environment (e.g., urban vs. rural), the work [61] proposes a Hetero-ConvLSTM framework, which incorporates spatial graph features and spatial model ensemble to address the spatial heterogeneity in the traffic data. In order to further improve prediction granularity, the work [68] proposes the differential time-varying graph neural network to capture the immediate changes of traffic status and dynamic inter-subregion correlations. The Work [25] proposes a deep dynamic fusion network to model both spatial-temporal dependencies and automatically aggregating heterogeneous external factors in a dynamic manner for fine-grained traffic accident forecasting.

Besides the work of traffic accident forecasting, the detection of accidents is also important for the urban transportation system, because immediately handling of traffic accidents can reduce the loss. Several vision-based traffic accident detection methods have been proposed based on surveillance video, they detected accidents by analyzing whether there is a collision between target objects in image or video. For example, the work [44] proposes a real-time automated traffic accident detection using the Histogram of

Flow Gradient(HFG). It first extracts HFG-based features from video shots, then employs logistic regression to predict the probability of the occurrence of accidents. The work[49] considers an accident as an unusual incident, it proposes a denoising autoencoder framework to extract deep representation by only training over the normal traffic videos. Then, the possibility of an accident is determined based on the reconstruction error and the likelihood of the deep representation. The work [26] proposes an integrated two-streaming convolutional network architecture that consists of a spatial stream network for object detection and a temporal stream network for multiple-object tracking. Then, it detects accidents by incorporating appearance features and motion features from these two networks.

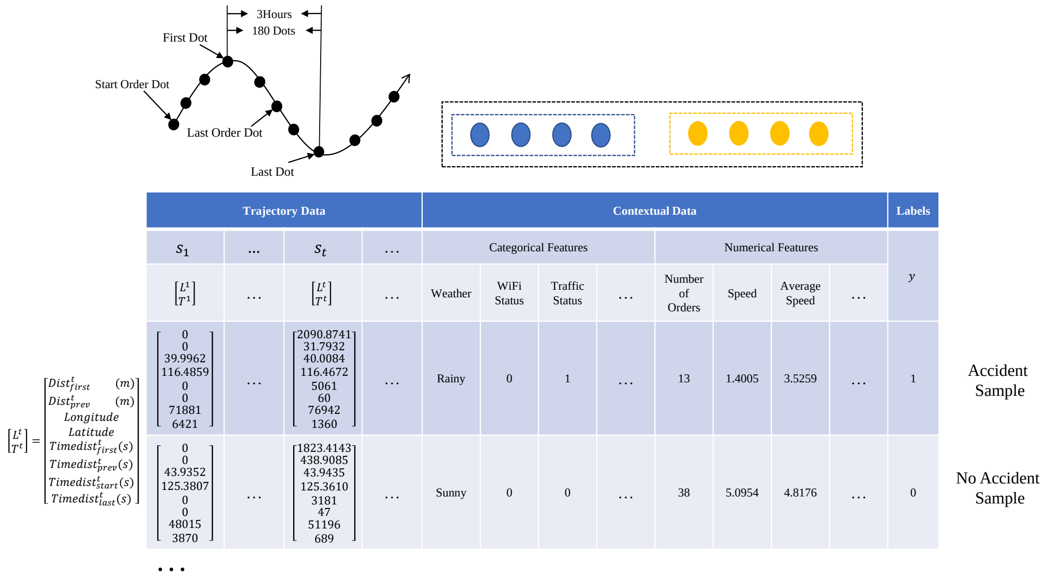
Previous works on traffic accident analysis consider accidents from a macro perspective, they forecast and detect the accidents of a certain domain, segment, or corner. To our best knowledge, our work is the first study on detecting personal traffic accidents based on individual trajectory and basic contextual data.

2.2 Trajectory Data Mining

Trajectory data mining [5, 19, 33, 35, 57] has become an increasingly important research theme, attracting attention from numerous areas, including computer science, sociology, and geography[64]. Our takeaway rider traffic accident detection problem is closely related to a typical kind of trajectory data mining task, i.e., trajectory classification. Generally, trajectory data are collected by location recording devices, and trajectory classification is divided into three steps, i.e., trajectory preparation, feature extraction, and classification [4]. For many existing classification models, the dimensions of input data are required to be the same so that they can be measured. However, the lengths of two arbitrary trajectories may be largely different from each other. Therefore trajectories are unified into a fixed length in a prepossessing step[23, 24, 62]. To reduce the information loss of unify, the work [60] segments trajectory into sub-trajectories with a fixed length in the preparation step. Sometimes, trajectory data is presented in the form of images or videos, which needs to track objects through all images to generate trajectory data for preparation [54, 62]. In feature extraction, spatial information is extracted to characterize trajectory data. Then, trajectory data are classified directly by measuring the spatial distance in previous works [32, 41]. However, in recent years, probabilistic inference models involve solving the problem by classifying trajectory data directly without feature extraction [2, 12]. The neural network has also been employed for trajectory classification, excellent deep learning models such as CNN and LSTM are used to construct an end-to-end deep structure for trajectory classification [47]. Besides trajectory classification for traffic accident detection, there are many other trajectory data mining methods for various traffic tasks. For example, the work [10, 14] uses a CNN architecture to predict travel modes based on raw GPS trajectories. The work [36] proposes an automated trajectory classification framework based on the Bi-LSTM model to classify raw trajectories into different transportation modes. The work [37] proposes a trajectory classifier called Spatio-Temporal GRU to better model the spatio-temporal correlations and irregular temporal intervals prevalently present in spatio-temporal trajectories. The work [29] proposes a Spatio-Temporal LSTM and extends it in an encoder-decoder manner which models the contextual historic visit information in order to boost the location prediction performance. The work [58] designs a deep learning model called LATL which not only adopts an adaptive attention network to model the distinct features of locations, but also implements time gates and distance gates into the LSTM network to capture the spatio-temporal relation between consecutive locations for destination prediction. The work [53] proposes a deep learning framework for travel time estimation. It presents a geo-convolution operation to capture spatial correlations between trajectory dots and captured temporal dependencies of trajectory dots by stacking recurrent units on the geo-convolution layer. The work [15] proposes an RNN-based semi-supervised model TULER, which exploits the trajectory data to capture the implicit semantics of user mobility patterns, and then identifies and links trajectories to the user who generates in the location based social networks. The work [39] builds person-specific mobility graphs by GPS trajectory data generated by personal smartphones, and then feeds them into GCNs to predict the destination of a user's visit at a certain location.

3 PRELIMINARIES

In this paper, we consider the traffic accident detection of takeaway riders based on the trajectory data and basic contextual data of riders. We denote the trajectory data as S, and the basic data as X. The data set for training consists of n instances $(\{S, X\}, y)$, where $y \in \{0, 1\}$ is the associated label indicating traffic accident happened to rider ($y = 1$ means an accident occurs, $y = 0$ otherwise). The task of traffic accident detection is to construct a model $\hat{y} = TAD_{model}(S, X)$ to predict whether an accident happened to the rider. To illustrate the format of our data, Fig. 2 shows the samples of the data, including both trajectory and contextual data. We describe the details of trajectory data and basic contextual data in following subsection.



Trajectory Data				Contextual Data								Labels	
s_1	…	s_t	…	Categorical Features				Numerical Features				y	
$\begin{bmatrix} L^1 \\ T^1 \end{bmatrix}$	…	$\begin{bmatrix} L^t \\ T^t \end{bmatrix}$	…	Weather	WiFi Status	Traffic Status	…	Number of Orders	Speed	Average Speed	…		
$\begin{bmatrix} 0 \\ 0 \\ 39.9962 \\ 116.4859 \\ 0 \\ 0 \\ 71881 \\ 6421 \end{bmatrix}$	…	$\begin{bmatrix} 2090.8741 \\ 31.7932 \\ 40.0084 \\ 116.4672 \\ 5061 \\ 60 \\ 76942 \\ 1360 \end{bmatrix}$	…	Rainy	0	1	…	13	1.4005	3.5259	…	1	Accident Sample
$\begin{bmatrix} 0 \\ 0 \\ 43.9352 \\ 125.3807 \\ 0 \\ 0 \\ 48015 \\ 3870 \end{bmatrix}$	…	$\begin{bmatrix} 1823.4143 \\ 438.9085 \\ 43.9435 \\ 125.3610 \\ 3181 \\ 47 \\ 51196 \\ 689 \end{bmatrix}$	…	Sunny	0	0	…	38	5.0954	4.8176	…	0	No Accident Sample

…

$$\begin{bmatrix} L^t \\ T^t \end{bmatrix} = \begin{bmatrix} Dist^t_{first} \quad (m) \\ Dist^t_{prev} \quad (m) \\ Longitude \\ Latitude \\ Timedist^t_{first}(s) \\ Timedist^t_{prev}(s) \\ Timedist^t_{start}(s) \\ Timedist^t_{last}(s) \end{bmatrix}$$

Fig. 2. Samples of data.

3.1 Trajectory Data

Trajectory data in our scenario is represented as a sequence of locations and times of trajectory dots $S = \{s_1, ..., s_t, ...\}$, where $s_t = \{L^t; T^t\}$. Specifically, $L = \{Dist^t_{first}; Dist^t_{prev}; Longitude^t; Latitude^t\}$

denotes the location features of trajectory, where $Dist^t_{first}$ represents the distance to the first dot and $Dist^t_{prev}$ represents the distance to the previous dot. $T = \{Timedist^t_{first}; Timedist^t_{prev}; Timedist^t_{start}; Timedist^t_{last}\}$ denotes the time features of trajectory, where $Timedist^t_{first}$ represents time distance to the first dot, $Timedist^t_{prev}$ represents time distance to the previous dot, and $Timedist^t_{start}$ and $Timedist^t_{last}$ represent time distance to the start and the last order respectively. To avoid inefficiency caused by useless information, we only record the GPS signals of the rider every minute in three hours around the last order, so the number of trajectory dot we collected $n_d = 180$, the trajectory data derived from GPS signals has the dimension of $8 \times n_d$, i.e. $S \in \mathbb{R}^{8 \times n_d}$.

3.2 Contextual Data

Contextual data consists of both the current features and historical statistic features formed as categories and numerical values. The categorical features contain the current status of the environment and mobile devices, such as weather, traffic congestion status, and WiFi connection status of rider's GPS device. The numerical features contain both current and historical statistical information about riders, such as the total number of orders, current speed, and the average speed. Each categorical feature is represented as a vector of one-hot embedding, and each numerical feature is represented as normalized value or the vector of one-hot embedding after discretization. In summary, contextual data has the dimension of 85, i.e., $X \in \mathbb{R}^{85}$.

4 THE TARA-NET MODEL

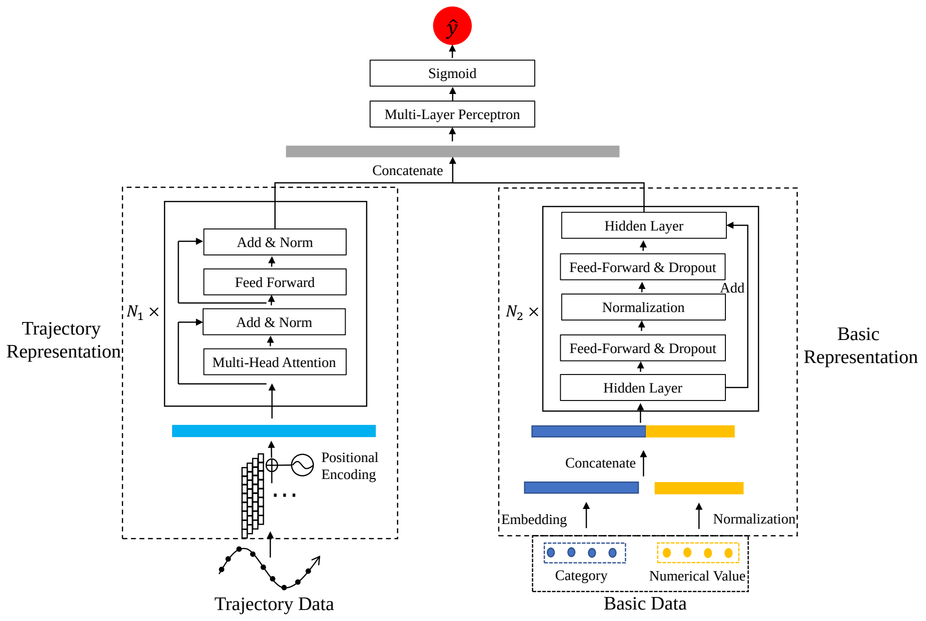


Fig. 3. The TRAR-Net model architecture. TARA-Net consists of a trajectory representation component, a contextual representation component, and a multi-layer perceptron module. The details of the three components are described in sect. 4.1,sect. 4.2 and sect. 4.3, respectively.

To utilize both trajectory data and contextual data for accident detection, we propose a fusion network named TARA-Net. As depicted in Fig. 3, TARA-Net consists of three components. A trajectory representation module (TR) is used to extract trajectory features. A basic representation module (BR) is used to extract basic contextual features, and a multi-layer perceptron module (MLP) analyzes concatenation features to detect whether an accident happened. The detection model can be formulated as follows,

$$\hat{y} = Sigmoid\ (MLP(\,TR\ (S), BR\ (X))). \tag{1}$$

4.1 The Trajectory Representation Component

Trajectory data is a sequence data. The representation of sequence data has been widely studied in the field of natural language processing[7, 13, 20]. Transformer [52] is an encoder-decoder model based on attention mechanisms used to produce word representation for the natural language process task. It only consists of self-attention and fully connected layers, dispensing with recurrence and convolutions entirely, which can make better use of both long term and short term information of sequential data. So we construct a transformer encoder by stacking multi-head attention mechanism and fully connected network to get a more comprehensive representation of trajectory data.

For a given trajectory with t dots $S = \{s_1, ..., s_t, ...\}$, where s_t is the vector representation of the t-th dot. To make use of the order of the sequence, we add "positional encodings" to inject positional information to the feature vectors of trajectory dots. The positional encodings have the same dimension as s_t, so we can add it to s_t by summed. There are many kinds of positional encodings, we use cosine and sine functions to calculate positional encodings for Odd dimension and Even dimension respectively.

$$PE_{(t,2i)} = sin(t/1000^{2i/d_{s_t}}) \tag{2}$$

$$PE_{(t,2i+1)} = cos(t/1000^{2i/d_{s_t}}), \tag{3}$$

where t is the position of trajectory dot, i is the index of positional encoding value and d_{s_t} represents the dimension of s_t. According to the definition of positional encoding, it can generate a vector of arbitrary dimension. In order to combine positional encoding $PE_{(t)}$ with s_t, we generate $PE_{(t)}$ in the same dimension as s_t. Then, the input of encoder can be represented as a matrix $M = (m_1, ..., m_t, ...), M \in \mathbb{R}^{d_m \times n_d}$, where $m_t = s_t + PE_{(t)}$.

The transformer encoder is composed of a stack of N identical layers, where each layer has two sublayers: Multi-head self attention mechanism and fully connected feed-forward network. We employ a residual connection [18] around each sublayer, which is then followed by a layer normalization [1]. The attention mechanism produces output based on the input Querys, Keys and Values. The output is computed as the weighted sum of the values where the weight assigned to each Value is computed by Query and the corresponding Key. In this work, we use self-attention mechanism that generates Querys, Keys and Values by only input M, i.e., $Q = MW^Q$,$K = MW^K$,$V = MW^V$, where $Q, K \in \mathbb{R}^{n_d \times d_k}$ and $V \in \mathbb{R}^{n_d \times d_v}$ are the feature matrices of Querys, Keys and Values, $W^Q, W^K \in \mathbb{R}^{d_m \times d_k}$ and $W^V \in \mathbb{R}^{d_m \times d_v}$ are parameter matrices. Then, we use "Scaled Dot-Product" to compute the output of attention as follows,

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{4}$$

To counteract the effect of small gradients brought by big d_k, we set $\frac{1}{\sqrt{d_k}}$ as a scaling factor and apply a softmax function to assign weights to Values.

Following the previous work[52], it's beneficial to linearly project the Query, Keys and Values h times with different learned parameter matrices, so we use Multi-head attention to replace the single attention. Multi-head attention can be thought as an ensemble of h single self-attentions, which allows the model to jointly attend to information from different representation subspaces. Formally, the formula can be written as follows,

$$MultiHead(M) = Concat(head_1, head_2, ..., head_h)W^O, \tag{5}$$

$$head_i = Attention(Q_i, K_i, V_i), \tag{6}$$

where Q_i, K_i, V_i are the representation matrices of the i-th self attention, and $W^O \in \mathbb{R}^{hd_v \times d_m}$ is the parameter matrix related to concatenate *heads*.

In addition to Multi-head attention, we append a fully connected feed-forward neural network after each attention module, which consists of two linear transformations with a ReLU activation in between.

$$FNN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{7}$$

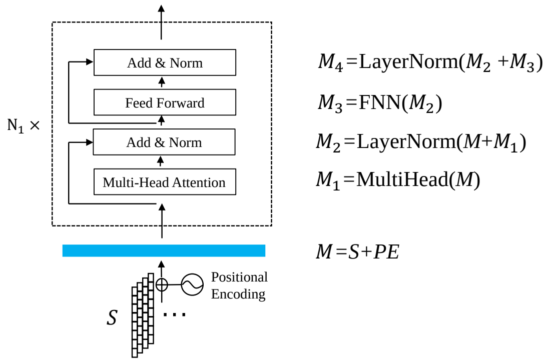


Fig. 4. Transfer process of each layer in transformer encoder.

In summary, the transformer encoder component extracts trajectory features by stacking N_1 identical Multi-head and FNN layers, the detailed transfer process of each layer are shown in Fig. 4. M_i denotes the output Matrix of layer i. We set $N_1 = 6$ in this work, so our encoder consists of 20 layers in total. Then, the output of 20-th layer M_{20} is the output of the encoder.

4.2 The Basic Contextual Representation Component

Contextual data of takeaway delivery has the characteristics of sparseness, which requires more complex functions to represent the implicit connections between sparse features. According to the universal approximation theorem[22], a single layer feed-forward network is enough to represent any function. But this layer may be very large, we need more parameters to represent a nonlinear function by single-layer than multi-layer, which causes efficiency issue. Furthermore, the large single-layer network is prone to be overfitting. To solve the above problems, researchers construct a network with more layers. Take image recognition models for example, since AlexNet [30], the latest networks become deeper and deeper. AlexNet has only 5 convolutional layers, while the subsequent VGG network [48] and GoogleNet [51] have 19 and 22 layers respectively. However, the performance of deep networks cannot be improved by simply stacking more layers. When training a deep neural network, the gradient is back-propagated to the previous layer, and the repeated multiplication of multilayers may cause the gradient infinitely small, which is also known as the vanishing gradient problem. As the number of layers of the network becomes larger, its performance tends to saturate and even begin to decline rapidly.

Residual Network [18] is one of the most pioneering work in the field of computer vision and deep learning, which can use deeper layers to improve representation ability. The core idea of the residual network is to introduce an "identity shortcut connection" to connect not only the previous layer but the layer before a certain stride with the current layer. The stride is a parameter adjusted according to the practical situation. The building block of the residual network is shown in Fig. 5.

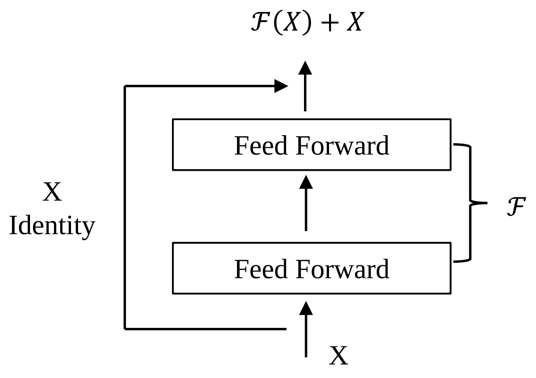


Fig. 5. The building block of a residual network.

Formally, the building block can be formulated as follows,

$$y = \mathcal{F}(X, \{W_i, b_i\}) + X, \tag{8}$$

where X and y are the input and output of the building block, W_i and b_i represent the weights and bias of i-th layer respectively. For the example in Fig. 5, the residual stride sets to be 2. Then, the mapping function is defined as $\mathcal{F} = W_2(W_1X + b_1) + b_2$. The operation $\mathcal{F} + X$ is performed by a shortcut connection and element-wise addition to retain the gradient of the deep network.

The basic contextual data features consist of the one-hot embedding of categorical data and the normalized numerical data, which have the characteristics of sparsity and small value. Therefore, using a deep network to automatically construct the intersections of sparse features and retain the gradient is particularly important for basic data features. To this end, we construct a residual network component to extract basic data features by stacking N_2 identical building blocks. Each building block consists of two hidden layers, two feed-forward layers, and one normalization layer, the detailed transfer process of each layer is shown in Fig. 6.

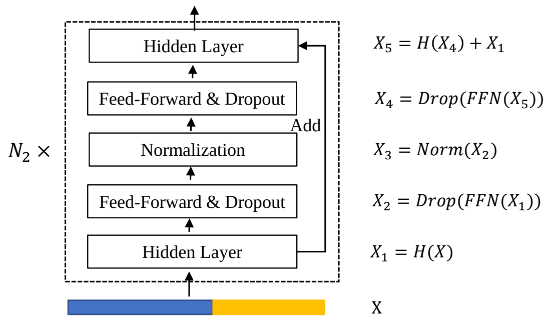


Fig. 6. Transfer process of each layer in a residual network.

X_i denotes the output of layer i. In order to avoid overfitting, we add dropout operations [50] to each feed-forward layer. We set $N_2 = 5$, so our residual network has a total of 30 layers, then the output of 25-th layer X_{25} is the output of the network.

4.3 The Multi-Layer Perceptron Component

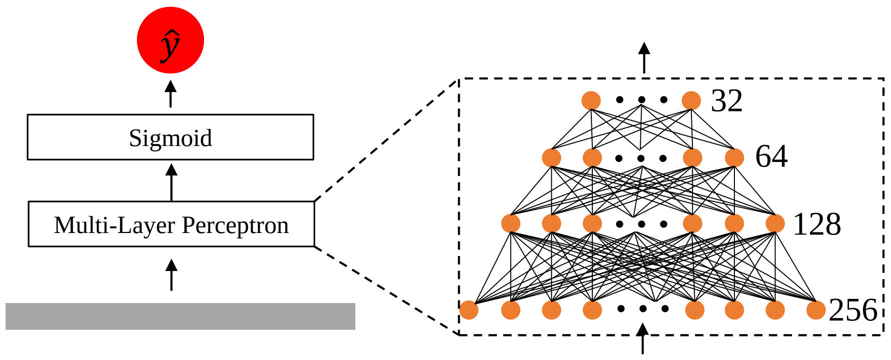


Fig. 7. Structure of multilayer perceptron.The integer values on the right hand side represent the number of neurons in each layer, and Sigmoid is an activation function $Sigmoid(x) = \frac{1}{1+e^{-x}}$, which can map any real number to the interval of $(0, 1)$.

The original trajectory data and basic contextual data are collected across the above two components. We obtain two corresponding representation vectors $M_{20}(-1, :)$ and X_{25}, where $M_{20}(-1, :)$ represents the last vector of the matrix M_{20}. Aiming at predicting the probability of traffic accident by the concatenation of $M_{20}(-1,:)$ and X_{25}, we employ a pyramid-shaped multilayer perceptron to connect high-dimensional features and one-dimensional output. The structure of MLP is shown in Fig. 7. The predict probability of traffic accident can be calculated by,

$$\hat{y} = Sigmoid(MLP(Concat(M_{20}(-1,:), X_{25}))) \tag{9}$$

4.4 Model Training

The proposed TARA-Net model is an end-to-end deep model stacked by the above three components, its parameters can be easily updated by backpropagating the loss of predict results. We train the TARA-Net model by minimizing the average cross-entropy defined as follows,

$$L = \frac{1}{n}\sum_{i=1}^{n} -(y \cdot log(\hat{y}) + (1-y) \cdot log(1-\hat{y})), \tag{10}$$

where n is the number of training samples, $\hat{y}$ is the predicted probability of an accident, and y is the ground truth of whether an accident happened, $y = 1$ represents an accident happened, otherwise $y = 0$. However, the positive examples $(y = 1)$ is far less than the negative examples $(y = 0)$ in our takeaway rider traffic accident detection scenario, which is always referred to as a class imbalance problem. Also, the magnitude of training data is not large enough, which may result in an overfitting problem. Therefore, we exploit an oversampling for positive examples to alleviate class imbalance, and employ a novel regularization method called flooding that keeps the training loss to stay around a small constant value, to avoid zero training loss.

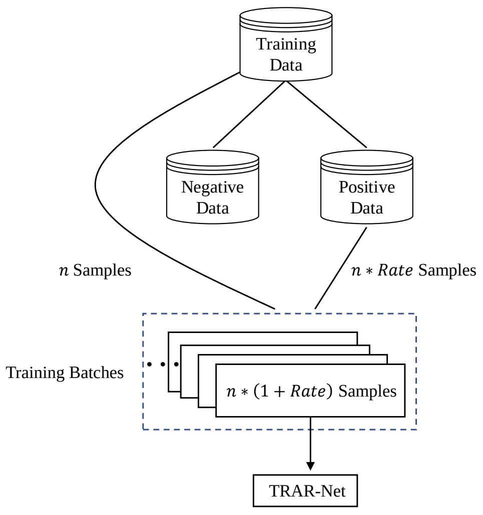


Fig. 8. Sampling Procedure

4.4.1 *Oversampling for positive examples.*

The sampling procedure is shown in Fig. 8. For each training batch, we randomly extracting n samples from the original training data and $n * Rate$ samples from the positive data. As a result, the data of each batch has a similar number of positive and negative samples, which is very beneficial to the training of TARA-Net. The training loss can be rewritten as:

$$L = \frac{1}{n * (1 + Rate)} \sum_{i=1}^{n*(1+Rate)} -(y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})), \tag{11}$$

where $n * (1 + Rate)$ is the number of oversampled training samples.

4.4.2 Flooding. To avoid overfitting, *flooding* prevents further reduction of the training loss when it reaches a reasonably small value, which is named the *flooding level*. The loss function after *flooding* can be rewritten as follows,

$$\tilde{L} = |L - b| + b. \tag{12}$$

Note that when $b = 0$, then $\tilde{L} = L$. $\tilde{L}$ has the same gradient direction as L when $L > b$ but the opposite direction when $L < b$. This means that we perform gradient descent when the training loss is above the *flooding level*, but gradient ascent otherwise.

5 EXPERIMENTS

We evaluate the effectiveness of our proposed model TARA-net on two tasks: transportation model classification and takeaway rider accident detection. The experiments of transportation model classification illustrate the advantage of our trajectory representation component over other trajectory mining methods. The experiments of accident detection show the superiority of our fusion network TARA-Net.

5.1 Transportation Mode Classification

5.1.1 Dataset. Geolife [65–67] is a trajectory dataset commonly used in data mining community which was collected by Microsoft Asia. This dataset contains 17621 trajectories of which about 8000 have transportation mode labels (4 types, bike, walk, car and bus). According to work [37], we split Geolife in a ratio of 7:1:2 to get corresponding training, validation, and test datasets.

Table 1. Experimental results for classification accuracy on Geolife

	SVM	CNN	LSTM	GRU	Conv-LSTM	ST-GRU	TARA-Net
Accuracy	86.11%	87.08%	88.39%	89.76%	89.85%	91.25%	**92.63%**

5.1.2 Performance Evaluation. We compare our TARA-Net to the baselines including both maching learning based and RNN based approaches. Notably, our model removes the contextual representation component in this classification task. Table 1 clearly demonstrates that our TARA-Net outperforms all baselines on transportation mode classification. We have the following observations:

- The RNN based methods (LSTM [21], GRU [8], Conv-LSTM [46], ST-GRU [37]) are superior to the traditional machine learning methods (SVM [9], CNN [31]). This is because the model with RNN structure can better extract the implicit features of spatio-temporal sequence data.

- Our TARA-Net outperforms all baselines. Since we remove the contextual representation component, it shows the superiority of our trajectory representation component over other trajectory representation methods.

5.2 Takeaway Rider Accident Detection

5.2.1 Dataset. We evaluate the performance of takeaway rider accident detection on real-world takeaway rider data from *Ele. me*. The dataset includes 154,097 instances of thousands of riders in a month. Each instance is represented as a tuple (S, X, y), where S is the trajectory data of a rider, X is the basic contextual data about the rider and environment, and $y \in \{0, 1\}$ is the label that indicates whether (S, X) corresponds to an accident. The dataset includes 74.4% negative instances $(y = 0)$ and 15.6% positive instances $(y = 1)$. According to the grade of the accident, the positive instances can be subdivided into the serious accident denoted as P_0 and general accident represented by P_1. We split the dataset into two parts, i.e., the data of the first three weeks is used for training, while the data of the remaining week is used for testing. The details of original and oversampled data are shown in Table 2.

Table 2. Distribution of original and oversampled data. We only oversample on Positive instances, so the negative instances of the original training data are the same as the negative instances of the oversampled data and testing data did not change.

Dateset	Neg	P_0	P_1
Training data	104237	700	18706
Oversampled training data	104237	1598	42537
Testing data	25819	165	4470

5.2.2 Evaluation Metrics. In this paper, we use five metrics to measure the performance of our method. AUC is defined as the area under the ROC curve. In other words, we randomly select a positive sample and a negative sample from the testing set, and AUC represents the probability that the predicted value of the positive sample is greater than the negative sample. The formulation of AUC is as follows,

$$AUC = \frac{\sum pred_{pos} > pred_{neg}}{positiveNum * negativeNum} \tag{13}$$

The traffic accident task is a binary classification. Recall and Precision are often used to measure the performance of binary classification methods. Recall represents the fraction of positive samples that are predicted correctly and precision represents the fraction of positive samples that are predicted to be positive. Formally, they are calculated as follows,

$$Recall = \frac{TP}{TP + FN}, \tag{14}$$

$$Precision = \frac{TP}{TP + FP}, \tag{15}$$

where the meaning of TP, FN, FP are shown in the confusion matrix in Table 3.

Since the output of our model is a continuous probability value, we need to set a threshold to generate the prediction category label, then calculate Precision and Recall. However, it's not easy to choose a fair threshold for all comparison methods. We further compare maximum of $F1$ and K-S (Kolmogorov-Smirnov)

Table 3. Confusion matrix of classification

True Label	Prediction Label	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

value on models and the corresponding precision and recall. The formulations of $Max\text{-}F1$ and $K\text{-}S$ are shown as follows,

$$Max\text{-}F1 = max(\frac{2 \times Precision \times Recall}{Precision + Recall}). \tag{16}$$

$$K\text{-}S = max(TPR - FPR), \tag{17}$$

where $FPR = \frac{FP}{FP+TN}$ and $TPR = \frac{TP}{FN+TP}$.

5.2.3 Model Comparison. We compare five models in our experiments: LR[28], FNN[3], Wide and Deep[6], XDeepFM[34], and TARA-Net. Since the previous traffic accident analysis method focused on the accident prediction of a certain area, such as a corner or a road segment, which can be considered as the regression of accident risk in the region. But our task is more like a binary classification based on trajectory data and sparse contextual data. So we compare our method with the classic and excellent models mentioned above, they perform well in many binary classification tasks, such as Click Through Rate prediction of the recommender system. It's worth noting that we add shortcut connections to FNN to avoid vanishing gradient, and compose Wide and Deep model by LR and FNN.

5.2.4 Parameter Settings. To evaluate the models on *Ele. me* dataset. To obtain fair comparison, we set the number of the epoch to be 10 and batch size to be 64 for all the models. For all the deep models, we use Adam[27] as the optimizer, Relu[16] as the activation function, the dimension of the hidden layer, and the dropout are set as 256 and 0.8 respectively. Also, we set $flooding = 0.1$ for our TARA-Net.

5.2.5 Performance Evaluation. We evaluate the models from three aspects, i.e., the performance on accident, the performance on P_0 accident, and the performance on P_1 accident. In the next two experimental settings, we remove P_0 or P_1 accident to measure the detection performance for P_1 accident and P_0 accident respectively.

Table 4. Performance on accidents

Metrics	LR	FNN	Wide and Deep	XDeepFM	TARA-Net
AUC	0.6470	0.8562	0.8830	0.8721	**0.8881**
Max-F1	0.3363	0.6089	0.6800	0.6581	**0.6938**
Max-F1-Precision	0.2611	0.6458	0.7809	0.7891	**0.7957**
Max-F1-Recall	0.4179	0.5814	0.6022	0.5644	**0.6151**
K-S	0.2478	0.5650	0.6212	0.5958	**0.6327**
K-S-Precision	0.2276	0.4815	**0.5289**	0.5204	0.5020
K-S-Recall	0.6334	0.7001	0.7392	0.7137	**0.7696**

Table 5. Performance on P_0 accidents

Metrics	LR	FNN	Wide and Deep	XDeepFM	TARA-Net
AUC	0.6560	0.8875	0.9202	0.8861	**0.9236**
Max-F1	0.0285	0.1409	0.3499	**0.4580**	0.3494
Max-F1-Precision	0.0135	0.0663	0.2440	**0.6186**	0.2408
Max-F1-Recall	0.0545	0.6364	0.6182	0.3636	**0.6364**
K-S	0.2542	0.6281	0.7313	0.6387	**0.7321**
K-S-Precision	0.0139	0.0275	**0.0599**	0.0525	0.0484
K-S-Recall	0.4545	0.7879	0.8061	0.7152	**0.8303**

Table 6. Performance on P_1 accidents

Metrics	LR	FNN	Wide and Deep	XDeepFM	TARA-Net
AUC	0.6467	0.8550	0.8816	0.8716	**0.8868**
Max-F1	0.3300	0.6039	0.6747	0.6541	**0.6885**
Max-F1-Precision	0.2534	0.6367	0.7735	0.7823	**0.7886**
Max-F1-Recall	0.4163	0.5794	0.5982	0.5619	**0.6109**
K-S	0.2493	0.5628	0.6176	0.5948	**0.6296**
K-S-Precision	0.2217	0.4717	**0.5210**	0.5110	0.4919
K-S-Recall	0.6349	0.6980	0.7342	0.7128	**0.7664**

The results of experiments from three aspects are shown in Table 4, Table 5, and Table 6. We highlight the best results in the tables. After a comprehensive comparison, we have the following observations:

- First, deep models have better performance than the shallow models on traffic accident detection. Obviously, we can see that LR (which is the only shallow model)performs the worst among all comparison methods in all metrics. Take AUC value as example, a simple deep model FNN outperforms LR by 32.33%, 35.29%, 32.21% on entire accident, $P0$ accident and $P1$ accident respectively. While the best deep model TARA-Net outperforms LR by 37.26%, 40.79%, 37.13%.
- Second, fusion deep models perform better than single deep models on traffic accident detection. It's worth noting that our wide and deep model only consists of two modules LR and FNN. However, its performance is greatly improved compared with LR and FNN from all aspects. The most significant improvement is the $Max\text{-}F1$ value of $P0$ accident, wide and deep model improves 1127.72% , 148.33% than LR and FNN on $Max\text{-}F1$ of P0 accident.
- Third, transformer encoder module improves the performance of traffic accident detection by trajectory data. The difference between wide and deep model and TARA-Net is that we replace wide LR of wide and deep by deep transformer encoder of TARA-Net, which brings comprehensive improvements to model performance except for the *K-S-Precision* metric. However, the precision and recall are a pair of negatively correlated metrics, TARA-Net performs 5.36% worse than wide and deep on *K-S-Precision*, but improve 4.11% , 1.85% on *K-S-Recall* and *K-S* respectively.
- Fourth, in our traffic accident detection scenario, the threshold determined by *K-S* corresponds to a higher recall, while the threshold determined by $Max\text{-}F1$ corresponds to higher precision. *K-S-Recall* is 25.12%, 30.47%, 25.45% higher than $Max\text{-}F1\text{-}Recall$ on three experiments respectively. In a real

business scenario, low precision means more human resources are needed to further filter accidents, while low recall means we will miss more accidents, which is considered more serious than the waste of human resources. So the threshold derived from K-S value can provide a higher recall with acceptable precision. It is more suitable for our traffic accident detection task.

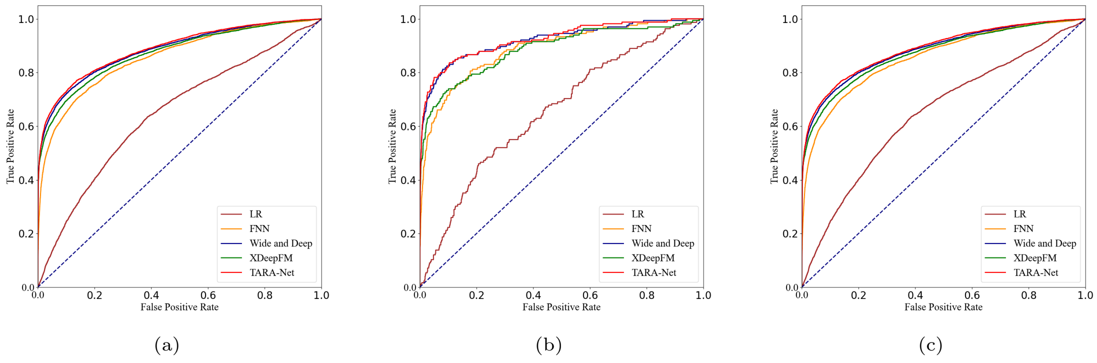


Fig. 9. Roc curve of TARA-Net. (a) represents the Roc curve of on total accident data, (b) and (c) represent the Roc curve of P_0 accident and P_1 accident respectively.

We show the Roc curve of experiments in Fig. 9. The figures also show the performance comparison of the models intuitively. In summary, the performance ranking is TARA-Net>Wide and Deep>XDeepFM>FNN>LR. Our method TARA-Net consistently performs better than the other benchmark methods.

Table 7. TARA-Net performance with different N_1

Metrics	Accident			P_0 Accident			P_1 Accident		
	N_1=3	N_1=5	N_1=7	N_1=3	N_1=5	N_1=7	N_1=3	N_1=5	N_1=7
AUC	0.8837	**0.8861**	0.8829	**0.9261**	0.9220	0.9204	0.8821	**0.8848**	0.8815
Max-F1	0.6838	**0.6867**	0.6863	0.3382	**0.3669**	0.3597	0.6781	**0.6815**	0.6811
K-S	0.6268	**0.6303**	0.6277	**0.7479**	0.7397	0.7463	0.6229	**0.6266**	0.6243

5.2.6 Selection of N_1 and N_2. In order to find reasonable numbers of building blocks of the transformer encoder and residual network N_1 and N_2. We choose N from $\{3, 5, 7\}$, the experimental results are shown in Table 7. We can see that our TARA-Net gets better performance when $N_1 = 5$. The experiments on N_2 has similar results, and we omit it here for brevity. Finally, we choose $N_0 = 5$ and $N1 = 5$ for our TARA-Net.

5.2.7 Selection of Oversampling Rate. To choose the appropriate oversampling rate, we analyze AUC variation about $Rate \in [0, 0.5]$, with the step as 0.05, the experimental result is shown in Fig. 10. We can see that there is a peak at $Rate = 0.2$. Although the AUC value still improves when $Rate > 0.4$, but the time consumption of model training will increase as the value of $Rate$ increases. Consider both accuracy and efficiency, we set $Rate = 0.2$ in this work.

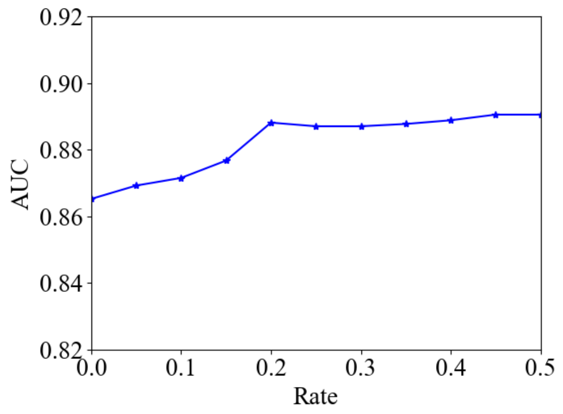


Fig. 10. AUC variation about oversampling rate $Rate$

6 CONCLUSIONS

In this paper, we propose TARA-Net, a fusion network for takeaway rider accident detection. TARA-Net consists of a transformer encoder component for trajectory feature representation, a residual network for sparse contextual feature representation, and a full connection feed-forward neural network to output the detection results. Furthermore, to overcome the challenge of class imbalance, we develop an over-sampling method to strengthen positive samples. We also introduce flooding to avoid the overfitting of the proposed deep model TARA-Net. We conduct extensive experiments on real-world *Ele. me* takeaway rider data to compare the effectiveness of TARA-Net and the baselines. Our experiment results demonstrate that our model constantly performs better than the other benchmark models in terms of serious accident detection and normal accident detection results.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Miguel Ángel Bautista, Antonio Hernández-Vela, Sergio Escalera, Laura Igual, Oriol Pujol, Josep Moya, Verónica Violant, and María Teresa Anguera. 2016. A Gesture Recognition System for Detecting Behavioral Patterns of ADHD. *IEEE Trans. Cybern.* 46, 1 (2016), 136–147.

[3] George Bebis and Michael Georgiopoulos. 1994. Feed-forward neural networks. *IEEE Potentials* 13, 4 (1994), 27–31.

[4] Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. 2019. Trajectory data classification: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 4 (2019), 1–34.

[5] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 900–911.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 1724–1734.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation.

arXiv preprint arXiv:1406.1078 (2014).

[9] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[10] Sina Dabiri and Kevin Heaslip. 2018. Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transportation research part C: emerging technologies* 86 (2018), 360–371.

[11] Rupam Deb and Alan Wee-Chung Liew. 2016. Missing value imputation for the analysis of incomplete traffic accident data. *Information sciences* 339 (2016), 274–289.

[12] Maxime Devanne, Stefano Berretti, Pietro Pala, Hazem Wannous, Mohamed Daoudi, and Alberto Del Bimbo. 2017. Motion segment decomposition of RGB-D sequences for human behavior understanding. *Pattern Recognit.* 61 (2017), 222–233.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[14] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Akihisa Kawanobe. 2016. Deep feature extraction from trajectories for transportation mode estimation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 54–66.

[15] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying Human Mobility via Trajectory Embeddings.. In *IJCAI*, Vol. 17. 1689–1695.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[17] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 1725–1731.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[19] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Chao Tian, and Yu Zheng. 2018. Detecting Vehicle Illegal Parking Events using Sharing Bikes' Trajectories.. In *KDD*. 340–349.

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[22] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[23] Han Hu, Jianjiang Feng, and Jie Zhou. 2014. Exploiting unsupervised and supervised constraints for subspace clustering. *IEEE transactions on pattern analysis and machine intelligence* 37, 8 (2014), 1542–1557.

[24] Weiming Hu, Xi Li, Guodong Tian, Stephen Maybank, and Zhongfei Zhang. 2013. An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE transactions on pattern analysis and machine intelligence* 35, 5 (2013), 1051–1065.

[25] Chao Huang, Chuxu Zhang, Peng Dai, and Liefeng Bo. 2019. Deep Dynamic Fusion Network for Traffic Accident Forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2673–2681.

[26] Xiaohui Huang, Pan He, Anand Rangarajan, and Sanjay Ranka. 2020. Intelligent Intersection: Two-stream Convolutional Networks for Real-time Near-accident Detection in Traffic Video. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 6, 2 (2020), 1–28.

[27] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[28] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. 2002. *Logistic regression*. Springer.

[29] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction.. In *IJCAI*, Vol. 18. 2341–2347.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[32] Xi Li, Weiming Hu, and Wei Hu. 2006. A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering. In *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*. IEEE Computer Society, 591–594.

[33] Yuhong Li, Jie Bao, Yanhua Li, Yingcai Wu, Zhiguo Gong, and Yu Zheng. 2016. Mining the most influential k-location set from massive trajectories. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances*

in Geographic Information Systems. 1–4.

[34] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 1754–1763.

[35] Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. 2016. Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 1–10.

[36] Hongbin Liu and Ickjai Lee. 2017. End-to-end trajectory transportation mode classification using Bi-LSTM recurrent neural network. In *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. IEEE, 1–5.

[37] Hongbin Liu, Hao Wu, Weiwei Sun, and Ickjai Lee. 2019. Spatio-temporal GRU for trajectory classification. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1228–1233.

[38] Jianming Lv, Qing Li, Qinghui Sun, and Xintong Wang. 2018. T-CONV: A Convolutional Neural Network for Multi-scale Taxi Trajectory Prediction. In *IEEE International Conference on Big Data and Smart Computing*. 82–89.

[39] Henry Martin, Dominik Bucher, Esra Suel, Pengxiang Zhao, Fernando Perez-Cruz, and Martin Raubal. 2018. Graph convolutional neural networks for human activity purpose imputation. In *NIPS spatiotemporal workshop at the 32nd Annual conference on neural information processing systems (NIPS 2018)*.

[40] Randa Oqab Mujalli, Griselda López, and Laura Garach. 2016. Bayes classifiers for imbalanced traffic accidents datasets. *Accident Analysis & Prevention* 88 (2016), 37–51.

[41] Mirco Nanni and Dino Pedreschi. 2006. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.* 27, 3 (2006), 267–289.

[42] VA Olutayo and AA Eludire. 2014. Traffic accident analysis using decision trees and neural networks. 2 (2014), 22–28.

[43] Honglei Ren, You Song, Jingxin Liu, Yucheng Hu, and Jinzhi Lei. 2017. A Deep Learning Approach to the Prediction of Short-term Traffic Accident Risk. *CoRR* abs/1710.09543 (2017). arXiv:1710.09543 http://arxiv.org/abs/1710.09543

[44] Samy Sadeky, Ayoub Al-Hamadiy, Bernd Michaelisy, and Usama Sayed. 2010. Real-time automatic traffic accident recognition using hfg. In *2010 20th International Conference on Pattern Recognition*. IEEE, 3348–3351.

[45] Bharti Sharma, Vinod Kumar Katiyar, and Kranti Kumar. 2016. Traffic accident prediction model using support vector machines with Gaussian kernel. In *Proceedings of fifth international conference on soft computing for problem solving*. Springer, 1–10.

[46] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214* (2015).

[47] Yemin Shi, Yonghong Tian, Yaowei Wang, and Tiejun Huang. 2017. Sequential deep trajectory descriptor for action recognition with three-stream CNN. *IEEE Transactions on Multimedia* 19, 7 (2017), 1510–1520.

[48] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[49] Dinesh Singh and Chalavadi Krishna Mohan. 2018. Deep spatio-temporal representation for detection of road accidents using stacked autoencoder. *IEEE Transactions on Intelligent Transportation Systems* 20, 3 (2018), 879–887.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[53] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2500–2507.

[54] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. 2011. Action recognition by dense trajectories. In *CVPR 2011*. IEEE, 3169–3176.

[55] Jing Wang and Xin Wang. 2011. An ontology-based traffic accident risk mapping framework. In *International Symposium on Spatial and Temporal Databases*. Springer, 21–38.

[56] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[57] Guojun Wu, Yichen Ding, Yanhua Li, Jie Bao, Yu Zheng, and Jun Luo. 2017. Mining spatio-temporal reachable regions over massive trajectory data. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 1283–1294.

[58] Jiajie Xu, Jing Zhao, Rui Zhou, Chengfei Liu, Pengpeng Zhao, and Lei Zhao. 2019. Destination prediction a deep learning based approach. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[59] Kaiping Xu, Zheng Qin, Guolong Wang, Kai Huang, Shuxiong Ye, and Huidi Zhang. 2018. Collision-Free LSTM for Human Trajectory Prediction. In *Proceedings of International Conference on MultiMedia Modeling*. Springer International Publishing, Cham, 106–116.

[60] Yuan Yuan, Yachuang Feng, and Xiaoqiang Lu. 2016. Statistical hypothesis detector for abnormal event detection in crowded scenes. *IEEE transactions on cybernetics* 47, 11 (2016), 3597–3608.

[61] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 984–992.

[62] Tianzhu Zhang, Hanqing Lu, and Stan Z Li. 2009. Learning semantic scene models by object classification and trajectory clustering. In *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 1940–1947.

[63] Yang Zhang, Xiangyu Dong, Lanyu Shang, Daniel Zhang, and Dong Wang. 2020. A multi-modal graph neural network approach to traffic risk forecasting in smart urban sensing. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[64] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 1–41.

[65] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. 312–321.

[66] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[67] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web*. 791–800.

[68] Zhengyang Zhou, Yang Wang, Xike Xie, Lianliang Chen, and Hengchang Liu. 2020. RiskOracle: A Minute-Level Citywide Traffic Accident Forecasting Framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1258–1265.